macromedia®

# FLASH® 8

## Guide de référence du langage ActionScript 2.0

**Remerciements**

Gestion de projet : JuLee Burdekin

Principaux rédacteurs : Francis Cheng, Robert Dixon, Shimul Rahim

Autres rédacteurs : Jen deHaan, Thais Derich, Guy Haas, David Jacowitz, Jeff Swartz

Exemples développés par : Luke Bayes, Francis Cheng, Robert Dixon, Ali Mills, Jeff Swartz

Révision : Linda Adler, Geta Carlson, Evelyn Eldridge, John Hammett, Noreen Maher, Mark Nigara, Lisa Stanziano, Anne Szabla, Jessie Wood

Gestion de la production : Patrice O'Neill

Conception et production : Adam Barnett, John Francis, Brett Jarvis, Mario Reynoso

Remerciements particuliers à : Peter deHaan, Gary Grossman, Lee Thomason et l'équipe Flash Player

Première édition : Septembre 2005

Macromedia, Inc.
601 Townsend St.
San Francisco, CA 94103
Etats-Unis

# Table des matières

CHAPITRE 1

# Eléments du langage ActionScript

1

Cette section fournit des informations sur la syntaxe, l'utilisation et des exemples de code concernant les fonctions et les propriétés globales (ces éléments n'appartenant pas à une classe ActionScript), ainsi que les directives de compilation ; pour les constantes, elle décrit les opérateurs, instructions et mots-clés utilisés dans ActionScript et définis par la spécification de langage ECMAScript (ECMA-262), version 4.

## Directives de compilation

Cette section regroupe les directives à inclure dans votre fichier ActionScript pour demander au compilateur de prétraiter certaines instructions. Ne placez pas de point-virgule (;) à la fin de la ligne qui contient la directive.

### Résumé des directives de compilation

| Directive | Description |
|---|---|
| `#endinitclip` | Indique la fin d'un bloc d'actions d'initialisation. |
| `#include` | Inclut le contenu du fichier spécifié, comme si les commandes du fichier faisaient partie du script d'appel. |
| `#initclip` | Indique le début d'un bloc d'actions d'initialisation. |

## Directive #endinitclip

`#endinitclip`

Indique la fin d'un bloc d'actions d'initialisation.

*Ne placez pas de point-virgule (;) à la fin de la ligne qui contient la directive #endinitclip.*

**Disponibilité :** Flash Player 6.0 ; ActionScript 1.0

### Exemple

```
#initclip
```

```
...initialization actions go here...
#endinitclip
```

## Directive #include

*#include "[path]filename.as":String*

Inclut le contenu du fichier spécifié, comme si les commandes du fichier faisaient partie du script d'appel. La directive #include est appelée lors de la compilation. Cela signifie que si vous apportez des modifications à un fichier externe, vous devez enregistrer le fichier et recompiler tous les fichiers FLA qui l'utilisent.

Si vous utilisez le bouton Vérifier la syntaxe pour un script contenant des instructions #include, la syntaxe des fichiers inclus est également vérifiée.

Vous pouvez utiliser #include dans des fichiers FLA et dans des fichiers de script externes, mais pas dans les fichiers de classe ActionScript 2.0.

Vous pouvez omettre le chemin, ou spécifier un chemin relatif ou absolu pour le fichier à inclure. Si vous ne spécifiez pas de chemin, le fichier AS doit figurer dans l'un des emplacements suivants :

- Le même répertoire que le fichier FLA. Le même répertoire que le script contenant l'instruction #include
- Le répertoire Include global, qui peut prendre l'une des formes suivantes : --Windows 2000 ou Windows XP : C:\Documents and Settings*utilisateur* \Local Settings\ Application Data\Macromedia\Flash 8*langue*\Configuration\Include --Macintosh OS X : Disque dur/Users/Library/Application Support/Macromedia/Flash 8/*langue*/ Configuration/Include
- Le répertoire *programme Flash 8**langue*\First Run\Include. Si vous enregistrez un fichier à cet endroit, il est copié dans le répertoire Include global au prochain démarrage de Flash.

Pour spécifier un chemin relatif pour le fichier AS, placez un point (.) pour représenter le répertoire actuel, deux points (. .) pour représenter le répertoire parent et une barre oblique (/) pour représenter les sous-répertoires. Consultez les exemples suivants :

Pour spécifier un chemin absolu pour le fichier AS, appliquez le format correspondant à votre plate-forme (Macintosh ou Windows). Consultez les exemples suivants : (Cette utilisation n'est pas recommandée car elle nécessite une structure de répertoires identique sur l'ordinateur servant à compiler le script.)

*Si vous placez des fichiers dans le répertoire First Run/Include ou dans le répertoire global Include, sauvegardez ces fichiers. En effet, si vous devez désinstaller et réinstaller Flash, ces répertoires risquent d'être supprimés ou remplacés.*

*Ne placez pas de point-virgule (;) à la fin de la ligne qui contient la directive #include.*

**Disponibilité :** Flash Player 4.0 ; ActionScript 1.0

### Paramètres

*[path]filename.as*:String - *filename.as*Le nom de fichier et le chemin facultatif du script à ajouter au panneau Actions ou au script actuel. Nous recommandons d'utiliser l'extension *.as*.

### Exemple

Les exemples suivants indiquent différentes manières de spécifier un chemin pour un fichier à inclure dans votre script :

```
// Note that #include statements do not end with a semicolon (;)
// AS file is in same directory as FLA file or script
// or is in the global Include directory or the First Run/Include directory
#include "init_script.as"

// AS file is in a subdirectory of one of the above directories
// The subdirectory is named "FLA_includes"
#include "FLA_includes/init_script.as"
// AS file is in a subdirectory of the script file directory
// The subdirectory is named "SCRIPT_includes"
#include "SCRIPT_includes/init_script.as"
// AS file is in a directory at the same level as one of the above
  directories
// AS file is in a directory at the same level as the directory
// that contains the script file
// The directory is named "ALL_includes"
#include "../ALL_includes/init_script.as"

// AS file is specified by an absolute path in Windows
// Note use of forward slashes, not backslashes
#include "C:/Flash_scripts/init_script.as"

// AS file is specified by an absolute path on Macintosh
#include "Mac HD:Flash_scripts:init_script.as"
```

## Directive #initclip

```
#initclip [order:Number]
```

Indique le début d'un bloc d'actions d'initialisation. Lorsque plusieurs clips sont initialisés simultanément, vous pouvez utiliser le paramètre `order` pour spécifier l'initialisation devant se produire en premier. Les actions d'initialisation sont exécutées lorsqu'un symbole de clip est défini. Si le clip est un symbole exporté, les actions d'initialisation sont exécutées avant les actions de l'image 1 du fichier SWF. Sinon, elles sont exécutées immédiatement avant les actions portant sur l'image qui contient la première occurrence du symbole de clip correspondant.

Les actions d'initialisation sont exécutées uniquement après la lecture d'un fichier SWF. Utilisez-les pour les initialisations uniques, telles que la définition de classe et l'enregistrement.

*Ne placez pas de point-virgule (;) à la fin de la ligne qui contient la directive #initclip.*

**Disponibilité :** Flash Player 6.0 ; ActionScript 1.0

## Paramètres

*`order`*`:Number` [facultatif] - Un entier non négatif spécifiant l'ordre d'exécution des blocs de code `#initclip`. Ce paramètre est facultatif. Vous devez spécifier la valeur à l'aide d'un littéral entier (seules les valeurs décimales sont autorisées, et non pas les valeurs hexadécimales), et non à l'aide d'une variable. Si vous incluez plusieurs blocs `#initclip` dans un symbole de clip unique, le compilateur utilise alors la dernière valeur `order` spécifiée dans ce symbole de clip pour tous les blocs `#initclip` de ce symbole.

## Exemple

Dans l'exemple suivant, le code ActionScript est placé sur l'image 1 au sein d'une occurrence de clip. Un fichier texte variables.txt est placé dans le même répertoire.

```
#initclip

trace("initializing app");

var variables:LoadVars = new LoadVars();

variables.load("variables.txt");

variables.onLoad = function(success:Boolean) {

 trace("variables loaded:"+success);

 if (success) {
 for (i in variables) {
 trace("variables."+i+" = "+variables[i]);
 }
 }
};
```

```
#endinitclip
```

# Constantes

Une constante est une variable qui représente une propriété dont la valeur ne change jamais. Cette section décrit des constantes globales qui sont disponibles pour tous les scripts.

## Résumé des constantes

| Modificateurs | Constante | Description |
|---|---|---|
| | `false` | Valeur booléenne unique qui représente l'opposé de `true`. |
| | `Infinity` | Spécifie la valeur IEEE-754 représentant l'infini positif. |
| | `-Infinity` | Spécifie la valeur IEEE-754 représentant l'infini négatif. |
| | `NaN` | Variable prédéfinie incluant la valeur IEEE-754 pour NaN (n'est pas un nombre). |
| | `newline` | Insère un caractère de retour chariot (`\r`) qui insère une ligne vierge dans le texte généré par votre code. |
| | `null` | Valeur spéciale qui peut être affectée à des variables ou renvoyée par une fonction en l'absence de données. |
| | `true` | Valeur booléenne unique qui représente l'opposé de `false`. |
| | `undefined` | Une valeur spéciale, qui indique généralement qu'une variable n'a pas encore reçu de valeur. |

## Constante false

Valeur booléenne unique qui représente l'opposé de `true`.

Lorsque le typage automatique de données convertit `false` en nombre, il renvoie 0. Lorsqu'il convertit `false` en chaîne, il renvoie `"false"`.

**Disponibilité :** Flash Player 5 ; ActionScript 1.0

### Exemple

Cet exemple indique comment le typage automatique de données convertit `false` en nombre et en chaîne :

```
var bool1:Boolean = Boolean(false);
```

```
// converts it to the number 0
trace(1 + bool1); // outputs 1

// converts it to a string
trace("String: " + bool1); // outputs String: false
```

## Constante Infinity

Spécifie la valeur IEEE-754 représentant l'infini positif. La valeur de cette constante est identique à `Number.POSITIVE_INFINITY`.

**Disponibilité :** Flash Player 5 ; ActionScript 1.0

### Voir également

`POSITIVE_INFINITY (propriété Number.POSITIVE_INFINITY)`

## Constante -Infinity

Spécifie la valeur IEEE-754 représentant l'infini négatif. La valeur de cette constante est identique à `Number.NEGATIVE_INFINITY`.

**Disponibilité :** Flash Player 5 ; ActionScript 1.0

### Voir également

`NEGATIVE_INFINITY (propriété Number.NEGATIVE_INFINITY)`

## Constante NaN

Variable prédéfinie incluant la valeur IEEE-754 pour NaN (n'est pas un nombre). Pour déterminer si la valeur d'un nombre est NaN, utilisez `isNaN()`.

**Disponibilité :** Flash Player 5 ; ActionScript 1.0

### Voir également

`Fonction isNaN`, `NaN (propriété Number.NaN)`

## Constante newline

Insère un caractère de retour chariot (`\r`) qui insère une ligne vierge dans le texte généré par votre code. Utilisez `newline` pour ménager un espace pour les informations extraites par une fonction ou une instruction dans votre code.

**Disponibilité :** Flash Player 4 ; ActionScript 1.0

### Exemple

L'exemple suivant indique comment `newline` affiche la sortie à partir de l'instruction `trace()` sur plusieurs lignes.

```
var myName:String = "Lisa", myAge:Number = 30;
trace(myName+myAge);
trace("-----");
trace(myName+newline+myAge);
// output:
Lisa30
-----
Lisa
30
```

### Voir également

`Fonction trace`

## Constante null

Valeur spéciale qui peut être affectée à des variables ou renvoyée par une fonction en l'absence de données. Vous pouvez utiliser `null` pour représenter les valeurs manquantes ou dont le type de données n'est pas défini.

**Disponibilité :** Flash Player 5 ; ActionScript 1.0

### Exemple

L'exemple suivant vérifie les six premières valeurs d'un tableau indexé et renvoie un message si aucune valeur n'est définie (si la valeur == `null`) :

```
var testArray:Array = new Array();
testArray[0] = "fee";
testArray[1] = "fi";
testArray[4] = "foo";

for (i = 0; i < 6; i++) {
 if (testArray[i] == null) {
 trace("testArray[" + i + "] == null");
 }
}
```

Le code suivant est renvoyé :

```
testArray[2] == null
testArray[3] == null
```

```
testArray[5] == null
```

# Constante true

Valeur booléenne unique qui représente l'opposé de `false`. Lorsque le typage automatique de données convertit `true` en nombre, il renvoie 1. Lorsqu'il convertit `true` en chaîne, il renvoie `"true"`.

**Disponibilité :** Flash Player 5 ; ActionScript 1.0

## Exemple

L'exemple suivant illustre l'utilisation de `true` dans une instruction `if` :

```
var shouldExecute:Boolean;
// ...
// code that sets shouldExecute to either true or false goes here
// shouldExecute is set to true for this example:

shouldExecute = true;

if (shouldExecute == true) {
 trace("your statements here");
}

// true is also implied, so the if statement could also be written:
// if (shouldExecute) {
// trace("your statements here");
// }
```

L'exemple suivant indique comment le typage automatique de données convertit `true` en nombre 1 :

```
var myNum:Number;
myNum = 1 + true;
trace(myNum); // output: 2
```

## Voir également

`Constante false`, `Boolean`

# Constante undefined

Une valeur spéciale, qui indique généralement qu'une variable n'a pas encore reçu de valeur. Une référence à une valeur non définie renvoie la valeur spéciale `undefined`. Le code ActionScript `typeof(undefined)` renvoie la chaîne `"undefined"`. L'unique valeur du type undefined `undefined` est `undefined`.

Dans les fichiers publiés pour Flash Player 6 ou version précédente, la valeur de `String(undefined)` est " " (une chaîne vide). Dans les fichiers publiés pour Flash Player 7 ou version ultérieure, la valeur de `String(undefined)` est `"undefined"` (`undefined` est converti en chaîne).

Dans les fichiers publiés pour Flash Player 6 ou version précédente, la valeur de `Number(undefined)` est `0`. Dans les fichiers publiés pour Flash Player 7 ou version ultérieure, la valeur de `Number(undefined)` est `NaN`.

La valeur `undefined` est similaire à la valeur spéciale `null`. Lorsque les propriétés `null` et `undefined` sont comparées avec l'opérateur d'égalité (==), elles sont considérées comme égales. Lorsque les propriétés `null` et `undefined` sont comparées avec l'opérateur d'égalité stricte (===), elles sont considérées comme différentes.

**Disponibilité :** Flash Player 5 ; ActionScript 1.0

## Exemple

Dans l'exemple suivant, la variable `x` n'a pas été déclarée, sa valeur est donc undefined `undefined`.

Dans la première section du code, l'opérateur d'égalité (==) compare la valeur de `x` à la valeur `undefined` ; le résultat approprié est ensuite envoyé au panneau de sortie.

Dans la deuxième section du code, l'opérateur d'égalité (==) compare les valeurs `null` et `undefined`.

```
// x has not been declared
trace("The value of x is "+x);

if (x == undefined) {
 trace("x is undefined");
} else {
 trace("x is not undefined");
}

trace("typeof (x) is "+typeof (x));

if (null == undefined) {
 trace("null and undefined are equal");
} else {
 trace("null and undefined are not equal");
}
```

Le résultat suivant s'affiche dans le panneau de sortie.

```
The value of x is undefined
x is undefined
typeof (x) is undefined
null and undefined are equal
```

# Fonctions globales

Cette section regroupe des fonctions intégrées qui sont disponibles dans tout fichier SWF ayant recours à ActionScript. Ces fonctions globales couvrent un vaste ensemble de tâches communes de programmation, telles que l'application des types de données (`Boolean()`, `int()`, etc.), la production d'informations de débogage (`trace()`) et la communication avec Flash Player ou le navigateur (`fscommand()`).

## Résumé des fonctions globales

| Modificateurs | Signature | Description |
|---|---|---|
| | `Array([numElements:Number], [elementN:Object])` | Crée un nouveau tableau vide ou convertit les éléments spécifiés en tableau. |
| | `asfunction(function:String, parameter:String)` | Un protocole spécial dédié aux URL des champs de texte HTML permettant à un lien HREF d'appeler une fonction ActionScript. |
| | `Boolean(expression:Object)` | Convertit le paramètre *`expression`* en une valeur booléenne et renvoie `true` ou `false`. |
| | `call(frame:Object)` | *Déconseillée* à partir de Flash Player 5. Il est recommandé d'utiliser l'instruction `function`. Exécute le script dans l'image appelée sans positionner la tête de lecture sur celle-ci. |
| | `chr(number:Number)` | *Déconseillée* à partir de Flash Player 5. Il est recommandé d'utiliser la méthode `String.fromCharCode()`. Convertit les numéros de code ASCII en caractères. |
| | `clearInterval(intervalID:Number)` | Arrête l'appel `setInterval()`. |
| | `duplicateMovieClip(target:Object, newname:String, depth:Number)` | Crée une occurrence de clip pendant la lecture du fichier SWF. |
| | `escape(expression:String)` | Convertit le paramètre en chaîne et applique le format de code URL, où tous les caractères qui ne sont pas de type alphanumérique sont remplacés par des séquences hexadécimales %. |
| | `eval(expression:Object)` | Accède aux variables, propriétés, objets ou clips en fonction de leur nom. |

| Modificateurs | Signature | Description |
|---|---|---|
| | `fscommand(command:String, parameters:String)` | Permet au fichier SWF de communiquer avec Flash Player ou le programme hébergeant Flash Player, tel qu'un navigateur Web. |
| | `getProperty(my_mc:String, property)` | Renvoie la valeur de la propriété spécifiée pour le clip *my_mc*. |
| | `getTimer()` | Renvoie le nombre de millisecondes qui se sont écoulées depuis le début de la lecture du fichier SWF. |
| | `getURL(url:String, [window:String], [method:String])` | Charge un document en provenance d'une URL spécifique dans une fenêtre ou transmet des variables à une autre application à une URL donnée. |
| | `getVersion()` | Renvoie une chaîne contenant la version de Flash Player et des informations sur la plate-forme. |
| | `gotoAndPlay([scene:String], frame:Object)` | Place la tête de lecture sur l'image spécifiée dans une séquence et commence la lecture à partir de cette image. |
| | `gotoAndStop([scene:String], frame:Object)` | Place la tête de lecture sur l'image spécifiée sur une séquence et l'arrête à ce niveau. |
| | `ifFrameLoaded([scene:String], frame:Object)` | *Déconseillée* à partir de Flash Player 5. Cette fonction est déconseillée. Macromedia vous recommande d'utiliser la propriété `MovieClip._framesloaded`. Vérifie si le contenu d'une image spécifique est disponible localement. |
| | `int(value:Number)` | *Déconseillée* à partir de Flash Player 5. Il est recommandé d'utiliser la méthode `Math.round()`. Convertit un nombre décimal en valeur entière en tronquant la valeur décimale. |
| | `isFinite(expression:Object)` | Evalue l'*expression* et renvoie `true` s'il s'agit d'un nombre fini ou `false` s'il s'agit de l'infini ou de l'infini négatif. |
| | `isNaN(expression:Object)` | Evalue le paramètre et renvoie `true` si la valeur est `NaN` (not a number - n'est pas un nombre). |
| | `length(expression:String, variable:Object)` | *Déconseillée* à partir de Flash Player 5. Cette fonction, à l'instar de toutes les fonctions de chaîne, est déconseillée. Macromedia vous recommande d'utiliser les méthodes de la classe String et la propriété `String.length` pour effectuer les mêmes opérations. Renvoie la longueur de la chaîne ou variable spécifiée. |

| Modificateurs | Signature | Description |
|---|---|---|
| | `loadMovie(url:String, target:Object, [method:String])` | Charge un fichier SWF ou JPEG dans Flash Player pendant la lecture du fichier SWF d'origine. |
| | `loadMovieNum(url:String, level:Number, [method:String])` | Charge un fichier SWF ou JPEG dans l'un des niveaux de Flash Player pendant la lecture du fichier SWF. |
| | `loadVariables(url:String, target:Object, [method:String])` | Lit les données dans un fichier externe, tel qu'un fichier texte ou du texte généré par ColdFusion, un script CGI, des pages ASP (Active Server Pages), PHP ou un script Perl et définit les valeurs pour les variables dans un clip cible. |
| | `loadVariablesNum(url:String, level:Number, [method:String])` | Lit les données dans un fichier externe, tel qu'un fichier texte ou du texte généré par ColdFusion, un script CGI, des pages ASP (Active Server Pages), PHP ou un script Perl et définit les valeurs pour les variables dans un niveau de Flash Player. |
| | `mbchr(number:Number)` | *Déconseillée* à partir de Flash Player 5. Il est recommandé d'utiliser la méthode `String.fromCharCode()`.<br>Convertit un numéro de code ASCII en caractère multi-octets. |
| | `mblength(string:String)` | *Déconseillée* à partir de Flash Player 5. Il est recommandé d'utiliser les méthodes et les propriétés de la classe String.<br>Renvoie la longueur de la chaîne de caractères multi-octets. |
| | `mbord(character:String)` | *Déconseillée* à partir de Flash Player 5. Il est recommandé d'utiliser la méthode `String.charCodeAt()`.<br>Convertit le caractère spécifié en nombre multi-octets. |
| | `mbsubstring(value:String, index:Number, count:Number)` | *Déconseillée* à partir de Flash Player 5. Il est recommandé d'utiliser la méthode `String.substr()`.<br>Extrait une nouvelle chaîne de caractères multi-octets d'une chaîne de caractères multi-octets. |
| | `MMExecute(command:String)` | Permet d'émettre des commandes de l'API Flash JavaScript (JSAPI) à partir d'ActionScript. |
| | `nextFrame()` | Place la tête de lecture sur l'image suivante. |
| | `nextScene()` | Place la tête de lecture sur l'image 1 de la séquence suivante. |

| Modificateurs | Signature | Description |
|---|---|---|
| | `Number(expression:Object)` | Convertit le paramètre *`expression`* en valeur numérique. |
| | `Object([value:Object])` | Crée un objet vide ou convertit le nombre, la chaîne ou la valeur booléenne spécifié en objet. |
| | `on(mouseEvent:Object)` | Spécifie l'événement de type souris ou pression de touche devant déclencher une action. |
| | `onClipEvent(movieEvent:Object)` | Déclenche les actions définies pour une instance spécifique de clip. |
| | `ord(character:String)` | *Déconseillée* à partir de Flash Player 5. Il est recommandé d'utiliser les méthodes et les propriétés de la classe String.<br>Convertit les caractères en numéros de code ASCII. |
| | `parseFloat(string:String)` | Convertit une chaîne en nombre à virgule flottante. |
| | `parseInt(expression:String, [radix:Number])` | Convertit une chaîne en entier. |
| | `play()` | Fait avancer la tête de lecture au sein du scénario. |
| | `prevFrame()` | Place la tête de lecture sur l'image précédente. |
| | `prevScene()` | Place la tête de lecture sur l'image 1 de la séquence précédente. |
| | `print(target:Object, boundingBox:String)` | Imprime le clip `target` en fonction des limites spécifiées par le paramètre (`bmovie`, `bmax` ou `bframe`). |
| | `printAsBitmap(target:Object, boundingBox:String)` | Imprime le clip `target` en tant que bitmap en fonction des limites spécifiées par le paramètre (`bmovie`, `bmax` ou `bframe`). |
| | `printAsBitmapNum(level:Number, boundingBox:String)` | Imprime un niveau dans Flash Player en tant que bitmap en fonction des limites spécifiées par le paramètre (`bmovie`, `bmax` ou `bframe`). |
| | `printNum(level:Number, boundingBox:String)` | Imprime le niveau dans Flash Player en fonction des limites spécifiées par le paramètre `boundingBox` (`bmovie`, `bmax`, `bframe`). |
| | `random(value:Number)` | *Déconseillée* à partir de Flash Player 5. Il est recommandé d'utiliser la méthode `Math.random()`.<br>Renvoie un entier aléatoire compris entre 0 et un inférieur au nombre entier spécifié dans le paramètre ‹code›‹em›value‹/em›‹/code› . |

| Modificateurs | Signature | Description |
|---|---|---|
| | `removeMovieClip(target:Object)` | Supprime le clip spécifié. |
| | `setInterval(functionReference:Function, interval:Number, [param:Object], objectReference:Object, methodName:String)` | Appelle une fonction ou une méthode d'un objet à des intervalles périodiques pendant la lecture d'un fichier SWF. |
| | `setProperty(target:Object, property:Object, expression:Object)` | Modifie la valeur des propriétés d'un clip pendant la lecture de ce dernier. |
| | `showRedrawRegions(enable:Boolean, [color:Number])` | Permet au débogueur de délimiter les zones redessinées de l'écran. |
| | `startDrag(target:Object, [lock:Boolean], [left,top,right,bottom:Number])` | Rend le clip *`target`* déplaçable pendant la lecture de l'animation. |
| | `stop()` | Arrête le fichier SWF en cours de lecture. |
| | `stopAllSounds()` | Arrête tous les sons en cours de diffusion à partir d'un fichier SWF, sans arrêter la tête de lecture. |
| | `stopDrag()` | Arrête l'opération de déplacement en cours. |
| | `String(expression:Object)` | Renvoie une chaîne représentant le paramètre spécifié. |
| | `substring(string:String, index:Number, count:Number)` | *Déconseillée* à partir de Flash Player 5. Il est recommandé d'utiliser la méthode `String.substr()`. Extrait une partie d'une chaîne. |
| | `targetPath(targetObject:Object)` | Renvoie une chaîne contenant le chemin cible de *`movieClipObject`*. |
| | `tellTarget(target:String, statement(s))` | *Déconseillée* à partir de Flash Player 5. Macromedia vous recommande d'utiliser une notation de type point (.) et l'instruction `with`.<br>Applique les instructions spécifiées dans le paramètre *`statements`* au scénario spécifié dans le paramètre *`target`*. |

| Modificateurs | Signature | Description |
|---|---|---|
| | `toggleHighQuality()` | *Déconseillée* à partir de Flash Player 5. Il est recommandé d'utiliser la méthode `_quality`. Active et désactive l'anticrénelage dans Flash Player. |
| | `trace(expression:Object)` | Evalue l'expression et renvoie le résultat. |
| | `unescape(string:String)` | Evalue le paramètre *x* en tant que chaîne, décode la chaîne qui est au format codé en URL (en convertissant toutes les séquences hexadécimales en caractères ASCII) et renvoie la chaîne. |
| | `unloadMovie(target:Object)` | Supprime le clip qui a été chargé par l'intermédiaire de la fonction `loadMovie()` de Flash Player. |
| | `unloadMovieNum(level:Number)` | Supprime un fichier SWF ou une image qui a été chargé par l'intermédiaire de la fonction `loadMovieNum()` de Flash Player. |
| | `updateAfterEvent()` | Met à jour l'affichage lorsque vous l'appelez à partir d'un gestionnaire ou via `setInterval()`. |

# Fonction Array

```
Array() : Array
Array(numElements:Number) : Array
Array(element0:Object, [element1, element2, ...elementN]) : Array
```

Crée un nouveau tableau de longueur 0 ou supérieure, ou un tableau contenant la liste des éléments spécifiés, probablement de types de données différents.

Utilisez la méthode `Array()` pour créer l'un des tableaux suivants :

- Un tableau vide
- Un tableau d'une longueur spécifique mais dont les éléments ont des valeurs non définies
- Un tableau dont les éléments ont des valeurs spécifiques

L'utilisation de cette fonction revient à créer un tableau avec le constructeur Array (voir « Constructeur pour la classe Array »).

Vous pouvez transmettre un nombre (`numElements`) ou la liste des éléments contenant un ou plusieurs types différents (`element0, element1, ... elementN`).

Les paramètres qui peuvent accepter plusieurs types de données sont répertoriés dans la signature sous le type `Object`.

**Disponibilité :** Flash Player 6 ; ActionScript 1.0

### Paramètres

*numElements*:Number [facultatif] - Un entier positif spécifiant le nombre d'éléments contenus dans le tableau. Vous pouvez spécifier numElements ou la liste des éléments, mais pas les deux.

*elementN*:Object [facultatif] - Un ou plusieurs paramètres, element0, element1, ... , elementN, dont les valeurs peuvent être de n'importe quel type. Les paramètres qui peuvent accepter plusieurs types de données sont répertoriés sous le type Object. Vous pouvez spécifier numElements ou la liste des éléments, mais pas les deux.

### Renvoie

Array - Un tableau.

### Exemple

```
var myArray:Array = Array();
myArray.push(12);
trace(myArray); //traces 12
myArray[4] = 7;
trace(myArray); //traces 12,undefined,undefined,undefined,7
```

Utilisation 2 : L'exemple suivant crée un tableau de longueur 4 qui n'inclut aucun élément défini :

```
var myArray:Array = Array(4);
trace(myArray.length); // traces 4
trace(myArray); // traces undefined,undefined,undefined,undefined
```

Utilisation 3 : L'exemple suivant crée un tableau incluant trois éléments définis :

```
var myArray:Array = Array("firstElement", "secondElement", "thirdElement");
trace (myArray); // traces firstElement,secondElement,thirdElement
Unlike the Array class constructor, the Array() function does not use the
  keyword new .
```

### Voir également

Array

## Protocole asfunction

```
asfunction:function:Function, parameter:String
```

Un protocole spécial dédié aux URL des champs de texte HTML permettant à un lien HREF d'appeler une fonction ActionScript. Dans les champs de texte HTML, vous pouvez créer des liens à l'aide de la balise `A` HTML. L'attribut HREF de la balise `A` contient une URL qui utilise un protocole standard tel que HTTP, HTTPS ou FTP. Le protocole `asfunction` est un protocole supplémentaire spécifique à Flash, ce qui engendre l'appel d'une fonction ActionScript à partir du lien.

**Disponibilité :** Flash Player 5 ; ActionScript 1.0

## Paramètres

*`function`*`:String` - Un identificateur pour une fonction.

*`parameter`*`:String` - Une chaîne transmise à la fonction nommée dans le paramètre *`function`*.

## Exemple

Dans l'exemple suivant, la fonction `playMP3()` est définie. L'objet TextField list_txt est créé et défini de manière à pouvoir restituer le texte HTML. Le texte Track 1 et Track 2 constitue des liens dans le champ de texte. La fonction `playMP3()` est appelée lorsque l'utilisateur clique sur le lien et lit le fichier MP3 transmis en tant que paramètre de l'appel asfunction.

```
var myMP3:Sound = new Sound();
function playMP3(mp3:String) {
 myMP3.loadSound(mp3, true);
 myMP3.onLoad = function(success) {
 if (!success) {
 // code to handle errors here
 }
 };
}
this.createTextField("list_txt", this.getNextHighestDepth(), 0, 0, 200,
  100);
list_txt.autoSize = true;
list_txt.html = true;
list_txt.multiline = true;
list_txt.htmlText = "<a href=\"asfunction:playMP3, track1.mp3\">Track 1</
  a><br>";
list_txt.htmlText += "<a href=\"asfunction:playMP3, track2.mp3\">Track 2</
  a><br>";
```

Lorsque vous cliquez sur un lien, le fichier son MP3 est lu dans Flash Player.

## Voir également

`htmlText (propriété TextField.htmlText)`

# Fonction Boolean

`Boolean(expression:Object) : Boolean`

Convertit le paramètre `expression` en valeur booléenne et renvoie une valeur comme indiqué dans la liste suivante :

- Si `expression` est une valeur booléenne, la valeur renvoyée est `expression`.
- Si `expression` est un nombre, la valeur renvoyée est `true` si le nombre diffère de 0 ; sinon, la valeur renvoyée est `false`.

Si `expression` est une chaîne, renvoie l'une des valeurs suivantes :

- Dans les fichiers publiés pour Flash Player 6 ou version antérieure, la chaîne est tout d'abord convertie en nombre. Sa valeur est `true` pour les nombres différents de zéro, `false` dans les autres cas.
- Dans les fichiers publiés pour Flash Player 7 ou version ultérieure, le résultat est `true` si la longueur de la chaîne est supérieure à zéro et `false` en cas de chaîne vide.

Si `expression` est une chaîne, le résultat est `true` si la longueur de cette chaîne est supérieure à zéro, `false` en cas de chaîne vide.

- Si l'`expression` est `undefined` ou `NaN` (n'est pas un nombre), la valeur renvoyée est `false`.
- Si `expression` est un clip ou un objet, la valeur renvoyée est `true`.

Contrairement au constructeur de classe Boolean, la fonction `Boolean()` n'utilise pas le mot-clé new. De plus, le constructeur de classe Boolean initialise un objet booléen sur `false` si aucun paramètre n'est spécifié, bien que la fonction `Boolean()` renvoie `undefined` en l'absence de paramètres.

**Disponibilité :** Flash Player 5 ; ActionScript 1.0

## Paramètres

*`expression`*`:Object` - Une expression à convertir en valeur booléenne.

## Renvoie

`Boolean` - Une valeur booléenne.

## Exemple

```
trace(Boolean(-1)); // output: true
trace(Boolean(0)); // output: false
trace(Boolean(1)); // output: true


trace(Boolean(true)); // output: true
```

```
trace(Boolean(false)); // output: false


trace(Boolean("true")); // output: true
trace(Boolean("false")); // output: true

trace(Boolean("Craiggers")); // output: true
trace(Boolean("")); // output: false
```

Si les fichiers sont publiés pour Flash Player 6 ou version antérieure, les résultats diffèrent pour trois des exemples précédents :

```
trace(Boolean("true")); // output: false
trace(Boolean("false")); // output: false
trace(Boolean("Craiggers")); // output: false
```

Cet exemple illustre une différence significative entre l'utilisation de la fonction `Boolean()` et celle de la classe Boolean. La fonction `Boolean()` permet de créer une valeur Boolean, tandis que la classe Boolean crée un objet Boolean. Les valeurs booléennes sont comparées en fonction de leur valeur, tandis que les objets booléens sont comparés par référence.

```
// Variables representing Boolean values are compared by value
var a:Boolean = Boolean("a"); // a is true
var b:Boolean = Boolean(1); // b is true
trace(a==b); // true

// Variables representing Boolean objects are compared by reference
var a:Boolean = new Boolean("a"); // a is true
var b:Boolean = new Boolean(1); // b is true
trace(a == b); // false
```

### Voir également

`Boolean`

# Fonction call

```
call(frame)
```

*Déconseillée* à partir de Flash Player 5. Il est recommandé d'utiliser l'instruction `function`.

Exécute le script dans l'image appelée sans positionner la tête de lecture sur celle-ci. Les variables locales n'existent pas après l'exécution du script.

- Si les variables ne sont pas déclarées dans un bloc (`{}`) mais que la liste d'action a été exécutée à l'aide d'une action call(), les variables sont locales et expirent à la fin de la liste actuelle.

- Si les variables ne sont pas déclarées dans un bloc et que la liste d'action actuelle n'a pas été exécutée à l'aide de l'action call(), les variables sont interprétées en tant que variables de scénario.

**Disponibilité :** Flash Player 4 ; ActionScript 1.0

### Paramètres

*frame*:Object - L'étiquette ou le numéro d'une image dans le scénario.

### Voir également

Instruction function, call (méthode Function.call)

## Fonction chr

```
chr(number:Number) : String
```

*Déconseillée* à partir de Flash Player 5. Il est recommandé d'utiliser la méthode String.fromCharCode().

Convertit les numéros de code ASCII en caractères.

**Disponibilité :** Flash Player 4 ; ActionScript 1.0

### Paramètres

*number*:Number - Un numéro de code ASCII.

### Renvoie

String - La valeur de caractère du code ASCII spécifié.

### Exemple

L'exemple suivant convertit le nombre 65 en lettre A et l'affecte à la variable myVar : `myVar = chr(65);`

### Voir également

fromCharCode (méthode String.fromCharCode)

## Fonction clearInterval

```
clearInterval(intervalID:Number) : Void
```

Arrête l'appel setInterval().

**Disponibilité :** Flash Player 6 ; ActionScript 1.0

### Paramètres

*intervalID*:Number - Un identificateur numérique (entier) renvoyé par un appel à setInterval().

### Exemple

L'exemple suivant définit, puis supprime un appel interval :

```
function callback() {
 trace("interval called: "+getTimer()+" ms.");
}

var intervalID:Number = setInterval(callback, 1000);
```

Vous devez supprimer l'intervalle lorsque vous n'avez plus besoin d'utiliser la fonction. Créez un bouton intitulé clearInt_btn et utilisez le code ActionScript suivant pour supprimer setInterval() :

```
clearInt_btn.onRelease = function(){
 clearInterval( intervalID );
 trace("cleared interval");
};
```

### Voir également

Fonction setInterval

## Fonction duplicateMovieClip

```
duplicateMovieClip(target:String, newname:String, depth:Number) : Void
duplicateMovieClip(target:MovieClip, newname:String, depth:Number) : Void
```

Crée une occurrence de clip pendant la lecture du fichier SWF. La tête de lecture des clips dupliqués commence toujours à l'image 1, quelle que soit la position de la tête de lecture dans le clip d'origine. Les variables du clip d'origine ne sont pas copiées dans le clip dupliqué. Utilisez la fonction ou la méthode removeMovieClip() pour supprimer une occurrence de clip créée avec duplicateMovieClip().

**Disponibilité :** Flash Player 4 ; ActionScript 1.0

### Paramètres

*target*:Object - Le chemin cible du clip à dupliquer. Ce paramètre peut être de type String (tel que « my_mc ») ou une référence directe à l'occurrence de clip (par exemple my_mc). Les paramètres qui peuvent accepter plusieurs types de données sont répertoriés sous le type Object.

*newname*:String - Un identificateur unique pour le clip dupliqué.

*depth*:Number - Un niveau de profondeur unique pour le clip dupliqué. Le niveau de profondeur correspond à l'ordre d'empilement des clips dupliqués. Cet ordre d'empilement correspond à l'ordre d'empilement des calques dans le scénario ; les clips dont le niveau de profondeur est inférieur sont masqués par les clips de niveau supérieur. Vous devez associer un niveau de profondeur à chaque clip pour ne pas remplacer les fichiers SWF figurant à des profondeurs non utilisées.

### Exemple

Dans l'exemple suivant, une nouvelle occurrence de clip intitulée `img_mc.` est créée. Une image est chargée dans le clip, puis le clip `img_mc` est dupliqué. Le clip dupliqué est intitulé `newImg_mc` ; ce nouveau clip est déplacé sur la scène afin de ne pas recouvrir le clip d'origine et la même image est chargée dans le deuxième clip.

```
this.createEmptyMovieClip("img_mc", this.getNextHighestDepth());
img_mc.loadMovie("http://www.helpexamples.com/flash/images/image1.jpg");
duplicateMovieClip(img_mc, "newImg_mc", this.getNextHighestDepth());
newImg_mc._x = 200;
newImg_mc.loadMovie("http://www.helpexamples.com/flash/images/image1.jpg");
```

Pour supprimer le clip dupliqué, vous pouvez ajouter ce code pour un bouton intitulé `myButton_btn`.

```
this.myButton_btn.onRelease = function(){
 removeMovieClip(newImg_mc);
};
```

### Voir également

`Fonction removeMovieClip`, `duplicateMovieClip (méthode MovieClip.duplicateMovieClip)`, removeMovieClip (méthode MovieClip.removeMovieClip)

## Fonction escape

```
escape(expression:String) : String
```

Convertit le paramètre en chaîne et applique le format de code URL, où tous les caractères qui ne sont pas de type alphanumérique sont remplacés par des séquences hexadécimales %. Lorsque cette fonction est utilisée dans une chaîne codée au format URL, le symbole pour-cent (%) introduit les caractères d'échappement et ne doit pas être confondu avec l'opérateur modulo (%).

**Disponibilité :** Flash Player 5 ; ActionScript 1.0

### Paramètres

*expression*:String - L'expression est convertie en chaîne et est codée au format URL.

### Renvoie

String - Chaîne codée au format URL.

### Exemple

Le code suivant donne le résultat someuser%40somedomain%2Ecom :

```
var email:String = "someuser@somedomain.com";
trace(escape(email));
```

Dans cet exemple, le symbole (@) a été remplacé par %40 et le point (.) par %2E. Cela est particulièrement utile lorsque vous essayez de transmettre des informations à un serveur distant et si les données contiennent des caractères spéciaux (par exemple, & ou ?), comme indiqué dans le code suivant :

```
var redirectUrl:String = "http://
  www.somedomain.com?loggedin=true&username=Gus";
getURL("http://www.myothersite.com?returnurl="+ escape(redirectUrl));
```

### Voir également

Fonction unescape

## Fonction eval

```
eval(expression:Object) : Object
eval(expression:String) : Object
```

Accède aux variables, propriétés, objets ou clips en fonction de leur nom. Lorsque l'expression est une variable ou une propriété, la fonction renvoie la valeur de cette variable ou de cette propriété. Si l'expression est un objet ou un clip, la fonction renvoie une référence de l'objet ou du clip. Si l'élément nommé dans l'expression est introuvable, la fonction renvoie undefined.

Sous Flash 4, eval() permettait de simuler des tableaux ; à partir de Flash 5, vous devez utiliser la classe Array pour ce faire.

Sous Flash 4, vous pouvez également utiliser eval() pour définir de façon dynamique la valeur d'une variable ou d'un nom d'occurrence et l'extraire. Cette opération est également possible avec l'opérateur de tableau ([]).

A partir de Flash 5, vous ne pouvez plus recourir à eval() pour définir de façon dynamique et extraire la valeur d'une variable ou d'un nom d'occurrence, car vous ne pouvez pas utiliser eval() dans la partie gauche d'une équation. Par exemple, remplacez le code

```
eval ("var" + i) = "first";
```

par :

```
this["var"+i] = "first"
```

ou par :

```
set ("var" + i, "first");
```

**Disponibilité :** Flash Player 5 ; ActionScript 1.0

### Paramètres

*expression*:Object - Le nom d'une variable, d'une propriété, d'un objet ou d'un clip à extraire. Ce paramètre peut être de type String ou une référence directe à l'occurrence d'objet (les guillemets (" ") sont facultatifs.)

### Renvoie

Object - Une valeur, une référence à un objet ou un clip, ou undefined.

### Exemple

L'exemple suivant utilise eval() pour définir les propriétés des clips nommés de façon dynamique. Ce code ActionScript définit la propriété _rotation de trois clips intitulés square1_mc, square2_mc et square3_mc.

```
for (var i = 1; i <= 3; i++) {
 setProperty(eval("square"+i+"_mc"), _rotation, 5);
}
```

Vous pouvez également utiliser le code ActionScript suivant :

```
for (var i = 1; i <= 3; i++) {
 this["square"+i+"_mc"]._rotation = -5;
}
```

### Voir également

Array, Instruction set variable

## Fonction fscommand

```
fscommand(command:String, parameters:String) : Void
```

Permet au fichier SWF de communiquer avec Flash Player ou le programme hébergeant Flash Player, tel qu'un navigateur Web. La fonction fscommand() permet également de transmettre des messages à Macromedia Director ou à Visual Basic (VB), Visual C++ et autres programmes ayant recours aux contrôles ActiveX.

La fonction `fscommand()` permet à un fichier SWF de communiquer avec un script dans une page Web. Cependant, l'accès au script est contrôlé par le paramètre `allowScriptAccess` de la page Web. (Vous pouvez définir cet attribut dans le code HTML intégré au fichier SWF, par exemple dans la balise `PARAM` pour Internet Explorer ou dans la balise `EMBED` pour Netscape. Lorsque l'attribut `allowScriptAccess` est défini sur `"never"`, un fichier SWF ne peut pas accéder aux scripts de la page Web. A partir de Flash Player 7, lorsque l'attribut `allowScriptAccess` est défini sur `"always"`, un fichier SWF peut toujours accéder aux scripts de la page Web. Lorsque l'attribut `allowScriptAccess` est défini sur `"sameDomain"`, les scripts sont uniquement autorisés à partir des fichiers SWF qui se trouvent dans le même domaine que la page Web ; les scripts sont toujours autorisés sur les versions précédentes de Flash Player. Si l'attribut `allowScriptAccess` n'est pas spécifié dans une page HTML, il est défini par défaut sur `"sameDomain"` pour les fichiers SWF, version 8 et ultérieure ; et sur `"always"` pour les fichiers SWF, version 7 et antérieure.

Utilisation 1 : Pour utiliser `fscommand()` afin d'envoyer un message à Flash Player, vous devez utiliser les commandes et les paramètres prédéfinis. Le tableau suivant indique les valeurs que vous pouvez spécifier pour les paramètres `fscommand()` `command` et `parameters` de la fonction. Ces valeurs contrôlent les fichiers SWF lus par Flash Player, y compris les projections. (Une *projection* est un fichier SWF enregistré sous un format permettant de l'exécuter en tant qu'application autonome, pouvant être lue sans Flash Player.)

| Commande | Paramètre | Rôle |
|---|---|---|
| `quit` | Aucun | Ferme la projection. |
| `fullscreen` | `true` ou `false` | Spécifiez `true` pour exécuter Flash Player en mode plein écran. Spécifiez `false` pour rétablir le mode d'affichage normal du menu. |
| `allowscale` | `true` ou `false` | Si vous spécifiez `false`, le lecteur affiche toujours la taille d'origine du fichier SWF, sans le redimensionner. Si vous spécifiez `true`, le fichier SWF adopte l'échelle 100 % du lecteur. |

| Commande | Paramètre | Rôle |
|---|---|---|
| `showmenu` | `true` ou `false` | Spécifiez `true` pour activer le jeu complet d'éléments de menu contextuel. Spécifiez `false` pour masquer tous les éléments de menu contextuel, à l'exception de A propos de Flash Player et Paramètres. |
| `exec` | Chemin de l'application | Exécute une application depuis la projection. |
| `trapallkeys` | `true` ou `false` | Spécifiez `true` pour envoyer tous les événements de touche, y compris les touches de raccourci, au gestionnaire `onClipEvent(keyDown/keyUp)` de Flash Player. |

**Disponibilité :**

- Les commandes décrites dans le tableau ne sont pas disponibles pour les lecteurs Web.
- Toutes les commandes sont disponibles dans les applications autonomes, telles que les projections.
- Seuls `allowscale` et `exec` sont disponibles sur les lecteurs de tests d'animation.

La commande `exec` ne peut contenir que les caractères compris entre A et Z, a et z, 0 et 9, les caractères point (.) et souligné (_). La commande `exec` ne s'exécute que dans le sous-répertoire fscommand. En d'autres termes, si vous utilisez la commande `exec` pour appeler une application, cette application doit résider dans un sous-répertoire appelé fscommand. La commande `exec` ne fonctionne qu'à partir d'un fichier de projection Flash.

Utilisation 2 : Pour utiliser `fscommand()` pour envoyer un message à un langage de programmation tel que JavaScript dans un navigateur Web, vous pouvez transmettre deux arguments quelconques avec les paramètres `command` et `parameters`. Ces paramètres peuvent être des chaînes ou des expressions. Ils sont utilisés dans une fonction JavaScript qui gère ou *traite* la fonction `fscommand()`.

Dans un navigateur Web, la fonction `fscommand()` appelle la fonction JavaScript `moviename_DoFScommand`, résidant dans la page Web qui contient le fichier SWF. Pour `moviename`, attribuez le nom de l'objet Flash utilisé pour l'attribut `NAME` de la balise `EMBED` ou la propriété ID de la balise `OBJECT`. Si vous attribuez le nom `myMovie` au fichier SWF, la fonction JavaScript `myMovie_DoFScommand` est appelée.

Dans la page Web qui contient le fichier SWF, définissez l'attribut `allowScriptAccess` de manière à autoriser, ou non, le fichier SWF à accéder à la page Web. (Vous pouvez définir cet attribut dans le code HTML intégré au fichier SWF, par exemple dans la balise `PARAM` pour Internet Explorer ou dans la balise `EMBED` pour Netscape.) Lorsque l'attribut `allowScriptAccess` est défini sur `"never"`, les scripts externes échouent systématiquement. Lorsque l'attribut `allowScriptAccess` est défini sur `"always"`, les scripts externes sont acceptés systématiquement. Lorsqu'il est défini sur `"sameDomain"`, les scripts sont uniquement autorisés à partir des fichiers SWF qui se trouvent dans le même domaine que la page Web. Si l'attribut `allowScriptAccess` n'est pas spécifié dans une page Web, il est défini par défaut sur `"sameDomain"` pour Flash Player 8, et sur `"always"` pour les versions précédentes de Flash Player.

Lorsque vous utilisez cette fonction, référez-vous au modèle de sécurité de Flash Player. Pour Flash Player 8, la fonction `fscommand()` n'est pas autorisée si le fichier SWF appelant se trouve dans le système de fichiers local ou dans le Sandbox de réseau local et si la page HTML contenant ce fichier se trouve dans un Sandbox non approuvé. Pour plus d'informations, consultez le :

- Chapitre 17, « Fonctionnement de la sécurité » du guide *Formation à ActionScript 2.0 dans Flash*
- Livre blanc concernant la sécurité de Flash Player 8 à l'adresse http://www.macromedia.com/go/fp8_security
- Livre blanc concernant les API liées à la sécurité de Flash Player 8 à l'adresse http://www.macromedia.com/go/fp8_security_apis

Utilisation 3 : La fonction `fscommand()` peut envoyer des messages à Macromedia Director. Ces messages sont interprétés par Lingo (le langage de script de Director) comme des chaînes, des événements ou un code Lingo exécutable. Si le message est une chaîne ou un événement, vous devez écrire le code Lingo devant recevoir le message de la fonction `fscommand()` et exécuter une action dans Director. Pour plus d'informations, consultez le centre d'assistance de Director à l'adresse www.macromedia.com/support/director.

Utilisation 4 : Dans VisualBasic, Visual C++ et dans d'autres programmes ayant recours aux contrôles ActiveX, la fonction `fscommand()` envoie un événement VB avec deux chaînes qui peut être traité dans le langage de programmation de l'environnement. Pour plus d'informations, utilisez les mots-clés « méthode Flash » pour effectuer une recherche dans le centre de support de Flash à l'adresse www.macromedia.com/support/flash.

*Remarque :* Si vous procédez à la publication de Flash Player 8 ou version ultérieure, la classe ExternalInterface offre de meilleures fonctionnalités de communication entre JavaScript et ActionScript (Utilisation 2), et entre ActionScript et VisualBasic, Visual C++ ou d'autres programmes pouvant héberger les contrôles ActiveX (Utilisation 4). Vous devriez continuer à utiliser la fonction `fscommand()` pour envoyer des messages à Flash Player (Utilisation 1) et à Macromedia Director (Utilisation 3).

**Disponibilité :** Flash Player 3 ; ActionScript 1.0

## Paramètres

*`command`*`:String` - Une chaîne transmise à l'application hôte ou une commande passée à Flash Player.

*`parameters`*`:String` - Une chaîne transmise à l'application hôte ou une valeur passée à Flash Player.

## Exemple

Dans l'exemple suivant, la fonction `fscommand()` définit Flash Player pour qu'il affiche le fichier SWF en taille plein écran lorsque le bouton `fullscreen_btn` ou `unfullscreen_btn` est relâché :

```
this.fullscreen_btn.onRelease = function() {
 fscommand("fullscreen", true);
};

this.unfullscreen_btn.onRelease = function() {
 fscommand("fullscreen", false);
};
```

L'exemple suivant applique la fonction `fscommand()` à un bouton dans Flash afin d'ouvrir une boîte de message JavaScript dans une page HTML. Le message est envoyé à JavaScript en tant que paramètre `fscommand`.

Vous devez ajouter une fonction à la page Web qui contient le fichier SWF. Cette fonction, `myDocument _DoFSCommand()`, attend un appel `fscommand()`. Lorsque la fonction `fscommand()` est déclenchée dans Flash (par exemple, lorsqu'un utilisateur clique sur le bouton), les chaînes `command` et `parameter` sont transmises à la fonction `myDocument _DoFSCommand()`. Vous pouvez utiliser les chaînes transmises dans votre code JavaScript ou VBScript à votre guise. Dans cet exemple, la fonction contient une instruction conditionnelle `if` qui vérifie si la chaîne de commande est `"messagebox"`. Si tel est le cas, un message d'alerte JavaScript affiche le contenu de la chaîne de la fonction `fscommand() parameters`.

```
function myDocument_DoFSCommand(command, args) {
 if (command == "messagebox") {
 alert(args);
 }
}
```

Dans le document Flash, ajoutez la fonction `fscommand()` à un bouton :

```
fscommand("messagebox", "This is a message box called from within Flash.")
```

Vous pouvez utiliser des expressions pour les paramètres de la fonction `fscommand()`, comme indiqué dans l'exemple suivant :

```
fscommand("messagebox", "Hello, " + name + ", welcome to our website!")
```

Pour tester le fichier SWF, pointez sur Fichier > Aperçu avant publication > HTML. Si vous publiez votre fichier SWF en utilisant le modèle Flash avec FSCommand (accessible depuis la boîte de dialogue Paramètres de publication après avoir sélectionné la balise HTML), Flash insère la fonction `myDocument_DoFSCommand()` automatiquement. Les attributs `NAME` et `ID` du fichier SWF constitueront le nom du fichier. Par exemple, pour le fichier myDocument.fla, les attributs seront définis sur `myDocument`.

### Voir également

`ExternalInterface (flash.external.ExternalInterface)`

## Fonction getProperty

```
getProperty(my_mc:Object, property:Object) : Object
```

Renvoie la valeur de la propriété spécifiée pour le clip *my_mc*.

**Disponibilité :** Flash Player 4 ; ActionScript 1.0

### Paramètres

*my_mc*`:String` - Le nom d'occurrence d'un clip pour lequel la propriété est extraite.

*property* - Une propriété d'un clip.

### Renvoie

`Object` - La valeur de la propriété spécifiée.

### Exemple

L'exemple suivant crée un nouveau clip `someClip_mc` et affiche la valeur alpha (`_alpha`) du clip `someClip_mc` dans le panneau de sortie :

```
this.createEmptyMovieClip("someClip_mc", 999);
trace("The alpha of "+getProperty(someClip_mc, _name)+" is:
  "+getProperty(someClip_mc, _alpha));
```

## Fonction getTimer

```
getTimer() : Number
```

Renvoie le nombre de millisecondes qui se sont écoulées depuis le début de la lecture du fichier SWF.

**Disponibilité :** Flash Player 4 ; ActionScript 1.0

### Renvoie

`Number` - Le nombre de millisecondes qui se sont écoulées depuis le début de la lecture du fichier SWF.

### Exemple

Dans l'exemple suivant, les fonctions `getTimer()` et `setInterval()` sont utilisées pour créer un minuteur simple :

```
this.createTextField("timer_txt", this.getNextHighestDepth(), 0, 0, 100,
  22);
function updateTimer():Void {
 timer_txt.text = getTimer();
}

var intervalID:Number = setInterval(updateTimer, 100);
```

## Fonction getURL

```
getURL(url:String, [window:String, [method:String]]) : Void
```

Charge un document en provenance d'une URL spécifique dans une fenêtre ou transmet des variables à une autre application à une URL donnée. Pour tester cette fonction, assurez-vous que le fichier à charger existe à l'emplacement prévu. Pour utiliser une URL absolue (par exemple, *`http://www.myserver.com`*), vous devez disposer d'une connexion réseau.

**Disponibilité :** Flash Player 4 ; ActionScript 1.0

## Paramètres

*url*:String - L'URL permettant d'obtenir le document.

*window*:String [facultatif] - Spécifie la fenêtre ou le cadre HTML dans lequel le document doit se charger. Vous pouvez entrer le nom d'une fenêtre spécifique ou le sélectionner à partir des noms cibles réservés suivants :

- _self spécifie le cadre actif de la fenêtre en cours d'utilisation.
- _blank crée une fenêtre.
- _parent appelle le parent du cadre actif.
- _top sélectionne le cadre de plus haut niveau de la fenêtre active.

*method*:String [facultatif] - Une méthode GET ou POST permettant d'envoyer des variables. En l'absence de variables, omettez ce paramètre. La méthode GET ajoute les variables à la fin de l'URL et est utilisée lorsque les variables sont peu nombreuses. La méthode POST place les variables dans un en-tête HTTP distinct et s'applique aux variables longues de type chaîne.

## Exemple

Cet exemple charge une image dans un clip. Lorsque l'utilisateur clique sur l'image, une nouvelle URL est chargée dans une nouvelle fenêtre de navigateur.

```
var listenerObject:Object = new Object();
listenerObject.onLoadInit = function(target_mc:MovieClip) {
 target_mc.onRelease = function() {
 getURL("http://www.macromedia.com/software/flash/flashpro/", "_blank");
 };
};
var logo:MovieClipLoader = new MovieClipLoader();
logo.addListener(listenerObject);
logo.loadClip("http://www.helpexamples.com/flash/images/image1.jpg",
 this.createEmptyMovieClip("macromedia_mc", this.getNextHighestDepth()));
```

Dans l'exemple suivant, la fonction getURL() est utilisée pour envoyer un message électronique :

```
myBtn_btn.onRelease = function(){
 getURL("mailto:you@somedomain.com");
};
```

Dans le code ActionScript suivant, JavaScript est utilisé pour ouvrir une fenêtre d'alerte lorsque le fichier SWF est intégré à une fenêtre de navigateur (sachez que lorsque vous appelez JavaScript à l'aide de getURL(), le paramètre url est limité à 508 caractères) :

```
myBtn_btn.onRelease = function(){
 getURL("javascript:alert('you clicked me')");
};
```

Vous pouvez également utiliser la méthode `GET` ou `POST` pour envoyer des variables. L'exemple suivant utilise la méthode `GET` pour ajouter des variables à une URL :

```
var firstName:String = "Gus";
var lastName:String = "Richardson";
var age:Number = 92;
myBtn_btn.onRelease = function() {
 getURL("http://www.macromedia.com", "_blank", "GET");
};
```

Le code ActionScript suivant utilise la méthode `POST` pour placer les variables dans l'en-tête HTTP. Assurez-vous de tester vos documents dans une fenêtre de navigateur ; sinon, vos variables sont envoyées à l'aide de la méthode `GET` :

```
var firstName:String = "Gus";
var lastName:String = "Richardson";
var age:Number = 92;
getURL("http://www.macromedia.com", "_blank", "POST");
```

### Voir également

`Fonction loadVariables`, `send (XML.send, méthode)`, `sendAndLoad (XML.sendAndLoad, méthode)`

## Fonction getVersion

```
getVersion() : String
```

Renvoie une chaîne contenant la version de Flash Player et des informations sur la plate-forme. La fonction `getVersion` ne renvoie des informations qu'à partir de la version 5 de Flash Player.

**Disponibilité :** Flash Player 5 ; ActionScript 1.0

### Renvoie

`String` - Une chaîne contenant la version de Flash Player et des informations sur la plate-forme.

### Exemple

Les exemples suivants identifient le numéro de version du lecteur Flash Player sur lequel est lu le fichier SWF :

```
var flashVersion:String = getVersion();
trace(flashVersion); // output: WIN 8,0,1,0
trace($version); // output: WIN 8,0,1,0
trace(System.capabilities.version); // output: WIN 8,0,1,0
```

La chaîne suivante est renvoyée par la fonction `getVersion` :

```
WIN 8,0,1,0
```

Cette chaîne renvoyée indique que la plate-forme utilisée est Microsoft Windows, et que le numéro de version de Flash Player est la version majeure 8, version mineure 1 (8.1).

### Voir également

`os (propriété capabilities.os)`, `version (propriété capabilities.version)`

## Fonction gotoAndPlay

```
gotoAndPlay([scene:String], frame:Object) : Void
```

Place la tête de lecture sur l'image spécifiée dans une séquence et commence la lecture à partir de cette image. Si aucune séquence n'est spécifiée, la tête de lecture passe à l'image spécifiée de la séquence en cours. Le paramètre *`scene`* est réservé au scénario racine. Vous ne pouvez pas l'utiliser dans les scénarios des clips ou autres objets du document.

**Disponibilité :** Flash Player 2 ; ActionScript 1.0

### Paramètres

*`scene`*`:String` [facultatif] - Une chaîne qui spécifie le nom de la séquence cible de la tête de lecture.

*`frame`*`:Object` - Un nombre représentant le numéro d'image ou une chaîne représentant l'étiquette de l'image cible de la tête de lecture.

### Exemple

Dans l'exemple suivant, un document contient deux séquences : `sceneOne` et `sceneTwo`. La séquence 1 contient une étiquette d'image sur l'image 10 intitulée `newFrame` et deux boutons, `myBtn_btn` et `myOtherBtn_btn`. Ce code ActionScript est placé sur l'image 1, séquence 1 du scénario principal.

```
stop();
myBtn_btn.onRelease = function(){
 gotoAndPlay("newFrame");
};

myOtherBtn_btn.onRelease = function(){
 gotoAndPlay("sceneTwo", 1);
};
```

Lorsque l'utilisateur clique sur les boutons, la tête de lecture se déplace à l'emplacement spécifié et continue la lecture.

### Voir également

gotoAndPlay (méthode MovieClip.gotoAndPlay), Fonction nextFrame, Fonction play, Fonction prevFrame

## Fonction gotoAndStop

```
gotoAndStop([scene:String], frame:Object) : Void
```

Place la tête de lecture sur l'image spécifiée sur une séquence et l'arrête à ce niveau. Si aucune séquence n'est spécifiée, la tête de lecture passe à l'image de la séquence en cours. Le paramètre *scene* est réservé au scénario racine. Vous ne pouvez pas l'utiliser dans les scénarios des clips ou autres objets du document.

**Disponibilité :** Flash Player 2 ; ActionScript 1.0

### Paramètres

*scene*:String [facultatif] - Une chaîne qui spécifie le nom de la séquence cible de la tête de lecture.

*frame*:Object - Un nombre représentant le numéro d'image ou une chaîne représentant l'étiquette de l'image cible de la tête de lecture.

### Exemple

Dans l'exemple suivant, un document contient deux séquences : sceneOne et sceneTwo. La séquence 1 contient une étiquette d'image sur l'image 10 intitulée newFrame et deux boutons, myBtn_btn et myOtherBtn_btn. Ce code ActionScript est placé sur l'image 1, séquence 1 du scénario principal :

```
stop();

myBtn_btn.onRelease = function(){
 gotoAndStop("newFrame");
};

myOtherBtn_btn.onRelease = function(){
 gotoAndStop("sceneTwo", 1);
};
```

Lorsque l'utilisateur clique sur les boutons, la tête de lecture se déplace à l'emplacement spécifié et arrête la lecture.

### Voir également

gotoAndStop (méthode MovieClip.gotoAndStop), Fonction stop, Fonction play, Fonction gotoAndPlay

## Fonction ifFrameLoaded

```
ifFrameLoaded([scene:String], frame) {
statement(s);
}
```

*Déconseillée* à partir de Flash Player 5. Cette fonction est déconseillée. Macromedia vous recommande d'utiliser la propriété `MovieClip._framesloaded`.

Vérifie si le contenu d'une image spécifique est disponible localement. Utilisez la fonction `ifFrameLoaded` pour commencer à lire une animation simple pendant le téléchargement du reste du fichier SWF sur l'ordinateur local. La différence d'utilisation entre les fonctions `_framesloaded` et `ifFrameLoaded` réside dans le fait que `_framesloaded` vous permet d'ajouter des instructions `if` ou `else` personnalisées.

**Disponibilité :** Flash Player 4 ; ActionScript 1.0

### Paramètres

*scene*:String [facultatif] - Une chaîne qui spécifie le nom de la séquence à charger.

*frame*:Object - Le numéro ou l'étiquette d'image devant être chargé(e) avant l'exécution de l'instruction suivante.

### Voir également

, addListener (méthode MovieClipLoader.addListener)

## Fonction int

```
int(value:Number) : Number
```

*Déconseillée* à partir de Flash Player 5. Il est recommandé d'utiliser la méthode `Math.round()`.

Convertit un nombre décimal en valeur entière en tronquant la valeur décimale. Cette fonction est l'équivalent de la fonction `Math.floor()` si le paramètre *value* est positif et de la fonction `Math.ceil()` si le paramètre *value* est négatif.

**Disponibilité :** Flash Player 4 ; ActionScript 1.0

### Paramètres

*value*:Number - Un nombre devant être arrondi à un entier.

### Renvoie

`Number` - Le nombre entier tronqué.

### Voir également

`round (méthode Math.round)`, `floor (méthode Math.floor)`, `ceil (méthode Math.ceil)`

## Fonction isFinite

```
isFinite(expression:Object) : Boolean
```

Evalue l'*`expression`* et renvoie `true` s'il s'agit d'un nombre fini ou `false` s'il s'agit de l'infini ou de l'infini négatif. La présence du signe infini ou infini négatif indique une erreur mathématique, telle que la division par 0.

**Disponibilité :** Flash Player 5 ; ActionScript 1.0

### Paramètres

*`expression`*`:Object` - Une valeur booléenne, une variable ou toute autre expression à évaluer.

### Renvoie

`Boolean` - Une valeur booléenne.

### Exemple

L'exemple suivant affiche les valeurs renvoyées pour `isFinite` :

```
isFinite(56)
// returns true

isFinite(Number.POSITIVE_INFINITY)
//returns false
```

## Fonction isNaN

```
isNaN(expression:Object) : Boolean
```

Evalue le paramètre et renvoie `true` si la valeur est `NaN` (not a number - n'est pas un nombre). Cette fonction permet de s'assurer qu'une expression mathématique a été évaluée correctement en tant que nombre.

**Disponibilité :** Flash Player 5 ; ActionScript 1.0

### Paramètres

*expression*:Object - Une valeur booléenne, une variable ou toute autre expression à évaluer.

### Renvoie

Boolean - Une valeur booléenne.

### Exemple

Le code suivant illustre les valeurs renvoyées pour la fonction isNaN() :

```
trace( isNaN("Tree") );
// returns true

trace( isNaN(56) );
// returns false

trace( isNaN(Number.POSITIVE_INFINITY) )
// returns false
```

L'exemple suivant indique comment utiliser la fonction isNAN() afin de vérifier si une expression mathématique contient une erreur :

```
var dividend:Number;
var divisor:Number;
divisor = 1;
trace( isNaN(dividend/divisor) );
// output: true
// The output is true because the variable dividend is undefined.
// Do not use isNAN() to check for division by 0 because it will return
  false.
// A positive number divided by 0 equals Infinity
  (Number.POSITIVE_INFINITY).
// A negative number divided by 0 equals -Infinity
  (Number.NEGATIVE_INFINITY).
```

### Voir également

Constante NaN, NaN (propriété Number.NaN)

## Fonction length

```
length(expression:String)length(variable)
```

*Déconseillée* à partir de Flash Player 5. Cette fonction, à l'instar de toutes les fonctions de chaîne, est déconseillée. Macromedia vous recommande d'utiliser les méthodes de la classe String et la propriété String.length pour effectuer les mêmes opérations.

Renvoie la longueur de la chaîne ou variable spécifiée.

**Disponibilité :** Flash Player 4 ; ActionScript 1.0

### Paramètres

*expression*:String - Une chaîne.

*variable*:Object - Le nom d'une variable.

### Renvoie

Number - La longueur de la chaîne ou variable spécifiée.

### Exemple

L'exemple suivant renvoie la longueur de la chaîne « Hello » : `length("Hello");` Le résultat est 5.

### Voir également

Opérateur " (séparateur de chaîne), String, length (propriété String.length)

## Fonction loadMovie

```
loadMovie(url:String, target:Object, [method:String]) : Void
loadMovie(url:String, target:String, [method:String]) : Void
```

Charge un fichier SWF, JPEG, GIF ou PNG dans un clip Flash Player lors de la lecture du fichier SWF d'origine. La prise en charge des fichiers GIF non animés, des fichiers PNG et des fichiers JPEG a été ajoutée à Flash Player 8. Si vous chargez un fichier GIF animé, seule la première image s'affiche.

*Conseil* : Si vous souhaitez contrôler la progression du téléchargement, utilisez `MovieClipLoader.loadClip()` à la place de cette fonction.

La fonction `loadMovie()` permet d'afficher plusieurs fichiers SWF à la fois et de basculer vers l'un de ces derniers sans avoir à charger un autre document HTML. En l'absence de la fonction `loadMovie()`, Flash Player affiche un seul fichier SWF.

Si vous souhaitez charger un fichier SWF ou JPEG à un niveau spécifique, utilisez `loadMovieNum()` à la place de `loadMovie()`.

Lorsqu'un fichier SWF est chargé dans un clip cible, vous pouvez utiliser le chemin cible de ce clip pour cibler le fichier SWF chargé. Un fichier SWF ou une image chargé dans une cible hérite de la position, des propriétés de rotation et d'échelle du clip ciblé. Le coin supérieur gauche de l'image chargée ou du fichier SWF s'aligne sur le point de référence du clip ciblé. Sinon, lorsque la cible correspond au scénario racine, le coin supérieur gauche de l'image ou du fichier SWF s'aligne sur le coin supérieur gauche de la scène.

La fonction `unloadMovie()` permet de supprimer les fichiers SWF chargés avec `loadMovie()`.

Lorsque vous utilisez cette fonction, référez-vous au modèle de sécurité de Flash Player.

Pour Flash Player 8 :

- Le chargement n'est pas autorisé si le clip appelant se trouve dans le Sandbox du système de fichiers local et si le clip chargé provient d'un Sandbox de réseau.
- Le chargement n'est pas autorisé si le fichier SWF appelant se trouve dans un Sandbox de réseau et si le clip à charger est local.
- L'accès au Sandbox de réseau à partir du Sandbox approuvé en local ou du Sandbox de réseau local requiert une autorisation émanant du site Web via un fichier de régulation interdomaines.
- Les clips qui se trouvent dans le Sandbox du système de fichiers local ne peuvent pas créer de scripts dans les clips du Sandbox de réseau local (et inversement).

Pour Flash Player 7 et version ultérieure :

- Les sites Web peuvent autoriser l'accès interdomaines à une ressource via un un fichier de régulation interdomaines.
- La programmation entre les fichiers SWF est limitée selon le domaine d'origine des fichiers SWF. Utilisez la méthode `System.security.allowDomain()` pour ajuster ces restrictions.

Pour plus d'informations, consultez le :

- Chapitre 17, « Fonctionnement de la sécurité » du guide *Formation à ActionScript 2.0 dans Flash*
- Livre blanc concernant la sécurité de Flash Player 8 à l'adresse http://www.macromedia.com/go/fp8_security
- Livre blanc concernant les API liées à la sécurité de Flash Player 8 à l'adresse http://www.macromedia.com/go/fp8_security_apis

**Disponibilité :** Flash Player 3 ; ActionScript 1.0

## Paramètres

*`url`*`:String` - L'URL absolue ou relative du fichier SWF ou JPEG à charger. Un chemin relatif doit être relatif au fichier SWF au niveau 0. Les URL absolues doivent inclure la référence de protocole, telle que http:// ou file:///.

*`target`*`:Object` - Une référence à un objet clip ou une chaîne représentant le chemin d'un clip cible. Le clip cible est remplacé par le fichier SWF chargé ou l'image.

*method*:String [facultatif] - Spécifie une méthode HTTP d'envoi des variables. Ce paramètre doit correspondre à la chaîne GET ou POST. En l'absence de variable à envoyer, omettez ce paramètre. La méthode GET ajoute les variables à la fin de l'URL et est utilisée lorsque les variables sont peu nombreuses. La méthode POST place les variables dans un en-tête HTTP distinct et s'applique aux variables longues de type chaîne.

## Exemple

Utilisation 1 : L'exemple suivant charge le fichier SWF circle.swf à partir du même répertoire et remplace un clip intitulé mySquare qui existe déjà sur la scène :

```
loadMovie("circle.swf", mySquare);
// equivalent statement (Usage 1): loadMovie("circle.swf",
  _level0.mySquare);
// equivalent statement (Usage 2): loadMovie("circle.swf", "mySquare");
```

L'exemple suivant charge le fichier SWF circle.swf à partir du même répertoire, mais remplace le clip principal au lieu du clip mySquare :

```
loadMovie("circle.swf", this);
// Note that using "this" as a string for the target parameter will not work
// equivalent statement (Usage 2): loadMovie("circle.swf", "_level0");
```

L'instruction loadMovie() suivante charge le fichier SWF sub.swf à partir du même répertoire dans un nouveau clip intitulé logo_mc, créé à l'aide de createEmptyMovieClip() :

```
this.createEmptyMovieClip("logo_mc", 999);
loadMovie("sub.swf", logo_mc);
```

Vous pouvez ajouter le code suivant pour charger une image JPEG intitulée image1.jpg à partir du même répertoire que le fichier SWF chargeant sub.swf. L'image JPEG est chargée lorsque vous cliquez sur un bouton intitulé myBtn_btn. Ce code charge l'image JPEG dans logo_mc. Par conséquent, il remplace sub.swf par l'image JPEG.

```
myBtn_btn.onRelease = function(){
 loadMovie("image1.jpg", logo_mc);
};
```

Utilisation 2 : L'exemple suivant charge le fichier SWF circle.swf à partir du même répertoire et remplace un clip intitulé mySquare qui existe déjà sur la scène :

```
loadMovie("circle.swf", "mySquare");
```

## Voir également

_level, propriété, Fonction loadMovieNum, loadMovie (méthode MovieClip.loadMovie), loadClip (méthode MovieClipLoader.loadClip), Fonction unloadMovie

# Fonction loadMovieNum

`loadMovieNum(url:String, level:Number, [method:String]) : Void`

Charge un fichier SWF, JPEG, GIF ou PNG dans un niveau lors de la lecture du fichier SWF d'origine. La prise en charge des fichiers GIF non animés, des fichiers PNG et des fichiers JPEG a été ajoutée à Flash Player 8. Si vous chargez un fichier GIF animé, seule la première image s'affiche.

*Conseil* : Si vous souhaitez contrôler la progression du téléchargement, utilisez `MovieClipLoader.loadClip()` à la place de cette fonction.

Normalement, Flash Player affiche un fichier SWF, puis se ferme. L'action `loadMovieNum()` permet d'afficher plusieurs fichiers SWF à la fois et de basculer vers l'un de ces derniers sans avoir à charger un autre document HTML.

Si vous souhaitez spécifier une cible et non pas un niveau, utilisez `loadMovie()` à la place de `loadMovieNum()`.

Flash Player empile les différents niveaux en commençant par le niveau 0. Ces niveaux correspondent à des feuilles de papier calque empilées les unes sur les autres, ils sont transparents à l'exception des objets placés à chaque niveau. Lorsque vous utilisez `loadMovieNum()`, vous devez spécifier le niveau de Flash Player devant recevoir le fichier SWF à charger. Lorsqu'un fichier SWF est chargé dans un niveau, utilisez la syntaxe, `_level N`, où `N` correspond au numéro du niveau cible.

Lorsque vous chargez un fichier SWF, vous pouvez spécifier le niveau de votre choix et charger des fichiers SWF dans un niveau qui comporte déjà un fichier de ce type. Dans ce cas, le nouveau fichier SWF remplace le fichier existant. Si vous chargez un fichier SWF dans le niveau 0, tous les autres niveaux de Flash Player sont vidés et le niveau 0 utilise le nouveau fichier. Le fichier SWF du niveau 0 définit le débit d'images, la couleur d'arrière-plan et la taille d'image de tous les autres fichiers SWF chargés.

L'action `loadMovieNum()` permet également de charger des fichiers JPEG dans un fichier SWF en cours de lecture. Pour les images et les fichiers SWF, le coin supérieur gauche de l'image s'aligne sur le coin supérieur gauche de la scène pendant le chargement du fichier. Dans les deux cas, le fichier chargé hérite des paramètres de rotation et de mise à l'échelle, et le contenu d'origine est remplacé au niveau spécifié.

*Remarque* : Les fichiers JPEG enregistrés au format progressif ne sont pas pris en charge.

La fonction `unloadMovieNum()` permet de supprimer des fichiers SWF ou des images qui ont été chargés avec `loadMovieNum()`.

Lorque vous utilisez cette méthode, prenez en considération le modèle de sécurité de Flash Player.

Pour Flash Player 8 :

- Le chargement n'est pas autorisé si le clip appelant se trouve dans le Sandbox du système de fichiers local et si le clip chargé provient d'un Sandbox de réseau.
- Le chargement n'est pas autorisé si le fichier SWF appelant se trouve dans un Sandbox de réseau et si le clip à charger est local.
- L'accès au Sandbox de réseau à partir du Sandbox approuvé en local ou du Sandbox de réseau local requiert une autorisation émanant du site Web via un fichier de régulation interdomaines.
- Les clips qui se trouvent dans le Sandbox du système de fichiers local ne peuvent pas créer de scripts dans les clips du Sandbox de réseau local (et inversement).

Pour Flash Player 7 et version ultérieure :

- Les sites Web peuvent autoriser l'accès interdomaines à une ressource via un un fichier de régulation interdomaines.
- La programmation entre les fichiers SWF est limitée selon le domaine d'origine des fichiers SWF. Utilisez la méthode `System.security.allowDomain()` pour ajuster ces restrictions.

Pour plus d'informations, consultez le :

- Chapitre 17, « Fonctionnement de la sécurité » du guide *Formation à ActionScript 2.0 dans Flash*
- Livre blanc concernant la sécurité de Flash Player 8 à l'adresse http://www.macromedia.com/go/fp8_security
- Livre blanc concernant les API liées à la sécurité de Flash Player 8 à l'adresse http://www.macromedia.com/go/fp8_security_apis

**Disponibilité :** Flash Player 4 ; ActionScript 1.0

## Paramètres

*`url`*`:String` - L'URL absolue ou relative du fichier SWF ou JPEG à charger. Un chemin relatif doit faire référence au fichier SWF du niveau 0. Pour l'utilisation avec une version autonome de Flash Player ou en mode test dans l'application de programmation Flash, tous les fichiers SWF doivent être stockés dans le même dossier et les noms de fichier ne doivent pas inclure de spécifications de dossier ou lecteur de disque.

*`level`*`:Number` - Un entier spécifiant le niveau de Flash Player dans lequel le fichier SWF doit se charger.

*method*:String [facultatif] - Spécifie une méthode HTTP d'envoi des variables. Ce paramètre doit correspondre à la chaîne GET ou POST. En l'absence de variable à envoyer, omettez ce paramètre. La méthode GET ajoute les variables à la fin de l'URL et est utilisée lorsque les variables sont peu nombreuses. La méthode POST place les variables dans un en-tête HTTP distinct et s'applique aux variables longues de type chaîne.

### Exemple

L'exemple suivant permet de charger l'image JPEG tim.jpg dans le niveau 2 de Flash Player :

```
loadMovieNum("http://www.helpexamples.com/flash/images/image1.jpg", 2);
```

### Voir également

Fonction unloadMovieNum, Fonction loadMovie, loadClip (méthode MovieClipLoader.loadClip), _level, propriété

## Fonction loadVariables

```
loadVariables(url:String, target:Object, [method:String]) : Void
```

Lit les données dans un fichier externe, tel qu'un fichier texte ou du texte généré par ColdFusion, un script CGI, des pages ASP (Active Server Pages), PHP ou un script Perl et définit les valeurs pour les variables dans un clip cible. Cette action permet également de mettre à jour les variables du fichier SWF actif en fonction des nouvelles valeurs.

Le texte de l'URL spécifiée doit être au format MIME standard *application/x-www-form-urlencoded* (un format standard utilisé par les scripts CGI). Vous pouvez spécifier autant de variables que nécessaire. Par exemple, cette séquence définit plusieurs variables :

```
company=Macromedia&address=600+Townsend&city=San+Francisco&zip=94103
```

Dans les fichiers SWF exécutés dans une version antérieure à Flash Player 7, l'*url* doit être dans le même superdomaine que le fichier SWF qui transmet cet appel. Un superdomaine est dérivé en supprimant le composant le plus à gauche de l'URL d'un fichier. Par exemple, un fichier SWF enregistré dans www.someDomain.com peut charger des données à partir d'une source figurant dans store.someDomain.com, car les deux fichiers appartiennent au même superdomaine que someDomain.com.

Dans les fichiers SWF, quelle que soit leur version, qui s'exécutent dans Flash Player 7 ou version ultérieure, *url* doit figurer dans le même domaine que le fichier SWF qui envoie cet appel (voir « Fonctions de sécurité de Flash Player » dans le guide *Utilisation d'ActionScript dans Flash*). Par exemple, un fichier SWF situé à l'adresse www.someDomain.com peut charger des données en provenance de sources qui figurent également à l'adresse www.someDomain.com. Si vous souhaitez charger des données à partir d'un autre domaine, vous pouvez placer un *fichier de régulation interdomaines* sur le serveur hébergeant le fichier SWF en cours d'accès. Pour plus d'informations, voir « A propos de l'autorisation de chargement de données interdomaines » dans *Utilisation d'ActionScript dans Flash*.

Si vous souhaitez charger des variables dans un niveau spécifique, utilisez `loadVariablesNum()` à la place de `loadVariables()`.

**Disponibilité :** Flash Player 4 ; ActionScript 1.0

## Paramètres

*url*:`String` - Une URL absolue ou relative par rapport à l'emplacement des variables. Si le fichier SWF effectuant cet appel s'exécute dans un navigateur Web, *url* doit appartenir au même domaine que le fichier SWF. Pour plus de détails, reportez-vous à la section Description.

*target*:`Object` - Le chemin cible d'un clip devant recevoir les variables chargées.

*method*:`String` [facultatif] - Spécifie une méthode HTTP d'envoi des variables. Ce paramètre doit correspondre à la chaîne `GET` ou `POST`. En l'absence de variable à envoyer, omettez ce paramètre. La méthode `GET` ajoute les variables à la fin de l'URL et est utilisée lorsque les variables sont peu nombreuses. La méthode `POST` place les variables dans un en-tête HTTP distinct et s'applique aux variables longues de type chaîne.

## Exemple

L'exemple suivant permet de charger les informations d'un fichier texte intitulé params.txt dans le clip `target_mc` créé à l'aide de `createEmptyMovieClip()`. La fonction `setInterval()` permet de vérifier la progression du chargement. Le script recherche une variable dans le fichier params.txt appelé `done`.

```
this.createEmptyMovieClip("target_mc", this.getNextHighestDepth());
loadVariables("params.txt", target_mc);
function checkParamsLoaded() {
 if (target_mc.done == undefined) {
 trace("not yet.");
 } else {
 trace("finished loading. killing interval.");
 trace("-------------");
 for (i in target_mc) {
```

```
 trace(i+": "+target_mc[i]);
 }
 trace("-------------");
 clearInterval(param_interval);
 }
}
var param_interval:Number = setInterval(checkParamsLoaded, 100);
```

Le fichier externe, `params.txt`, inclut le texte suivant :

```
var1="hello"&var2="goodbye"&done="done"
```

### Voir également

`Fonction loadVariablesNum`, `Fonction loadMovie`, `Fonction loadMovieNum`, `Fonction getURL`, `loadMovie (méthode MovieClip.loadMovie)`, `loadVariables (méthode MovieClip.loadVariables)`, `load (méthode LoadVars.load)`

## Fonction loadVariablesNum

```
loadVariablesNum(url:String, level:Number, [method:String]) : Void
```

Lit les données dans un fichier externe, tel qu'un fichier texte ou du texte généré par ColdFusion, un script CGI, des pages ASP (Active Server Pages), PHP ou un script Perl et définit les valeurs pour les variables dans un niveau de Flash Player. Vous pouvez également utiliser cette fonction pour mettre à jour les variables du fichier SWF actif afin de tenir compte des nouvelles valeurs.

Le texte de l'URL spécifiée doit être au format MIME standard *`application/x-www-form-urlencoded`*(un format standard utilisé par les scripts CGI). Vous pouvez spécifier autant de variables que nécessaire. Par exemple, cette séquence définit plusieurs variables :

```
company=Macromedia&address=601+Townsend&city=San+Francisco&zip=94103
```

Pour les fichiers SWF lus par une version antérieure à Flash Player 7, le paramètre *`url`* doit correspondre au superdomaine du fichier SWF envoyant cet appel. Un superdomaine est dérivé en supprimant le composant le plus à gauche de l'URL d'un fichier. Par exemple, un fichier SWF à l'adresse www.someDomain.com peut charger des données à partir d'une source à l'adresse store.someDomain.com dans la mesure où les deux fichiers figurent dans le même superdomaine que someDomain.com.

Dans les fichiers SWF, quelle que soit leur version, qui s'exécutent dans Flash Player 7 ou version ultérieure, *url* doit figurer dans le même domaine que le fichier SWF qui envoie cet appel (voir « Fonctions de sécurité de Flash Player » dans le guide *Utilisation d'ActionScript dans Flash*). Par exemple, un fichier SWF à l'adresse www.someDomain.com peut charger des données en provenance de sources qui figurent également à l'adresse www.someDomain.com. Si vous souhaitez charger des données à partir d'un autre domaine, vous pouvez placer un *fichier de régulation interdomaines* sur le serveur hébergeant le fichier SWF. Pour plus d'informations, voir « A propos de l'autorisation de chargement de données interdomaines » dans *Utilisation d'ActionScript dans Flash*.

Si vous souhaitez charger des variables dans un clip cible, utilisez `loadVariables()` à la place de `loadVariablesNum()`.

**Disponibilité :** Flash Player 4 ; ActionScript 1.0

## Paramètres

*url*:`String` - Une URL absolue ou relative par rapport à l'emplacement des variables. Si le fichier SWF effectuant cet appel s'exécute dans un navigateur Web, *url* doit appartenir au même domaine que le fichier SWF. Pour plus de détails, reportez-vous à la section Description.

*level*:`Number` - Un entier spécifiant le niveau de Flash Player devant recevoir les variables.

*method*:`String` [facultatif] - Spécifie une méthode HTTP d'envoi des variables. Ce paramètre doit correspondre à la chaîne `GET` ou `POST`. En l'absence de variable à envoyer, omettez ce paramètre. La méthode `GET` ajoute les variables à la fin de l'URL et est utilisée lorsque les variables sont peu nombreuses. La méthode `POST` place les variables dans un en-tête HTTP distinct et s'applique aux variables longues de type chaîne.

## Exemple

L'exemple suivant permet de charger les informations d'un fichier texte intitulé params.txt dans le scénario principal du fichier SWF au niveau 2 dans Flash Player. Les noms de variables des champs de texte doivent correspondre à ceux du fichier params.txt. La fonction `setInterval()` est utilisée pour vérifier la progression du chargement des données dans le fichier SWF. Le script recherche une variable dans le fichier params.txt appelé `done`.

```
loadVariablesNum("params.txt", 2);
function checkParamsLoaded() {
 if (_level2.done == undefined) {
 trace("not yet.");
 } else {
 trace("finished loading. killing interval.");
 trace("-------------");
 for (i in _level2) {
```

```
 trace(i+": "+_level2[i]);
 }
 trace("-------------");
 clearInterval(param_interval);
 }
}
var param_interval:Number = setInterval(checkParamsLoaded, 100);

// Params.txt includes the following text
var1="hello"&var2="goodbye"&done="done"
```

### Voir également

Fonction getURL, Fonction loadMovie, Fonction loadMovieNum, Fonction loadVariables, loadMovie (méthode MovieClip.loadMovie), loadVariables (méthode MovieClip.loadVariables), load (méthode LoadVars.load)

## Fonction mbchr

```
mbchr(number:Number)
```

*Déconseillée* à partir de Flash Player 5. Il est recommandé d'utiliser la méthode `String.fromCharCode()`.

Convertit un numéro de code ASCII en caractère multi-octets.

**Disponibilité :** Flash Player 4 ; ActionScript 1.0

### Paramètres

*number*:Number - Le nombre à convertir en caractère multi-octets.

### Voir également

fromCharCode (méthode String.fromCharCode)

## Fonction mblength

```
mblength(string:String) : Number
```

*Déconseillée* à partir de Flash Player 5. Il est recommandé d'utiliser les méthodes et les propriétés de la classe String.

Renvoie la longueur de la chaîne de caractères multi-octets.

**Disponibilité :** Flash Player 4 ; ActionScript 1.0

### Paramètres

*string*:String - La chaîne à mesurer.

### Renvoie

Number - La longueur de la chaîne de caractères multi-octets.

### Voir également

String, length (propriété String.length)

## Fonction mbord

mbord(character:String) : Number

*Déconseillée* à partir de Flash Player 5. Il est recommandé d'utiliser la méthode String.charCodeAt().

Convertit le caractère spécifié en nombre multi-octets.

**Disponibilité :** Flash Player 4 ; ActionScript 1.0

### Paramètres

*character*:String - *character* Le caractère à convertir en nombre multi-octets.

### Renvoie

Number - Le caractère converti.

### Voir également

charCodeAt (méthode String.charCodeAt)

## Fonction mbsubstring

mbsubstring(value:String, index:Number, count:Number) : String

*Déconseillée* à partir de Flash Player 5. Il est recommandé d'utiliser la méthode String.substr().

Extrait une nouvelle chaîne de caractères multi-octets d'une chaîne de caractères multi-octets.

**Disponibilité :** Flash Player 4 ; ActionScript 1.0

### Paramètres

*value*:String - La chaîne multi-octets à partir de laquelle il convient d'extraire une nouvelle chaîne multi-octets.

*index*:Number - Le numéro du premier caractère à extraire.

*count*:Number - Le nombre de caractères à inclure dans la chaîne extraite, caractère d'indice non compris.

### Renvoie

String - La chaîne extraite à partir de la chaîne de caractères multi-octets.

### Voir également

substr (méthode String.substr)

## Fonction MMExecute

MMExecute(*"Flash JavaScript API command;":String*) : String

Permet d'émettre des commandes de l'API Flash JavaScript (JSAPI) à partir d'ActionScript. Dans Flash MX 2004, la fonction MMExecute doit être appelée par une animation utilisée en tant que panneau Flash (fichier stocké dans le répertoire WindowSWF), par une boîte de dialogue XMLtoUI ou par l'interface utilisateur personnalisée d'un composant. Les commandes JSAPI n'ont aucun effet dans le lecteur, en mode test d'animation, ou en dehors de l'environnement de programmation.

La JSAPI de Flash comporte plusieurs objets, méthodes et propriétés permettant de dupliquer ou d'émuler les commandes pouvant être entrées par un utilisateur dans l'environnement de programmation. La JSAPI permet d'écrire des scripts qui développent Flash de plusieurs façons : ajout de commandes à des menus, manipulation d'objets sur la scène, répétition de séquences de commandes, etc.

De manière générale, un utilisateur exécute un script JSAPI en sélectionnant Commandes > Exécuter la commande. Cependant, vous pouvez utiliser cette fonction dans un script ActionScript pour appeler directement une commande JSAPI. Si vous utilisez MMExecute() dans un script pour l'image 1 de votre fichier, la commande s'exécute lors du chargement du fichier SWF.

Pour plus d'informations sur la JSAPI, voir www.macromedia.com/go/jsapi_info_en.

**Disponibilité :** Flash Player 7 ; ActionScript 1.0

### Paramètres

*command*:String - Toute commande pouvant être utilisée dans un fichier JSFL (Flash JavaScript).

### Renvoie

`String` - Le résultat représenté sous forme de chaîne, s'il y en a un, renvoyé par l'instruction JavaScript.

### Exemple

La commande suivante renvoie le nombre d'éléments contenus dans la bibliothèque du document actuel dans la fenêtre de trace. Vous devez exécuter cet exemple en tant que panneau Flash car les fichiers Flash ne peuvent pas appeler `MMExecute` s'ils sont exécutés en mode Tester l'animation ou dans le navigateur.

- Placez le code suivant dans l'image 1 du scénario principal d'un document Flash vide :

```
var numLibItems = MMExecute("fl.getDocumentDOM().library.items.length");

var message = numLibItems + " items in library";

MMExecute('fl.trace("' + message + '");');
```

- Enregistrez le fichier FLA dans le répertoire WindowSWF de votre répertoire Configuration, puis pointez sur Fichier > Publier (ou enregistrez-le ailleurs et publiez le fichier SWF directement dans ce répertoire, ou déplacez-le dans ce répertoire).
- Quittez et redémarrez l'application (vous devez effectuer cette étape lorsque vous ajoutez votre fichier dans le répertoire WindowSWF pour la première fois).

Vous pouvez désormais sélectionner votre fichier dans la partie inférieure du menu Fenêtre > Autres panneaux.

La fonction trace d'ActionScript ne fonctionne pas dans un panneau Flash ; cet exemple utilise la version JavaScript `fl.trace` pour obtenir la sortie. La copie des résultats de `MMExecute` dans un champ de texte faisant partie du fichier de votre panneau Flash peut s'avérer plus facile.

## Fonction nextFrame

```
nextFrame() : Void
```

Place la tête de lecture sur l'image suivante.

**Disponibilité :** Flash Player 2 ; ActionScript 1.0

### Exemple

Dans l'exemple suivant, lorsque l'utilisateur appuie sur la flèche droite ou bas, la tête de lecture se déplace jusqu'à l'image suivante et s'arrête. Si l'utilisateur appuie sur la flèche gauche ou haut, la tête de lecture se positionne sur l'image précédente et s'arrête. L'écouteur est initialisé pour attendre que l'utilisateur appuie sur la touche de direction et la variable `init` est utilisée pour empêcher que l'écouteur soit redéfini si la tête de lecture se repositionne sur l'image 1.

```
stop();

if (init == undefined) {
 someListener = new Object();
 someListener.onKeyDown = function() {
 if (Key.isDown(Key.LEFT) || Key.isDown(Key.UP)) {
 _level0.prevFrame();
 } else if (Key.isDown(Key.RIGHT) || Key.isDown(Key.DOWN)) {
 _level0.nextFrame();
 }
 };
 Key.addListener(someListener);
 init = 1;
}
```

### Voir également

`Fonction prevFrame`

## Fonction nextScene

```
nextScene() : Void
```

Place la tête de lecture sur l'image 1 de la séquence suivante.

**Disponibilité :** Flash Player 2 ; ActionScript 1.0

### Exemple

Dans l'exemple suivant, lorsqu'un utilisateur clique sur le bouton créé à l'exécution, la tête de lecture est positionnée sur l'image 1 de la séquence suivante. Créez deux séquences, puis entrez le code ActionScript suivant sur l'image 1 de la séquence 1.

```
stop();

if (init == undefined) {
 this.createEmptyMovieClip("nextscene_mc", this.getNextHighestDepth());
 nextscene_mc.createTextField("nextscene_txt", this.getNextHighestDepth(),
  200, 0, 100, 22);
```

```
 nextscene_mc.nextscene_txt.autoSize = true;
 nextscene_mc.nextscene_txt.border = true;
 nextscene_mc.nextscene_txt.text = "Next Scene";
 this.createEmptyMovieClip("prevscene_mc", this.getNextHighestDepth());
 prevscene_mc.createTextField("prevscene_txt", this.getNextHighestDepth(),
  00, 0, 100, 22);
 prevscene_mc.prevscene_txt.autoSize = true;
 prevscene_mc.prevscene_txt.border = true;
 prevscene_mc.prevscene_txt.text = "Prev Scene";
 nextscene_mc.onRelease = function() {
 nextScene();
 };

 prevscene_mc.onRelease = function() {
 prevScene();
 };

 init = true;
}
```

Assurez-vous de placer une action `stop()` sur l'image 1 de la séquence 2.

### Voir également

`Fonction prevScene`

## Fonction Number

```
Number(expression) : Number
```

Convertit le paramètre *`expression`* en nombre et renvoie une valeur comme indiqué dans la liste suivante :

- Si *`expression`* est un nombre, la valeur renvoyée est *`expression`*.
- Si *`expression`* est une valeur booléenne, la valeur renvoyée est 1 si *`expression`* est `true`, 0 si *`expression`* est `false`.
- Si *`expression`* est une chaîne, la fonction tente d'analyser *`expression`* en tant que nombre décimal avec un exposant facultatif à la fin (ainsi, 1,57505e-3).
- Si *`expression`* est `NaN`, la valeur renvoyée est `NaN`.
- Si *`expression`* est `undefined`, renvoie l'une des valeurs suivantes : - Dans les fichiers publiés pour Flash Player 6 ou version antérieure, le résultat est 0. - Dans les fichiers publiés pour Flash Player 7 ou version ultérieure, le résultat est `NaN`.

**Disponibilité :** Flash Player 4 ; ActionScript 1.0

### Paramètres

*expression*:Object - Une expression à convertir en nombre. Les nombres ou chaînes commençant par 0x sont interprété(e)s en tant que valeurs hexadécimales. Les nombres ou chaînes commençant par 0 sont interprété(e)s en tant que valeurs octales.

### Renvoie

Number - Un nombre ou NaN (n'est pas un nombre).

### Exemple

Dans l'exemple suivant, un champ de texte est créé sur la scène à l'exécution :

```
this.createTextField("counter_txt", this.getNextHighestDepth(), 0, 0, 100,
  22);
counter_txt.autoSize = true;
counter_txt.text = 0;
function incrementInterval():Void {
 var counter:Number = counter_txt.text;
 // Without the Number() function, Flash would concatenate the value instead
 // of adding values. You could also use "counter_txt.text++;"
 counter_txt.text = Number(counter) + 1;
}
var intervalID:Number = setInterval(incrementInterval, 1000);
```

### Voir également

Constante NaN, Number, Fonction parseInt, Fonction parseFloat

## Fonction Object

```
Object([value:Object]) : Object
```

Crée un objet vide ou convertit le nombre, la chaîne ou la valeur booléenne spécifié en objet. Cette commande revient à créer un objet avec le constructeur Object (voir « Constructeur de la classe Object »).

**Disponibilité :** Flash Player 5 ; ActionScript 1.0

### Paramètres

*value*:Object [facultatif] - Une valeur de type numérique, chaîne ou booléen.

### Renvoie

Object - Un objet.

### Exemple

Dans l'exemple suivant, un objet vide est créé, puis renseigné par des valeurs :

```
var company:Object = new Object();
company.name = "Macromedia, Inc.";
company.address = "600 Townsend Street";
company.city = "San Francisco";
company.state = "CA";
company.postal = "94103";
for (var i in company) {
 trace("company."+i+" = "+company[i]);
}
```

### Voir également

`Object`

## Gestionnaire on

`on(`*mouseEvent:Object*`) {`
*// your statements here*
`}`

Spécifie l'événement de type souris ou pression de touche devant déclencher une action.

**Disponibilité :** Flash Player 2 ; ActionScript 1.0

### Paramètres

*`mouseEvent`*`:Object` - *`mouseEvent`* est un déclencheur appelé par un événement *event*. Lorsque cet événement se produit, les instructions qui le suivent entre accolades ({ }) s'exécutent. Vous pouvez spécifier n'importe laquelle des valeurs suivantes pour le paramètre *`mouseEvent`* :

- `press` L'utilisateur appuie sur le bouton de la souris pendant que le pointeur de la souris survole le bouton.
- `release` L'utilisateur relâche le bouton de la souris pendant que le pointeur de la souris survole le bouton.
- `releaseOutside` Pendant que le pointeur de la souris survole le bouton, l'utilisateur appuie sur le bouton de la souris puis éloigne le pointeur du bouton juste avant le relâchement du bouton. Les événements `press` et `dragOut` précèdent toujours l'événement `releaseOutside`.
- `rollOut` Le pointeur quitte la zone du bouton.
- `rollOver` Le pointeur de la souris survole le bouton.

- `dragOut` Pendant que le pointeur de la souris survole le bouton, l'utilisateur appuie sur le bouton de la souris puis place le pointeur en dehors de la zone du bouton.
- `dragOver` Pendant que le pointeur est au-dessus du bouton, l'utilisateur appuie sur le bouton de la souris, fait glisser le pointeur en dehors de la zone du bouton, puis le ramène sur ce dernier.
- `keyPress « < key > »` L'utilisateur appuie sur la touche spécifiée du clavier. Pour la section *key* du paramètre, spécifiez une constante de touche, comme indiqué par le conseil de code du panneau Actions. Vous pouvez utiliser ce paramètre pour intercepter l'utilisation d'une touche, ce qui revient à contourner le comportement intégré de la touche spécifiée. L'emplacement du bouton n'a pas d'importance, il peut être sur la scène ou en dehors. L'une des limites de cette technique est que vous ne pouvez pas appliquer le gestionnaire `on()` pendant l'exécution ; vous devez l'appliquer pendant la programmation. Assurez-vous que Contrôle > Désactiver les raccourcis clavier est sélectionné ou que les touches associées à un comportement intégré ne seront pas ignorées lorsque vous testez l'application avec Contrôle > Tester l'animation.

Pour consulter la liste des constantes de touches, voir la classe Key.

### Exemple

Dans le script suivant, la fonction `startDrag()` s'exécute lorsque l'utilisateur clique sur le bouton de la souris et le script conditionnel est exécuté lorsqu'il relâche le bouton de la souris et que l'objet est déposé :

```
on (press) {
 startDrag(this);
}
on (release) {
 trace("X:"+this._x);
 trace("Y:"+this._y);
 stopDrag();
}
```

### Voir également

`Gestionnaire onClipEvent`, `Key`

## Gestionnaire onClipEvent

```
onClipEvent(movieEvent:Object) {
// your statements here
}
```

Déclenche les actions définies pour une instance spécifique de clip.

**Disponibilité :** Flash Player 5 ; ActionScript 1.0

## Paramètres

*movieEvent*:Object - *movieEvent* est un déclencheur appelé par un événement *event*. Lorsque cet événement se produit, les instructions qui le suivent entre accolades ({ }) s'exécutent. Vous pouvez spécifier n'importe laquelle des valeurs suivantes pour le paramètre *movieEvent* :

- `load` L'action commence dès que le clip est instancié et s'affiche dans le scénario.
- `unload` L'action commence dès la première image, après que le clip est supprimé du scénario. Les actions associées à l'événement `Unload` du clip sont traitées avant les actions associées à l'image affectée.
- `enterFrame` L'action est déclenchée de façon continue en suivant le débit d'images du clip. Les actions associées à l'événement `enterFrame` du clip sont traitées avant les actions sur les images associées aux images affectées.
- `mouseMove` L'action commence dès que la souris bouge. Les propriétés `_xmouse` et `_ymouse` permettent de déterminer la position du curseur.
- `mouseDown` L'action commence dès que l'utilisateur appuie sur le bouton gauche de la souris.
- `mouseUp` L'action commence dès que l'utilisateur relâche le bouton gauche de la souris.
- `keyDown` L'action commence dès que l'utilisateur appuie sur une touche. La méthode `Key.getCode()` permet d'extraire des informations sur la dernière touche utilisée.
- `keyUp` L'action commence dès que l'utilisateur relâche une touche. La méthode `Key.getCode()` permet d'extraire des informations sur la dernière touche utilisée.
- `data` L'action commence dès que des données sont reçues par une action `loadVariables()` ou `loadMovie()`. Lorsque ce paramètre est spécifié avec une action `loadVariables()`, l'événement `data` ne se produit qu'une seule fois, lorsque la dernière variable est chargée. Par contre, lorsqu'il est spécifié avec une action `loadMovie()`, l'événement `data` se répète, lors de la réception de chaque section de données.

## Exemple

L'exemple suivant utilise `onClipEvent()` avec l'événement de clip `keyDown` et est conçu pour être associé à un clip ou bouton. L'événement de clip `keyDown` est généralement utilisé avec une ou plusieurs méthodes et propriétés de l'objet Key. Le script suivant utilise `Key.getCode()` pour savoir sur quelle touche l'utilisateur a appuyé ; si la touche sur laquelle il a appuyé correspond à la propriété `Key.RIGHT`, la tête de lecture est positionnée sur l'image suivante ; si elle correspond à la propriété `Key.LEFT`, la tête de lecture est positionnée sur l'image précédente.

```
onClipEvent (keyDown) {
 if (Key.getCode() == Key.RIGHT) {
```

```
 this._parent.nextFrame();
 } else if (Key.getCode() == Key.LEFT) {
 this._parent.prevFrame();
 }
}
```

L'exemple suivant utilise `onClipEvent()` avec les événements de clips `load` et `mouseMove`. Les propriétés `_xmouse` et `_ymouse` suivent la position de la souris à chaque fois qu'elle se déplace, apparaissant dans le champ de texte créé à l'exécution.

```
onClipEvent (load) {
 this.createTextField("coords_txt", this.getNextHighestDepth(), 0, 0, 100,
  22);
 coords_txt.autoSize = true;
 coords_txt.selectable = false;
}
onClipEvent (mouseMove) {
 coords_txt.text = "X:"+_root._xmouse+",Y:"+_root._ymouse;
}
```

### Voir également

`Key`, `_xmouse (propriété MovieClip._xmouse)`, `_ymouse (propriété MovieClip._ymouse)`, `Gestionnaire on`, `Fonction updateAfterEvent`

## Fonction ord

```
ord(character:String) : Number
```

*Déconseillée* à partir de Flash Player 5. Il est recommandé d'utiliser les méthodes et les propriétés de la classe String.

Convertit les caractères en numéros de code ASCII.

**Disponibilité :** Flash Player 4 ; ActionScript 1.0

### Paramètres

*`character`*`:String` - Le caractère à convertir en numéro de code ASCII.

### Renvoie

`Number` - Le numéro de code ASCII du caractère spécifié.

### Voir également

`String`, `charCodeAt (méthode String.charCodeAt)`

# Fonction parseFloat

```
parseFloat(string:String) : Number
```

Convertit une chaîne en nombre à virgule flottante. Cette fonction lit, ou *analyse*, et renvoie les nombres dans une chaîne jusqu'à ce que cette dernière atteigne un caractère qui ne fait pas partie du nombre initial. Si la chaîne ne commence pas par un nombre qui peut être analysé, `parseFloat()` renvoie `NaN`. L'espace blanc qui précède un entier valide est ignoré, comme les caractères de fin non numériques.

**Disponibilité :** Flash Player 5 ; ActionScript 1.0

## Paramètres

*`string`*`:String` - La chaîne à lire et convertir en nombre à virgule flottante.

## Renvoie

`Number` - Un nombre ou `NaN` (n'est pas un nombre).

## Exemple

Les exemples suivants utilisent la fonction `parseFloat()` pour évaluer divers types de nombres :

```
trace(parseFloat("-2")); // output: -2
trace(parseFloat("2.5")); // output: 2.5
trace(parseFloat(" 2.5")); // output: 2.5
trace(parseFloat("3.5e6")); // output: 3500000
trace(parseFloat("foobar")); // output: NaN
trace(parseFloat("3.75math")); // output: 3.75
trace(parseFloat("0garbage")); // output: 0
```

## Voir également

`Constante NaN`, `Fonction parseInt`

# Fonction parseInt

```
parseInt(expression:String, [radix:Number]) : Number
```

Convertit une chaîne en entier. Si la chaîne spécifiée des paramètres ne peut pas être convertie en nombre, la fonction renvoie `NaN`. Les chaînes commençant par 0x sont interprétées en tant que nombres hexadécimaux. Les entiers commençant par 0 ou spécifiant une base 8 sont interprétés en tant que nombres octaux. L'espace blanc qui précède un entier valide est ignoré, comme les caractères de fin non numériques.

**Disponibilité :** Flash Player 5 ; ActionScript 1.0

## Paramètres

*expression*:String - Une chaîne à convertir en entier.

*radix*:Number [facultatif] - Un entier représentant la base du nombre à analyser. Les valeurs valides sont comprises entre 2 et 36.

## Renvoie

Number - Un nombre ou NaN (n'est pas un nombre).

## Exemple

Les exemples de cette section utilisent la fonction parseInt() pour évaluer divers types de nombres.

L'exemple suivant renvoie 3 :

```
parseInt("3.5")
```

L'exemple suivant renvoie NaN :

```
parseInt("bar")
```

L'exemple suivant renvoie 4 :

```
parseInt("4foo")
```

L'exemple suivant illustre une conversion hexadécimale qui renvoie 1016 :

```
parseInt("0x3F8")
```

L'exemple suivant illustre une conversion hexadécimale utilisant le paramètre *radix* facultatif qui renvoie 1000 :

```
parseInt("3E8", 16)
```

L'exemple suivant illustre une conversion binaire et renvoie 10, soit la représentation décimale du binaire 1010 :

```
parseInt("1010", 2)
```

Les exemples suivants illustrent l'analyse des nombres octaux et renvoient 511, soit la représentation décimale du nombre octal 777 :

```
parseInt("0777")
parseInt("777", 8)
```

## Voir également

, Fonction parseFloat

# Fonction play

```
play() : Void
```

Fait avancer la tête de lecture au sein du scénario.

**Disponibilité :** Flash Player 2 ; ActionScript 1.0

## Exemple

Dans l'exemple suivant, deux occurrences de clip intitulées `stop_mc` et `play_mc` se trouvent sur la scène. Le script ActionScript arrête la lecture du fichier SWF lorsque l'utilisateur clique sur l'occurrence de clip `stop_mc`. La lecture reprend lorsque l'utilisateur clique sur l'occurrence `play_mc`.

```
this.stop_mc.onRelease = function() {
 stop();
};
this.play_mc.onRelease = function() {
 play();
};
trace("frame 1");
```

## Voir également

`Fonction gotoAndPlay`, `gotoAndPlay (méthode MovieClip.gotoAndPlay)`

# Fonction prevFrame

```
prevFrame() : Void
```

Place la tête de lecture sur l'image précédente. Si l'image active est l'image 1, la tête de lecture ne bouge pas.

**Disponibilité :** Flash Player 2 ; ActionScript 1.0

## Exemple

Lorsque l'utilisateur clique sur un bouton intitulé `myBtn_btn` et que le code ActionScript suivant est placé sur une image du scénario correspondant à ce bouton, la tête de lecture est positionnée sur l'image précédente :

```
stop();
this.myBtn_btn.onRelease = function(){
 prevFrame();
};
```

## Voir également

`Fonction nextFrame`, prevFrame (méthode MovieClip.prevFrame)

# Fonction prevScene

```
prevScene() : Void
```

Place la tête de lecture sur l'image 1 de la séquence précédente.

**Disponibilité :** Flash Player 2 ; ActionScript 1.0

### Voir également

`Fonction nextScene`

# Fonction print

```
print(target:Object, boundingBox:String) : Void
```

Imprime le clip `target` en fonction des limites spécifiées par le paramètre (`bmovie`, `bmax` ou `bframe`). Si vous souhaitez imprimer des images spécifiques du clip cible, associez une étiquette `#p` à ces images. Bien que la fonction `print()` produise des impressions de meilleure qualité que la fonction `printAsBitmap()`, elle ne permet pas d'imprimer des clips comportant des transparences alpha ou des effets spéciaux de couleur.

Si vous utilisez `bmovie` pour le paramètre `boundingBox`, mais n'associez pas d'étiquette `#b` à une image, la zone d'impression est déterminée par la taille de la scène du clip chargé. (Le clip chargé n'hérite pas de la taille de la scène du clip principal.)

Tous les éléments imprimables d'un clip doivent avoir été chargés de façon intégrale avant le début de l'impression.

La fonction d'impression de Flash Player prend en charge les imprimantes PostScript et non PostScript. Les imprimantes non PostScript convertissent les vecteurs en bitmaps.

**Disponibilité :** Flash Player 4 ; ActionScript 1.0

### Paramètres

*`target`*`:Object` - Le nom d'occurrence du clip à imprimer. Par défaut, l'ensemble des images de l'occurrence cible peuvent être imprimées. Si vous souhaitez imprimer des images spécifiques du clip, associez une étiquette `#p` à ces images.

*`boundingBox`*`:String` - Un modificateur qui définit la zone d'impression du clip. Placez ce paramètre entre guillemets simples ou doubles (' ou ") et spécifiez l'une des valeurs suivantes :

- `bmovie` Désigne le cadre de délimitation d'une image spécifique dans un clip en tant que zone d'impression pour l'ensemble des images imprimables du clip. Associez une étiquette d'image `#b` à l'image dont vous souhaitez utiliser le cadre de délimitation en tant que zone imprimable.

- `bmax` Désigne une combinaison de l'ensemble des cadres de délimitation de l'ensemble des images imprimables en tant que zone d'impression. Spécifiez `bmax` si la taille des images imprimables de votre clip varie.
- `bframe` Indique que le cadre de délimitation de chaque image imprimable doit être utilisé en tant que zone d'impression pour cette image, ce qui change la zone d'impression de chaque image et met les objets à l'échelle pour les adapter à la zone d'impression. Utilisez `bframe` si vous avez des objets de différentes tailles dans chaque image et souhaitez que chaque objet remplisse la page imprimée.

### Exemple

L'exemple suivant imprime toutes les images imprimables dans `holder_mc` en appliquant une zone d'impression définie par le cadre de délimitation de chaque image :

```
this.createEmptyMovieClip("holder_mc", 999);
holder_mc.loadMovie("http://www.helpexamples.com/flash/images/image1.jpg");

this.myBtn_btn.onRelease = function() {
 print(this._parent.holder_mc, "bframe");
};
```

Dans le script ActionScript précédent, vous pouviez remplacer `bframe` par `bmovie` de manière à ce que la zone d'impression soit définie par le cadre de délimitation d'une image à laquelle est associée l'étiquette `#b`.

### Voir également

`Fonction printAsBitmap`, `Fonction printAsBitmapNum`, `PrintJob`, `Fonction printNum`

## Fonction printAsBitmap

```
printAsBitmap(target:Object, boundingBox:String) : Void
```

Imprime le clip `target` en tant que bitmap en fonction des limites spécifiées par le paramètre (`bmovie`, `bmax` ou `bframe`). Utilisez `printAsBitmap()` pour imprimer des clips qui contiennent des images avec des objets qui appliquent des effets de transparence ou de couleur. L'action `printAsBitmap()` imprime à la résolution la plus élevée disponible, de façon à préserver la définition et la qualité autant que faire se peut.

Si votre clip ne contient pas de transparences alpha ou d'effets de couleur, Macromedia recommande d'utiliser la fonction `print()` pour de meilleurs résultats qualitatifs.

Si vous utilisez `bmovie` pour le paramètre `boundingBox`, mais n'associez pas d'étiquette `#b` à une image, la zone d'impression est déterminée par la taille de la scène du clip chargé. (Le clip chargé n'hérite pas de la taille de la scène du clip principal.)

Tous les éléments imprimables d'un clip doivent avoir été chargés de façon intégrale avant le début de l'impression.

La fonction d'impression de Flash Player prend en charge les imprimantes PostScript et non PostScript. Les imprimantes non PostScript convertissent les vecteurs en bitmaps.

**Disponibilité :** Flash Player 4 ; ActionScript 1.0

## Paramètres

*target*:Object - Le nom d'occurrence du clip à imprimer. Par défaut, l'ensemble des images du clip sont imprimées. Si vous souhaitez imprimer des images spécifiques du clip, associez une étiquette #p à ces images.

*boundingBox*:String - Un modificateur qui définit la zone d'impression du clip. Placez ce paramètre entre guillemets simples ou doubles (' ou ") et spécifiez l'une des valeurs suivantes :

- bmovie Désigne le cadre de délimitation d'une image spécifique dans un clip en tant que zone d'impression pour l'ensemble des images imprimables du clip. Associez une étiquette d'image #b à l'image dont vous souhaitez utiliser le cadre de délimitation en tant que zone imprimable.
- bmax Désigne une combinaison de l'ensemble des cadres de délimitation de l'ensemble des images imprimables en tant que zone d'impression. Spécifiez le paramètre bmax lorsque la taille des images imprimables de votre clip varie.
- bframe Indique que le cadre délimitation de chaque image imprimable doit servir de zone d'impression pour cette image. Ceci change la zone d'impression de chaque image et met l'objet à l'échelle de la zone d'impression. Utilisez bframe si vous avez des objets de différentes tailles dans chaque image et souhaitez que chaque objet remplisse la page imprimée.

## Exemple

L'exemple suivant imprime toutes les images imprimables dans holder_mc en appliquant une zone d'impression définie par le cadre de délimitation de l'image :

```
this.createEmptyMovieClip("holder_mc", 999);
holder_mc.loadMovie("http://www.helpexamples.com/flash/images/image1.jpg");

this.myBtn_btn.onRelease = function() {
 printAsBitmap(this._parent.holder_mc, "bframe");
};
```

## Voir également

Fonction print, Fonction printAsBitmapNum, Fonction printNum, PrintJob

# Fonction printAsBitmapNum

```
printAsBitmapNum(level:Number, boundingBox:String) : Void
```

Imprime un niveau dans Flash Player en tant que bitmap en fonction des limites spécifiées par le paramètre (`bmovie`, `bmax` ou `bframe`). Utilisez `printAsBitmapNum()` pour imprimer des clips qui contiennent des images avec des objets qui appliquent des effets de transparence ou de couleur. L'action `printAsBitmapNum()` imprime à la résolution la plus élevée disponible, de façon à préserver la plus haute définition possible et la qualité autant que faire se peut. Pour calculer la taille du fichier imprimable d'une image à imprimer au format bitmap, multipliez la largeur en pixels par la hauteur en pixels par la résolution de l'imprimante.

Si votre clip ne contient pas de transparences alpha ou d'effets de couleur, utilisez `printNum()` pour de meilleurs résultats qualitatifs.

Si vous utilisez `bmovie` pour le paramètre `boundingBox`, mais n'associez pas d'étiquette `#b` à une image, la zone d'impression est déterminée par la taille de la scène du clip chargé. (Le clip chargé n'hérite pas de la taille de la scène de l'animation principale.)

Tous les éléments imprimables d'un clip doivent avoir été chargés de façon intégrale avant le début de l'impression.

La fonction d'impression de Flash Player prend en charge les imprimantes PostScript et non PostScript. Les imprimantes non PostScript convertissent les vecteurs en bitmaps.

**Disponibilité :** Flash Player 5 ; ActionScript 1.0

## Paramètres

*`level`*`:Number` - Le niveau de Flash Player à imprimer. Par défaut, l'ensemble des images du niveau sont imprimées. Si vous souhaitez imprimer des images spécifiques du niveau, associez une étiquette `#p` à ces images.

*`boundingBox`*`:String` - Un modificateur qui définit la zone d'impression du clip. Placez ce paramètre entre guillemets simples ou doubles (' ou ") et spécifiez l'une des valeurs suivantes :

- `bmovie` Désigne le cadre de délimitation d'une image spécifique dans un clip en tant que zone d'impression pour l'ensemble des images imprimables du clip. Associez une étiquette d'image `#b` à l'image dont vous souhaitez utiliser le cadre de délimitation en tant que zone imprimable.
- `bmax` Désigne une combinaison de l'ensemble des cadres de délimitation de l'ensemble des images imprimables en tant que zone d'impression. Spécifiez le paramètre `bmax` lorsque la taille des images imprimables de votre clip varie.

- `bframe` Indique que le cadre délimitation de chaque image imprimable doit servir de zone d'impression pour cette image. Ceci change la zone d'impression de chaque image et met l'objet à l'échelle de la zone d'impression. Utilisez `bframe` si vous avez des objets de différentes tailles dans chaque image et souhaitez que chaque objet remplisse la page imprimée.

### Exemple

L'exemple suivant imprime le contenu de la scène lorsque l'utilisateur clique sur le bouton `myBtn_btn`. La zone à imprimer est définie par le cadre de délimitation de l'image.

```
myBtn_btn.onRelease = function(){
 printAsBitmapNum(0, "bframe")
};
```

### Voir également

`Fonction print`, `Fonction printAsBitmap`, `PrintJob`, `Fonction printNum`

## Fonction printNum

```
printNum(level:Number, boundingBox:String) : Void
```

Imprime le niveau dans Flash Player en fonction des limites spécifiées par le paramètre `boundingBox` (`bmovie`, `bmax`, `bframe`). Si vous souhaitez imprimer des images spécifiques du clip cible, associez une étiquette `#p` à ces images. Bien que `printNum()` permette de bénéficier d'une meilleure qualité que `printAsBitmapNum()`, vous ne pouvez pas utiliser `printNum()` pour imprimer des animations comportant des transparences alpha ou des effets spéciaux de couleur.

Si vous utilisez `bmovie` pour le paramètre `boundingBox`, mais n'associez pas d'étiquette `#b` à une image, la zone d'impression est déterminée par la taille de la scène du clip chargé. (Le clip chargé n'hérite pas de la taille de la scène de l'animation principale.)

Tous les éléments imprimables d'un clip doivent avoir été chargés de façon intégrale avant le début de l'impression.

La fonction d'impression de Flash Player prend en charge les imprimantes PostScript et non PostScript. Les imprimantes non PostScript convertissent les vecteurs en bitmaps.

**Disponibilité :** Flash Player 5 ; ActionScript 1.0

### Paramètres

*`level`*`:Number` - Le niveau de Flash Player à imprimer. Par défaut, l'ensemble des images du niveau sont imprimées. Si vous souhaitez imprimer des images spécifiques du niveau, associez une étiquette `#p` à ces images.

*boundingBox*:String - Un modificateur qui définit la zone d'impression du clip. Placez ce paramètre entre guillemets simples ou doubles (' ou ") et spécifiez l'une des valeurs suivantes :

- bmovie Désigne le cadre de délimitation d'une image spécifique dans un clip en tant que zone d'impression pour l'ensemble des images imprimables du clip. Associez une étiquette d'image #b à l'image dont vous souhaitez utiliser le cadre de délimitation en tant que zone imprimable.
- bmax Désigne une combinaison de l'ensemble des cadres de délimitation de l'ensemble des images imprimables en tant que zone d'impression. Spécifiez le paramètre bmax lorsque la taille des images imprimables de votre clip varie.
- bframe Indique que le cadre délimitation de chaque image imprimable doit servir de zone d'impression pour cette image. Ceci change la zone d'impression de chaque image et met l'objet à l'échelle de la zone d'impression. Utilisez bframe si vous avez des objets de différentes tailles dans chaque image et souhaitez que chaque objet remplisse la page imprimée.

### Voir également

Fonction print, Fonction printAsBitmap, Fonction printAsBitmapNum, PrintJob

## Fonction random

```
random(value:Number) : Number
```

*Déconseillée* à partir de Flash Player 5. Il est recommandé d'utiliser la méthode Math.random().

Renvoie un entier aléatoire compris entre 0 et un inférieur au nombre entier spécifié dans le paramètre *value*.

**Disponibilité :** Flash Player 4 ; ActionScript 1.0

### Paramètres

*value*:Number - Un entier.

### Renvoie

Number - Un entier aléatoire.

### Exemple

L'utilisation suivante de la fonction random() renvoie une valeur de 0, 1, 2, 3 ou 4 :

```
random(5);
```

### Voir également

random (méthode Math.random)

## Fonction removeMovieClip

removeMovieClip(target:Object)

Supprime le clip spécifié.

**Disponibilité :** Flash Player 4 ; ActionScript 1.0

### Paramètres

*target*:Object - Le chemin cible d'une occurrence de clip créée avec duplicateMovieClip() ou le nom d'occurrence d'un clip créé avec MovieClip.attachMovie(), MovieClip.duplicateMovieClip(), ou MovieClip.createEmptyMovieClip().

### Exemple

L'exemple suivant crée un nouveau clip intitulé myClip_mc et le duplique. Le second clip est appelé newClip_mc. Les images sont chargées dans les deux clips. Lorsque l'utilisateur clique sur un bouton, button_mc, le clip dupliqué est retiré de la scène.

```
this.createEmptyMovieClip("myClip_mc", this.getNextHighestDepth());
myClip_mc.loadMovie("http://www.helpexamples.com/flash/images/image1.jpg");
duplicateMovieClip(this.myClip_mc, "newClip_mc",
  this.getNextHighestDepth());
newClip_mc.loadMovie("http://www.helpexamples.com/flash/images/
  image1.jpg");
newClip_mc._x = 200;
this.button_mc.onRelease = function() {
 removeMovieClip(this._parent.newClip_mc);
};
```

### Voir également

Fonction duplicateMovieClip, duplicateMovieClip (méthode MovieClip.duplicateMovieClip), attachMovie (méthode MovieClip.attachMovie), removeMovieClip (méthode MovieClip.removeMovieClip), createEmptyMovieClip (méthode MovieClip.createEmptyMovieClip)

# Fonction setInterval

`setInterval(`*functionReference:Function*, *interval:Number*, [*param1:Object*, *param2*, ..., *paramN*]`)` *: Number*
`setInterval(`*objectReference:Object, methodName:String*, *interval:Number*, [*param1:Object*, *param2*, ..., *paramN*]`)` *: Number*

Appelle une fonction ou une méthode d'un objet à des intervalles périodiques pendant la lecture d'un fichier SWF. Vous pouvez utiliser `setInterval()` pour exécuter une fonction de manière répétitive dans le temps.

Utilisez les conseils suivants lorsque vous utilisez `setInterval()` :

- Identifiez le domaine de la fonction appelée.
- Identifiez le domaine dans lequel l'ID d'intervalle (la valeur renvoyée de `setInterval()`) a été défini.
- Supprimez les intervalles définis avant d'en déterminer de nouveaux.

Ces conseils sont décrits de façon plus détaillée dans les paragraphes qui suivent.

*Identifiez le domaine de la fonction appelée.* Pour identifier le domaine de la fonction appelée, définissez l'objet sur lequel la méthode `setInterval()` peut s'exécuter (le domaine de l'objet) comme étant le premier paramètre et le nom de la méthode que vous souhaitez exécuter comme étant le deuxième paramètre (comme indiqué dans la deuxième signature). Ceci permet de s'assurer que la méthode voulue s'exécute à partir du domaine de la référence d'objet transmise. Lorsque cette méthode est ainsi exécutée, elle permet de faire référence à des variables de membre de l'objet utilisant le mot-clé `this`.

*Identifiez le domaine dans lequel l'identificateur d'intervalle a été défini.* Pour identifier le domaine dans lequel l'identificateur d'intervalle (`intervalId`) a été défini, vous pouvez l'affecter à une variable de membre du domaine de l'objet transmis à `setInterval()`. La fonction appelée peut ainsi localiser l'identificateur d'intervalle au niveau de `this.intervalId`.

*Supprimez les intervalles précédemment définis.* Pour supprimer les intervalles définis avant d'en déterminer de nouveaux, vous devez généralement appeler `clearInterval()` *avant* `setInterval()`. Ceci vous évite d'écraser ou de détruire votre variable `intervalId`, la seule référence à l'intervalle en cours d'exécution. Pour appeler `clearInterval()` avant `setInterval()`, le script d'initialisation et le script exécuté doivent avoir accès à `intervalId`, comme indiqué dans les exemples.

*Remarque :* Veillez à appeler `clearInterval()` pour arrêter la lecture en boucle du script.

**Disponibilité :** Flash Player 6 ; ActionScript 1.0

## Paramètres

*functionReference*:Function - Une référence à la fonction à appeler.

*interval*:Number - Le nombre de millisecondes séparant les appels de la fonction functionReference ou methodName transmise.

Si la valeur d'interval est inférieure à la cadence d'images du fichier SWF (par exemple, 10 images par seconde [fps] correspond à des intervalles de 100 millisecondes), la fonction d'intervalle est appelée aussi près que possible de la valeur interval. L'exécution de scripts longs, utilisant beaucoup de mémoire, au cours d'un intervalle entraîne des retards. Si la fonction appelée modifie les éléments visuels, vous devez utiliser la fonction updateAfterEvent() afin de vous assurer que l'écran est régulièrement actualisé. Si la valeur interval est supérieure à la cadence d'images du fichier SWF, la fonction interval n'est appelée que lorsque interval a expiré *et que* la tête de lecture passe sur l'image suivante, ce qui réduit l'impact à chaque fois que l'écran est actualisé.

*param*:Object [facultatif] - Paramètres transmis à la fonction envoyée à functionReference ou methodName. Les paramètres multiples doivent être séparés par des virgules : *param1* , *param2* , ..., *paramN*

*objectReference*:Object - Un objet contenant la méthode spécifiée par methodName.

*methodName*:String - Une méthode qui existe dans le domaine de l'objet spécifié par objectReference.

## Renvoie

Number - Un entier qui identifie l'intervalle (l'ID d'intervalle) que vous pouvez transmettre à clearInterval() pour annuler l'intervalle.

## Exemple

Exemple 1 : L'exemple suivant trace un message selon un intervalle de 20 millisecondes, jusqu'à 10 fois, puis supprime l'intervalle. Le domaine de l'objet, this, est transmis comme étant le premier paramètre ; le nom de la méthode, executeCallback, comme étant le deuxième. Ceci permet de s'assurer que executeCallback() s'exécute à partir du même domaine que celui du script effectuant l'appel.

```
var intervalId:Number;
var count:Number = 0;
var maxCount:Number = 10;
var duration:Number = 20;

function executeCallback():Void {
 trace("executeCallback intervalId: " + intervalId + " count: " + count);
 if(count >= maxCount) {
```

```
 clearInterval(intervalId);
 }
 count++;
}

intervalId = setInterval(this, "executeCallback", duration);
```

Exemple 2 : L'exemple suivant est similaire au premier, à la différence qu'il appelle `clearInterval()` avant `setInterval()`. Cette méthode peut empêcher l'obtention de boucles non souhaitées et s'avère particulièrement importante au sein des systèmes basés sur des événements, dans lesquels le script d'initialisation peut être exécuté à plusieurs reprises avant la suppression d'un intervalle spécifique.

```
var intervalId:Number;
var count:Number = 0;
var maxCount:Number = 10;
var duration:Number = 20;

function executeCallback():Void {
 trace("executeCallback intervalId: " + intervalId + " count: " + count);
 if(count >= maxCount) {
 clearInterval(intervalId);
 }
 count++;
}

function beginInterval():Void {
 if(intervalId != null) {
 trace("clearInterval");
 clearInterval(intervalId);
 }
 intervalId = setInterval(this, "executeCallback", duration);
}

beginInterval();
beginInterval();
beginInterval();
```

Exemple 3 : L'exemple suivant indique comment transmettre un argument personnalisé à la fonction appelée.

```
var intervalId:Number;
var count:Number = 0;
var maxCount:Number = 10;
var duration:Number = 20;
var colors:Array = new Array("red",
 "blue",
```

```
 "yellow",
 "purple",
 "green",
 "orange",
 "salmon",
 "pink",
 "lilac",
 "powder blue",
 "mint");

function executeCallback(param:String) {
 trace("executeCallback intervalId: " + intervalId + " count: " + count + "
  param: " + param);
 clearInterval(intervalId);
 if(count < maxCount) {
 count++;
 intervalId = setInterval(this, "executeCallback", duration,
  colors[count]);
 }
}

if(intervalId != null) {
 clearInterval(intervalId);
}

intervalId = setInterval(this, "executeCallback", duration, colors[count]);
```

Exemple 4 : L'exemple suivant indique comment utiliser correctement `setInterval()` à partir d'une classe ActionScript 2.0 personnalisée. Remarquez que, comme dans les exemples précédents, le paramètre `this` est transmis à la fonction `setInterval()` afin de s'assurer que la méthode appelée s'exécute dans le domaine approprié.

```
class CustomClass {
 private var intervalId:Number;
 private var count:Number = 0;
 private var maxCount:Number = 10;
 private var duration:Number = 20;

 public function CustomClass():Void {
 beginInterval();
 }

 private function beginInterval():Void {
 if(intervalId != null) {
 trace("clearInterval");
 clearInterval(intervalId);
 }
 intervalId = setInterval(this, "executeCallback", duration);
```

```
 }

 public function executeCallback():Void {
 trace("executeCallback intervalId: " + intervalId + " count: " + count);
 if(count >= maxCount) {
 clearInterval(intervalId);
 }
 count++;
 }
}
```

Dans un nouveau document, instanciez une nouvelle occurrence de la nouvelle classe :

```
var custom:CustomClass = new CustomClass();
```

### Voir également

`Fonction clearInterval`, `Fonction updateAfterEvent`, `Instruction class`

## Fonction setProperty

```
setProperty(target:Object, property:Object, expression:Object) : Void
```

Modifie la valeur des propriétés d'un clip pendant la lecture de ce dernier.

**Disponibilité :** Flash Player 4 ; ActionScript 1.0

### Paramètres

*`target`*`:Object` - Le chemin du nom d'occurrence du clip dont la propriété doit être définie.

*`property`*`:Object` - La propriété à définir.

*`expression`*`:Object` - Soit la nouvelle valeur littérale de la propriété, soit une équation qui reprend la nouvelle valeur de la propriété.

### Exemple

Le code ActionScript suivant crée un nouveau clip et charge une image dans celui-ci. Les coordonnées _xet _y sont définies pour le clip à l'aide de `setProperty()`. Lorsque vous cliquez sur le bouton intitulé `right_btn`, la coordonnée _x d'un clip nommé `params_mc` est incrémentée de 20 pixels.

```
this.createEmptyMovieClip("params_mc", 999);
params_mc.loadMovie("http://www.helpexamples.com/flash/images/image1.jpg");
setProperty(this.params_mc, _y, 20);
setProperty(this.params_mc, _x, 20);
this.right_btn.onRelease = function() {
```

```
 setProperty(params_mc, _x, getProperty(params_mc, _x)+20);
};
```

### Voir également

`Fonction getProperty`

## Fonction showRedrawRegions

```
showRedrawRegions(enable:Boolean, [color:Number]) : Void
```

Permet au débogueur de délimiter les zones redessinées de l'écran (c'est-à-dire les zones sales mises à jour). Les contours peuvent également être activés via l'option de menu Redessiner les régions.

**Disponibilité :** Flash Player 8 ; ActionScript 1.0

### Paramètres

*`enable`*`:Boolean` - Indique si Redessiner les régions doit être activé (`true`) ou désactivé (`false`). Lorsque l'option est définie sur `true`, les rectangles redessinés s'affichent. Lorsque l'option est définie sur `false`, les rectangles redessinés sont supprimés.

*`color`*`:Number` [facultatif] - La couleur utilisée pour dessiner. La valeur par défaut est rouge : 0xFF0000.

### Exemple

L'exemple suivant illustre la fonction `showRedrawRegions`.

```
var w:Number = 100;
var h:Number = 100;

var shape1:MovieClip = createShape("shape1");
shape1.onEnterFrame = function():Void {
 this._x += 5;
 this._y += 5;
}

var shape2:MovieClip = createShape("shape2");
shape2.onEnterFrame = function():Void {
 this._y += 5;
}

_global.showRedrawRegions(true);

function createShape(name:String):MovieClip {
 var mc:MovieClip = this.createEmptyMovieClip(name,
  this.getNextHighestDepth());
```

```
 mc.beginFill(0xFFCC00);
 mc.moveTo(200, 200);
 mc.curveTo(300, 200, 300, 100);
 mc.curveTo(300, 0, 200, 0);
 mc.curveTo(100, 0, 100, 100);
 mc.curveTo(100, 200, 200, 200);
 mc.endFill();
 return mc;
}
```

# Fonction startDrag

```
startDrag(target:Object, [lock:Boolean, left:Number, top:Number,
right:Number, bottom:Number]) : Void
```

Rend le clip *target* déplaçable pendant la lecture de l'animation. Vous ne pouvez déplacer qu'un seul clip à la fois. Après l'exécution d'une opération `startDrag()`, le clip reste déplaçable jusqu'à ce qu'il soit arrêté de façon explicite par `stopDrag()` ou jusqu'à ce qu'une action `startDrag()` soit appelée pour un autre clip.

**Disponibilité :** Flash Player 4 ; ActionScript 1.0

## Paramètres

*target*:`Object` - Le chemin cible du clip à déplacer.

*lock*:`Boolean` [facultatif] - Une valeur booléenne spécifiant si le clip à déplacer doit être verrouillé au centre de la position de la souris (`true`) ou verrouillé au point où l'utilisateur a cliqué sur le clip en premier lieu (`false`).

*left*,*top*,*right*,*bottom*:`Number` [facultatif] - Valeurs relatives aux coordonnées du parent du clip qui spécifient un rectangle de délimitation pour le clip.

## Exemple

L'exemple suivant crée, à l'exécution, un clip `pic_mc` que les utilisateurs peuvent faire glisser vers l'emplacement voulu en y associant les actions `startDrag()` et `stopDrag()`. Une image est chargée dans `pic_mc` à l'aide de la classe MovieClipLoader.

```
var pic_mcl:MovieClipLoader = new MovieClipLoader();
pic_mcl.loadClip("http://www.helpexamples.com/flash/images/image1.jpg",
 this.createEmptyMovieClip("pic_mc", this.getNextHighestDepth()));
var listenerObject:Object = new Object();
listenerObject.onLoadInit = function(target_mc) {
 target_mc.onPress = function() {
 startDrag(this);
 };
 target_mc.onRelease = function() {
 stopDrag();
```

```
 };
};
pic_mcl.addListener(listenerObject);
```

### Voir également

Fonction stopDrag, _droptarget (propriété MovieClip._droptarget), startDrag (méthode MovieClip.startDrag)

## Fonction stop

```
stop() : Void
```

Arrête le fichier SWF en cours de lecture. Cette fonction sert généralement à contrôler les clips avec des boutons.

**Disponibilité :** Flash Player 2 ; ActionScript 1.0

### Voir également

Fonction gotoAndStop, gotoAndStop (méthode MovieClip.gotoAndStop)

## Fonction stopAllSounds

```
stopAllSounds() : Void
```

Arrête tous les sons en cours de diffusion à partir d'un fichier SWF, sans arrêter la tête de lecture. Les sons diffusés en continu sont émis de nouveau lorsque la tête de lecture passe au-dessus des images contenant ces sons.

**Disponibilité :** Flash Player 3 ; ActionScript 1.0

### Exemple

Le code suivant crée un champ de texte dans lequel s'affichent les informations ID3 de la chanson. Une nouvelle occurrence de l'objet Sound est créée et votre fichier MP3 est chargé dans le fichier SWF. Les informations ID3 sont extraites du fichier audio. Lorsque l'utilisateur clique sur `stop_mc`, , le son s'interrompt. Lorsque l'utilisateur clique sur `play_mc`, , la chanson reprend à partir de la position à laquelle elle a été interrompue.

```
this.createTextField("songinfo_txt", this.getNextHighestDepth, 0, 0,
  Stage.width, 22);
var bg_sound:Sound = new Sound();
bg_sound.loadSound("yourSong.mp3", true);
bg_sound.onID3 = function() {
 songinfo_txt.text = "(" + this.id3.artist + ") " + this.id3.album + " - " +
  this.id3.track + " - "
 + this.id3.songname;
```

```
 for (prop in this.id3) {
 trace(prop+" = "+this.id3[prop]);
 }
 trace("ID3 loaded.");
};
this.play_mc.onRelease = function() {
 /* get the current offset. if you stop all sounds and click the play
  button, the MP3 continues from
 where it was stopped, instead of restarting from the beginning. */
 var numSecondsOffset:Number = (bg_sound.position/1000);
 bg_sound.start(numSecondsOffset);
};
this.stop_mc.onRelease = function() {
 stopAllSounds();
};
```

### Voir également

Sound

# Fonction stopDrag

```
stopDrag() : Void
```

Arrête l'opération de déplacement en cours.

**Disponibilité :** Flash Player 4 ; ActionScript 1.0

### Exemple

Le code suivant, placé dans le scénario principal, arrête le mouvement sur l'occurrence de clip `my_mc` lorsque l'utilisateur relâche le bouton de la souris :

```
my_mc.onPress = function () {
 startDrag(this);
}

my_mc.onRelease = function() {
 stopDrag();
}
```

### Voir également

`Fonction startDrag`, `_droptarget (propriété MovieClip._droptarget)`, `startDrag (méthode MovieClip.startDrag)`, `stopDrag (méthode MovieClip.stopDrag)`

# Fonction String

```
String(expression:Object) : String
```

Renvoie une chaîne représentant le paramètre spécifié, comme indiqué dans la liste suivante :

- Si *expression* est un nombre, la chaîne renvoyée représente le nombre sous forme de texte.
- Si *expression* est une chaîne, la chaîne renvoyée est *expression*.
- Si *expression* est un objet, la valeur renvoyée est une chaîne représentant l'objet généré en appelant la propriété string de l'objet ou en appelant `Object.toString()` en l'absence de ce type de propriété.
- Si *expression* est une valeur booléenne, la chaîne renvoyée est `"true"` ou `"false"`.
- Si *expression* est un clip, la valeur renvoyée est le chemin cible du clip avec la notation à barre oblique (/).

Si *expression* est `undefined`, la fonction renvoie l'une des valeurs suivantes :

- Dans les fichiers publiés pour Flash Player 6 ou version précédente, le résultat est une chaîne vide (" ").
- Dans les fichiers publiés pour Flash Player 7 ou version ultérieure, le résultat est `undefined`.

*Remarque :* La notation avec barre oblique n'est pas prise en charge par ActionScript 2.0.

**Disponibilité :** Flash Player 4 ; ActionScript 1.0

## Paramètres

*expression*`:Object` - Une expression à convertir en chaîne.

## Renvoie

`String` - Une chaîne.

## Exemple

Dans l'exemple suivant, vous utilisez ActionScript pour convertir les expressions spécifiées en chaîne :

```
var string1:String = String("3");
var string2:String = String("9");
trace(string1+string2); // output: 39
```

Etant donné que les deux paramètres sont des chaînes, les valeurs sont concaténées au lieu d'être ajoutées.

### Voir également

toString (méthode Number.toString), toString (méthode Object.toString), String, Opérateur " (séparateur de chaîne)

## Fonction substring

```
substring(string:String, index:Number, count:Number) : String
```

*Déconseillée* à partir de Flash Player 5. Il est recommandé d'utiliser la méthode `String.substr()`.

Extrait une partie d'une chaîne. Cette fonction est de base un tandis que les méthodes de l'objet String sont de base zéro.

**Disponibilité :** Flash Player 4 ; ActionScript 1.0

### Paramètres

*`string`*`:String` - La chaîne à partir de laquelle il convient d'extraire la nouvelle chaîne.

*`index`*`:Number` - Le numéro du premier caractère à extraire.

*`count`*`:Number` - Le nombre de caractères à inclure dans la chaîne extraite, caractère d'indice non compris.

### Renvoie

`String` - La sous-chaîne extraite.

### Voir également

substr (méthode String.substr)

## Fonction targetPath

```
targetpath(targetObject:Object) : String
```

Renvoie une chaîne contenant le chemin cible d'un objet MovieClip, Button, TextField ou Video. Le chemin cible est renvoyé sous forme de notation par point (.). Pour extraire le chemin cible sous forme de notation à barre oblique (/), utilisez la propriété `_target`.

**Disponibilité :** Flash Player 5 ; ActionScript 1.0

### Paramètres

*`targetObject`*`:Object` - Référence (par exemple, `_root` ou `_parent`) à l'objet pour lequel le chemin cible est extrait. Il peut s'agir d'un objet MovieClip, Button ou TextField.

### Renvoie

`String` - Une chaîne contenant le chemin cible de l'objet spécifié.

### Exemple

L'exemple suivant présente le chemin cible d'un clip dès la fin de son chargement :

```
this.createEmptyMovieClip("myClip_mc", this.getNextHighestDepth());
trace(targetPath(myClip_mc)); // _level0.myClip_mc
```

### Voir également

`Fonction eval`

## Fonction tellTarget

```
tellTarget(target:String) {
statement(s);
}
```

*Déconseillée* à partir de Flash Player 5. Macromedia vous recommande d'utiliser une notation de type point (.) et l'instruction `with`.

Applique les instructions spécifiées dans le paramètre *`statements`* au scénario spécifié dans le paramètre *`target`*. L'action `tellTarget` est particulièrement utile pour les contrôles de navigation. Affectez la fonction `tellTarget` aux boutons qui permettent d'arrêter ou de démarrer les clips ailleurs sur la scène. Vous pouvez également contraindre les clips à accéder à une image spécifique dans ce clip. Par exemple, vous pouvez affecter la fonction `tellTarget` aux boutons qui permettent d'arrêter ou de démarrer les clips sur la scène ou inviter les clips à atteindre une image spécifique.

Dans Flash 5 ou version ultérieure, vous pouvez utiliser une notation de type point (.) au lieu de l'action `tellTarget`. Vous pouvez utiliser l'action `with` pour publier plusieurs actions dans le même scénario. Vous pouvez utiliser l'action `with` pour cibler l'objet de votre choix, tandis que l'action `tellTarget` peut uniquement cibler les clips.

**Disponibilité :** Flash Player 3 ; ActionScript 1.0

### Paramètres

*`target`*`:String` - Une chaîne qui spécifie le chemin cible du scénario à contrôler.

*`statement(s)`* - Les instructions à exécuter lorsque la condition est `true`.

### Exemple

Cette instruction tellTarget contrôle la balle de l'occurrence de clip sur le scénario principal. L'image 1 de l'occurrence balle est vide et est associée à une action stop() : elle n'est donc pas visible sur la scène. Lorsque vous cliquez sur le bouton permettant d'effectuer l'action suivante, tellTarget indique à la tête de lecture de la balle d'atteindre l'image 2, où l'animation démarre :

```
on(release) {
tellTarget("_parent.ball") {
gotoAndPlay(2);
}
}
```

L'exemple suivant utilise une notation de type point (.) pour obtenir les mêmes résultats :

```
on(release) {
_parent.ball.gotoAndPlay(2);
}
```

Si vous devez émettre plusieurs commandes sur l'occurrence de balle, vous pouvez utiliser l'action with, comme indiqué dans l'instruction suivante :

```
on(release) {
with(_parent.ball) {
gotoAndPlay(2);
_alpha = 15;
_xscale = 50;
_yscale = 50;
}
}
```

### Voir également

Instruction with

## Fonction toggleHighQuality

```
toggleHighQuality()
```

*Déconseillée* à partir de Flash Player 5. Il est recommandé d'utiliser la méthode `_quality`.

Active et désactive l'anticrénelage dans Flash Player. L'anticrénelage adoucit les bords des objets et ralentit la lecture du fichier SWF. Cette action affecte tous les fichiers SWF dans Flash Player.

**Disponibilité :** Flash Player 2 ; ActionScript 1.0

### Exemple

Le code suivant peut être appliqué à un bouton qui permet d'activer et de désactiver l'anticrénelage lorsque l'utilisateur clique dessus :

```
on(release) {
toggleHighQuality();
}
```

### Voir également

, _quality, propriété

## Fonction trace

```
trace(expression:Object)
```

Vous pouvez utiliser Flash Debug Player pour capturer les sorties de la fonction `trace()` et afficher le résultat.

Cette instruction permet d'écrire des notes de programmation ou d'afficher des messages dans le panneau Sortie pendant le test d'un fichier SWF. Utilisez le paramètre *expression* pour vérifier l'existence d'une condition ou pour afficher des valeurs dans le panneau de sortie. L'instruction `trace()` est similaire à la fonction `alert` de JavaScript.

Vous pouvez également utiliser la commande Omettre les actions Trace de la boîte de dialogue Paramètres de publication pour supprimer les actions `trace()` du fichier SWF exporté.

**Disponibilité :** Flash Player 4 ; ActionScript 1.0

### Paramètres

*expression*:Object - Une expression à évaluer. Lorsqu'un fichier SWF s'exécute dans l'outil de programmation Flash (avec la commande Tester l'animation), la valeur du paramètre *expression* s'affiche dans le panneau de sortie.

### Exemple

L'exemple suivant utilise une instruction `trace()` pour afficher dans le panneau de sortie les méthodes et propriétés du champ de texte intitulé `error_txt` créé de manière dynamique :

```
this.createTextField("error_txt", this.getNextHighestDepth(), 0, 0, 100,
  22);
for (var i in error_txt) {
trace("error_txt."+i+" = "+error_txt[i]);
}
/* output:
error_txt.styleSheet = undefined
error_txt.mouseWheelEnabled = true
```

```
error_txt.condenseWhite = false
...
error_txt.maxscroll = 1
error_txt.scroll = 1
*/
```

# Fonction unescape

```
unescape(string:String) : String
```

Evalue le paramètre *x* en tant que chaîne, décode la chaîne qui est au format codé en URL (en convertissant toutes les séquences hexadécimales en caractères ASCII) et renvoie la chaîne.

**Disponibilité :** Flash Player 5 ; ActionScript 1.0

## Paramètres

*string*:String - Une chaîne comportant des séquences d'échappement hexadécimales.

## Renvoie

String - Une chaîne décodée à partir d'un paramètre codé au format URL.

## Exemple

L'exemple suivant illustre le processus de conversion escape/unescape :

```
var email:String = "user@somedomain.com";
trace(email);
var escapedEmail:String = escape(email);
trace(escapedEmail);
var unescapedEmail:String = unescape(escapedEmail);
trace(unescapedEmail);
```

Le résultat suivant s'affiche dans le panneau de sortie.

```
user@somedomain.com
user%40somedomain%2Ecom
user@somedomain.com
```

# Fonction unloadMovie

```
unloadMovie(target:MovieClip) : Void
unloadMovie(target:String) : Void
```

Supprime le clip qui a été chargé par l'intermédiaire de la fonction `loadMovie()` de Flash Player. Pour décharger un clip chargé avec `loadMovieNum()`, utilisez `unloadMovieNum()` au lieu de `unloadMovie()`.

**Disponibilité :** Flash Player 3 ; ActionScript 1.0

### Paramètres

*target*:Object - Le chemin cible d'un clip. Ce paramètre peut être de type String (tel que « my_mc » ) ou une référence directe à l'occurrence de clip (par exemple my_mc). Les paramètres qui peuvent accepter plusieurs types de données sont répertoriés sous le type Object.

### Exemple

L'exemple suivant crée un nouveau clip intitulé pic_mc et charge une image dans celui-ci. Elle est chargée à l'aide de la classe MovieClipLoader. Lorsque vous cliquez sur l'image, le clip est déchargé du fichier SWF :

```
var pic_mcl:MovieClipLoader = new MovieClipLoader();
pic_mcl.loadClip("http://www.helpexamples.com/flash/images/image1.jpg",
 this.createEmptyMovieClip("pic_mc", this.getNextHighestDepth()));
var listenerObject:Object = new Object();
listenerObject.onLoadInit = function(target_mc) {
 target_mc.onRelease = function() {
 unloadMovie(pic_mc);
 /* or you could use the following, which refers to the movie clip
  referenced by 'target_mc'. */
 //unloadMovie(this);
 };
};
pic_mcl.addListener(listenerObject);
```

### Voir également

loadMovie (méthode MovieClip.loadMovie), unloadClip (méthode MovieClipLoader.unloadClip)

## Fonction unloadMovieNum

unloadMovieNum(level:Number) : Void

Supprime un fichier SWF ou une image qui a été chargée par l'intermédiaire de la fonction loadMovieNum() de Flash Player. Pour décharger un fichier SWF ou une image chargée avec MovieClip.loadMovie(), utilisez unloadMovie() au lieu de unloadMovieNum().

**Disponibilité :** Flash Player 3 ; ActionScript 1.0

### Paramètres

*level*:Number - Le niveau (_level *N*) d'une animation chargée.

### Exemple

L'exemple suivant charge une image dans un fichier SWF. Lorsque vous cliquez sur `unload_btn`, le contenu chargé est supprimé.

```
loadMovieNum("yourimage.jpg", 1);
unload_btn.onRelease = function() {
 unloadMovieNum(1);
}
```

### Voir également

`Fonction loadMovieNum`, `Fonction unloadMovie`, `loadMovie (méthode MovieClip.loadMovie)`

## Fonction updateAfterEvent

```
updateAfterEvent() : Void
```

Met à jour l'affichage (indépendamment du nombre d'images par seconde défini pour l'animation) lorsque vous l'appelez à partir d'un gestionnaire `onClipEvent()` ou dans le cadre d'une fonction ou d'une méthode que vous transmettez à setInterval(). Flash ignore les appels de mise à jour de la fonction `updateAfterEvent` qui ne font pas partie d'un gestionnaire `onClipEvent()` ou bien d'une fonction ou d'une méthode transmise à setInterval(). Cette fonction est uniquement compatible avec certains gestionnaires Mouse et MovieClip : les gestionnaires mouseDown, mouseUp, mouseMove, keyDown et keyUp de la classe Mouse ; les gestionnaires onMouseMove, onMouseDown, onMouseUp, onKeyDown et onKeyUp de la classe MovieClip. Elle n'est pas compatible avec la classe Key.

**Disponibilité :** Flash Player 5 ; ActionScript 1.0

### Exemple

L'exemple suivant montre comment créer un curseur personnalisé intitulé `cursor_mc`. ActionScript est utilisé pour remplacer le curseur de la souris par `cursor_mc`. Ensuite, la fonction `updateAfterEvent()` est utilisée pour actualiser la scène de façon continue afin de lisser le mouvement du curseur.

```
Mouse.hide();
cursor_mc.onMouseMove = function() {
 this._x = this._parent._xmouse;
 this._y = this._parent._ymouse;
 updateAfterEvent();
};
```

### Voir également

`Gestionnaire onClipEvent`, `Fonction setInterval`

# Propriétés globales

Les propriétés globales sont disponibles dans tous les scripts et sont accessibles à tous les scénarios et domaines de votre document. Par exemple, les propriétés globales permettent d'accéder aux scénarios des autres clips chargés, à la fois relatifs (`_parent`) et absolus (`_root`). Elles permettent également de restreindre (`this`) ou d'étendre (`super`) le domaine. Vous pouvez utiliser les propriétés globales pour régler les paramètres d'exécution, tels que la compatibilité avec les lecteurs d'écran, la qualité de la lecture et la taille du tampon audio.

## Récapitulatif des propriétés globales

| Modificateurs | Propriété | Description |
|---|---|---|
| | `_accProps` | Permet de contrôler les options d'accessibilité relatives aux lecteurs d'écran pour les fichiers SWF, les clips, les boutons, les champs de texte dynamique et les champs de texte de saisie lors de l'exécution. |
| | `_focusrect` | Propriété (globale) ; spécifie si un rectangle jaune doit s'afficher autour du bouton ou du clip qui a le focus du clavier. |
| | `_global` | Une référence à l'objet global qui contient les principales classes ActionScript, telles que String, Object, Math et Array. |
| | `_highquality` | *Déconseillé*depuis Flash Player 5. Cette propriété a été déconseillée en faveur de`_quality`.<br>Spécifie le niveau d'anti-aliasing appliqué au fichier SWF actuel. |
| | `_level` | Une référence au scénario racine de `_level` *`N`*. |
| | `maxscroll` | *Déconseillé*depuis Flash Player 5. Cette propriété a été déconseillée en faveur de`TextField.maxscroll`.<br>Indique le numéro de ligne de la ligne supérieure de texte visible dans un champ de texte lorsque la dernière ligne du champ est également visible. |
| | `_parent` | Spécifie ou renvoie une référence au clip ou à l'objet qui contient le clip ou l'objet actuel. |
| | `_quality` | Définit ou extrait la qualité du rendu appliqué à un clip. |
| | `_root` | Spécifie ou renvoie une référence au scénario du clip racine. |

| Modificateurs | Propriété | Description |
|---|---|---|
| | `scroll` | *Déconseillé*depuis Flash Player 5. Cette propriété a été déconseillée en faveur de`TextField.scroll`. Contrôle l'affichage des informations dans un champ de texte associé à une variable. |
| | `_soundbuftime` | Établit le nombre de secondes de son en diffusion continue à placer en mémoire tampon. |
| | `this` | Fait référence à un objet ou une occurrence de clip. |

# _accProps, propriété

`_accProps`.*propertyName*
*instanceName*._accProps.*propertyName*

Permet de contrôler les options d'accessibilité relatives aux lecteurs d'écran pour les fichiers SWF, les clips, les boutons, les champs de texte dynamique et les champs de texte de saisie lors de l'exécution. Ces propriétés remplacent les paramètres correspondants, disponibles dans le panneau Accessibilité lors de la programmation. Pour appliquer les modifications de ces propriétés, vous devez appeler Accessibility.updateProperties().

Pour plus de détails sur le panneau Accessibilité, consultez la section « Panneau Accessibilité de Flash » dans le guide *Utilisation de Flash*.

Pour spécifier si le lecteur doit s'exécuter dans un environnement qui prend en charge les fonctions d'accessibilité, utilisez la méthode System.capabilities.hasAccessibility().

Le tableau suivant répertorie le nom et le type de données de chaque propriété `_accProps`, son paramètre équivalent dans le panneau Accessibilité et les types d'objet auxquels la propriété peut s'appliquer. Le terme *logique inverse* signifie que le paramètre de la propriété est l'inverse du paramètre correspondant dans le panneau Accessibilité. Par exemple, la définition de la propriété `silent` sur `true` (vrai) revient à désélectionner Rendre une animation accessible ou Rendre l'objet accessible.

| Propriété | Type de données | Equivalent dans le panneau Accessibilité | S'applique à |
|---|---|---|---|
| `silent` | Boolean | Rendre une animation accessible/Rendre l'objet accessible *(logique inverse)* | Fichiers SWF entiers Clips Boutons Texte dynamique Texte d'entrée |
| `forceSimple` | Boolean | Rendre les objets enfants accessibles *(logique inverse)* | Fichiers SWF entiers Clips |
| `name` | String | Nom | Fichiers SWF entiers Clips Boutons Texte d'entrée |
| `description` | String | Description | Fichiers SWF entiers Clips Boutons Texte dynamique Texte d'entrée |
| `shortcut` | String | Raccourci | Clips Boutons Texte d'entrée |

Pour le champ Raccourci, utilisez des noms sous la forme Ctrl+A. L'ajout d'un raccourci clavier au panneau Accessibilité ne crée pas un raccourci clavier ; il signale simplement la présence d'un raccourci aux lecteurs d'écran. Pour plus d'informations sur l'affectation d'un raccourci clavier à un objet accessible, consultez la section `Key.addListener()`.

Pour spécifier des paramètres qui correspondent au paramètre Index de tabulation dans le panneau Accessibilité, utilisez les propriétés Button.tabIndex, MovieClip.tabIndex ou TextField.tabIndex.

Il est impossible de spécifier un paramètre Etiquetage auto lors de l'exécution.

Pour faire référence à l'objet `_accProps` représentant le document Flash, omettez le paramètre `instanceName`. La valeur `_accProps` doit être un objet. Ceci signifie que si aucun objet `_accProps` n'existe, vous devez en créer un, comme indiqué dans l'exemple suivant, avant de pouvoir affecter des valeurs aux propriétés de l'objet `_accProps` :

```
if ( _accProps == undefined )
{
 _accProps = new Object();
}
_accProps.name = "My SWF file";
```

Lorsque `_accProps` est utilisé dans le paramètre *`instanceName`*, les modifications apportées aux propriétés `_accProps` s'appliquent à l'ensemble du fichier SWF. Par exemple, le code suivant définit la propriété `name` de la fonction d'accessibilité relative à l'ensemble du fichier SWF sur la chaîne `"Pet Store"`, puis appelle `Accessibility.updateProperties()` pour provoquer ce changement :

```
_accProps.name = "Pet Store";
Accessibility.updateProperties();
```

Par contraste, le code suivant définit la propriété `name` d'un clip, avec le nom d'occurrence `price_mc`, sur la chaîne `"Price"` :

```
price_mc._accProps.name = "Price";
Accessibility.updateProperties();
```

Si vous spécifiez plusieurs propriétés d'accessibilité, apportez autant de changements que nécessaire avant d'appeler `Accessibility.updateProperties()`, au lieu de l'appeler après chaque instruction de propriété, comme indiqué dans l'exemple suivant :

```
_accProps.name = "Pet Store";

animal_mc._accProps.name = "Animal";
animal_mc._accProps.description = "Cat, dog, fish, etc.";

price_mc._accProps.name = "Price";
price_mc._accProps.description = "Cost of a single item";

Accessibility.updateProperties();
```

Si vous ne spécifiez pas de propriété d'accessibilité pour un document ou un objet, toutes les valeurs définies dans le panneau Accessibilité sont implémentées.

Après avoir spécifié une propriété d'accessibilité, vous ne pouvez pas rétablir une valeur définie dans le panneau Accessibilité. Cependant, vous pouvez définir la propriété sur sa valeur par défaut (`false` pour les valeurs booléennes ; des chaînes vides pour les valeurs de type string) en supprimant la propriété de l'objet `_accProps`, comme indiqué dans l'exemple suivant :

```
 my_mc._accProps.silent = true; // set a property
// other code here
delete my_mc._accProps.silent; // revert to default value
```

La valeur _accProps doit être un objet. Ceci signifie que si aucun objet `_accProps` n'existe, vous devez en créer un, avant de pouvoir affecter des indices aux propriétés de l'objet `_accProps`.

```
if (_accProps == undefined)
{
 _accProps = new Object();
}
_accProps.name = "My movie";
```

**Disponibilité :** Flash Player 6,0,65,0; ActionScript 1.0

## Paramètres

*`propertyName`*`:Boolean or String` - Nom de propriété d'accessibilité (consultez la description suivante pour connaître les noms valides). *`instanceName`*

*`instanceName`*`:String` - Nom d'occurrence affecté à une occurrence de clip, un bouton, un champ de texte dynamique ou un champ de texte de saisie. Pour faire référence à l'objet `_accProps` représentant le document Flash, omettez le paramètre *`instanceName`*.

## Exemple

Si vous modifiez une image et souhaitez mettre à jour sa description d'accessibilité, vous pouvez utiliser le code ActionScript suivant :

```
my_mc.gotoAndStop(2);

if (my_mc._accProps == undefined ) {
 my_mc._accProps = new Object();
}

my_mc._accProps.name = "Photo of Mount Rushmore";
Accessibility.updateProperties();
```

## Voir également

`isActive (méthode Accessibility.isActive)`, `updateProperties (méthode Accessibility.updateProperties)`, `hasAccessibility (propriété capabilities.hasAccessibility)`

# _focusrect, propriété

```
_focusrect = Boolean;
```

Spécifie si un rectangle jaune doit s'afficher autour du bouton ou du clip qui a le focus du clavier. Si `_focusrect`est défini sur sa valeur par défaut, `true` (vrai), un rectangle jaune entoure le bouton ou le clip qui a le focus, lorsque l'utilisateur appuie sur la touche de tabulation pour parcourir les objets d'un fichier SWF. Spécifiez `false` (faux) si vous ne souhaitez pas afficher ce rectangle jaune. Cette propriété globale peut être remplacée pour des instances spécifiques.

Si la propriété `_focusrect` est définie sur `false` (faux), le comportement par défaut de tous les boutons et clips est tel que la navigation au clavier se limite à la touche Tab. Toutes les autres touches, ce qui inclut la touche Entrée et les touches directionnelles, sont ignorées. Pour restaurer l'intégralité de l'accès clavier, vous devez définir `_focusrect` sur `true` (vrai). Pour restaurer les fonctionnalités de clavier complet d'un bouton ou d'un clip spécifique, vous pouvez annuler cette propriété globale à l'aide de `Button._focusrect`ou de `MovieClip._focusrect`.

*Remarque :* Si vous utilisez un composant, puis si FocusManager prend le relais de Flash Player pour la gestion du focus, incluez cette propriété globale.

**Disponibilité :** Flash Player 4 ; ActionScript 1.0

## Exemple

L'exemple suivant démontre comment masquer le rectangle jaune autour des occurrences d'un fichier SWF lorsqu'elles ont le focus dans une fenêtre de navigateur. Créez des boutons ou clips et ajoutez le code ActionScript suivant dans l'image 1 du scénario :

```
_focusrect = false;
```

Définissez les paramètres de publication sur Flash Player 6, puis testez le fichier SWF dans une fenêtre de navigateur en pointant sur Fichier > Aperçu avant publication > HTML. Attribuez le focus au fichier SWF en cliquant sur celui-ci dans la fenêtre de navigateur, puis utilisez la touche Tab pour appliquer le focus à chaque occurrence. Si vous appuyez sur la touche Entrée ou Espace lorsque la propriété `_focusrect` est désactivée, cette opération ne permet pas d'appeler le gestionnaire d'événements `onRelease`. En revanche, celui-ci est appelé lorsque la propriété `_focusrect` est activée ou définie sur `true`.

## Voir également

`_focusrect (propriété Button._focusrect)`, `_focusrect (propriété MovieClip._focusrect)`

# propriété _global

```
_global.identifier
```

Une référence à l'objet global qui contient les principales classes ActionScript, telles que String, Object, Math et Array. Par exemple, vous pouvez créer une bibliothèque qui est exposée en tant qu'objet global ActionScript, similaire à l'objet Math ou Date. Contrairement aux variables et aux fonctions déclarées dans le scénario ou en local, les variables et les fonctions globales restent visibles pour tous les scénarios et les domaines du fichier SWF, pourvues qu'elles ne soient pas masquées par des identificateurs portant le même nom dans les domaines internes.

*Remarque :* Quand vous définissez une variable globale, vous devez utiliser le nom entièrement qualifié de la variable, par ex. _global.variableName. Le non respect de cette règle créera une variable locale du même nom qui masque la variable globale que vous essayez de définir.

*Renvoie* Une référence à l'objet global qui contient les principales classes ActionScript, telles que String, Object, Math et Array.

**Disponibilité :** Flash Player 6 ; ActionScript 1.0

## Exemple

L'exemple suivant crée une fonction de haut niveau, `factorial()`,, accessible à tous les scénarios et domaines d'un fichier SWF :

```
_global.factorial = function(n:Number) {
 if(n <= 1) {
 return 1;
 }
 else {
 return n * factorial(n - 1);
 }
}

trace(factorial(1)); // 1
trace(factorial(2)); // 2
trace(factorial(3)); // 6
trace(factorial(4)); // 24
```

L'exemple suivant illustre la façon dont des résultats inattendus sont obtenus si vous ne pouvez pas utiliser le nom complet de la variable lors de la définition de la valeur d'une variable globale :

```
_global.myVar = "globalVariable";
trace(_global.myVar); // globalVariable
trace(myVar); // globalVariable

myVar = "localVariable";
trace(_global.myVar); // globalVariable
```

```
trace(myVar); // localVariable
```

### Voir également

Instruction var, Instruction set variable

## _highquality, propriété

_highquality

*Déconseillé*depuis Flash Player 5. Cette propriété a été déconseillée en faveur de_quality.

Spécifie le niveau d'anti-aliasing appliqué au fichier SWF actuel. Spécifiez 2 (meilleure qualité) pour bénéficier de la meilleure qualité possible et activer le lissage de façon permanente. Spécifiez 1 (haute qualité) pour procéder à l'anti-aliasing ; ceci permet de lisser les bitmaps si le fichier SWF ne contient pas d'animation. Spécifiez 0 (faible qualité) pour empêcher l'anti-aliasing.

**Disponibilité :** Flash Player 4 ; ActionScript 1.0

### Exemple

Le code ActionScript suivant est placé sur le scénario principal et définit la propriété qualité globale de sorte qu'elle applique toujours le lissage des bitmaps dans les fichiers non animés.

```
_highquality = 1;
```

### Voir également

_quality, propriété

## _level, propriété

_level*N*

Une référence au scénario racine de _level *N* . Vous devez utiliser loadMovieNum() pour charger des fichiers SWF dans Flash Player avant d'utiliser la propriété _levelpour les cibler. Vous pouvez également utiliser _level *N* pour cibler un fichier SWF au niveau affecté par *N*.

Le fichier SWF initial qui est chargé dans une occurrence de Flash Player est chargé automatiquement dans _level0. Le fichier SWF dans _level0 définit le débit d'images, la couleur d'arrière-plan et la taille d'image de tous les fichiers SWF chargés par la suite. Les fichiers SWF sont alors empilés dans les niveaux situés au-dessus du fichier SWF de _level0.

Vous devez affecter un niveau à chaque fichier SWF que vous chargez dans Flash Player avec `loadMovieNum()`. L'ordre d'affectation des niveaux n'est pas important. Si vous affectez un niveau qui contient déjà un fichier SWF (ce qui inclut `_level0`), le fichier SWF de ce niveau est purgé et remplacé par le nouveau fichier SWF.

**Disponibilité :** Flash Player 4 ; ActionScript 1.0

### Exemple

L'exemple suivant arrête la tête de lecture dans le scénario principal du fichier SWF sub.swf chargé dans `_level9`. Le fichier sub.swf contient une animation et se trouve dans le même répertoire que le document incluant le code ActionScript suivant :

```
loadMovieNum("sub.swf", 9);
myBtn_btn.onRelease = function() {
 _level9.stop();
};
```

Dans l'exemple précédent, vous pouvez remplacer `_level9.stop()` par le code suivant :

```
_level9.gotoAndStop(5);
```

Cette action place la tête de lecture du scénario principal du fichier SWF chargé dans `_level9` sur l'image 5 au lieu de l'arrêter.

### Voir également

`Fonction loadMovie`, `swapDepths (méthode MovieClip.swapDepths)`

## maxscroll, propriété

*`variable_name`*`.maxscroll`

*Déconseillé*depuis Flash Player 5. Cette propriété a été déconseillée en faveur de`TextField.maxscroll`.

Indique le numéro de ligne de la ligne supérieure de texte visible dans un champ de texte lorsque la dernière ligne du champ est également visible. La propriété `maxscroll` travaille conjointement avec la propriété `scroll` pour contrôler la façon dont les informations apparaissent dans un champ de texte. Cette propriété peut être récupérée mais pas modifiée.

**Disponibilité :** Flash Player 4 ; ActionScript 1.0

### Voir également

`maxscroll (propriété TextField.maxscroll)`, `scroll (propriété TextField.scroll)`

# _parent, propriété

```
_parent.property
_parent._parent.property
```

Spécifie ou renvoie une référence au clip ou à l'objet qui contient le clip ou l'objet actuel. L'objet actuel est l'objet qui contient le code ActionScript faisant référence à `_parent`. Utilisez `_parent` pour spécifier un chemin relatif vers les clips ou les objets situés au-dessus du clip ou de l'objet actuel.

**Disponibilité :** Flash Player 5 ; ActionScript 1.0

## Exemple

Dans l'exemple suivant, un clip portant le nom d'occurrence `square_mc` est placé sur la scène. Un autre clip portant le nom d'occurrence `circle_mc` figure dans ce clip. Le code ActionScript suivant vous permet de modifier l'occurrence `circle_mc` (à savoir `square_mc`) lorsque vous cliquez sur le cercle. Lorsque vous utilisez un adressage relatif (`_parent` au lieu de `_root`), il peut être judicieux d'utiliser le bouton Insérer un chemin cible dans le panneau Actions en premier.

```
this.square_mc.circle_mc.onRelease = function() {
 this._parent._alpha -= 5;
};
```

## Voir également

`_root`, propriété, Fonction `targetPath`

# _quality, propriété

```
_quality:String
```

Définit ou extrait la qualité du rendu appliqué à un clip. Les polices de périphérique sont toujours aliasées, ce qui implique qu'elles ne sont pas affectées par la propriété `_quality`.

La propriété `_quality` peut être définie par les valeurs du tableau suivant.

| Valeur | Description | Anticrénelage des graphiques | Lissage des bitmaps |
|---|---|---|---|
| `"LOW"` | " LOW " Qualité de rendu inférieure. | Les graphiques ne sont pas anticrénelés. | Les bitmaps ne sont pas lissées. |
| `"MEDIUM"` | Qualité de rendu moyenne. Ce niveau de qualité convient aux animations qui ne contiennent pas de texte. | Les graphiques sont anticrénelés en utilisant une grille de 2 x 2 pixels. | Flash Player 8 : Les bitmaps sont lissées sur la base du paramètre `smoothing` utilisé dans les appels `MovieClip.attachBitmap()` et `MovieClip.beginBitmapFill()`.<br>Flash Player 6 et 7 : Les bitmaps ne sont pas lissées. |
| `"HIGH"` | Qualité de rendu supérieure. Il s'agit du paramètre de qualité de rendu par défaut de Flash. | Les graphiques sont anticrénelés en utilisant une grille de 4 x 4 pixels. | Flash Player 8 : Les bitmaps sont lissées sur la base du paramètre `smoothing` utilisé dans les appels `MovieClip.attachBitmap()` et `MovieClip.beginBitmapFill()`.<br>Flash Player 6 et 7 : Les bitmaps sont lissées si le clip est statique. |

| Valeur | Description | Anticrénelage des graphiques | Lissage des bitmaps |
|---|---|---|---|
| "BEST" | Très haute qualité de rendu. | Les graphiques sont anticrénelés en utilisant une grille de 4 x 4 pixels. | Flash Player 8 : Les bitmaps sont lissées sur la base du paramètre smoothing utilisé dans les appels MovieClip.attachBitmap() et MovieClip.beginBitmapFill(). Quand smoothing est réglé sur "Best", le rendu est de qualité supérieure quand le clip est réduit à l'échelle en utilisant un algorithme d'égalisation. Le rendu peut être ralenti, mais cela permet par exemple d'obtenir des vignettes de grandes images en haute qualité.<br>Flash Player 6 et 7 : Les bitmaps sont toujours lissées. |

**Disponibilité :** Flash Player 5 ; ActionScript 1.0

### Exemple

L'exemple suivant définit la qualité du rendu sur LOW :

```
_quality = "LOW";
```

## _root, propriété

_root.*movieClip*
_root.*action*
_root.*property*

Spécifie ou renvoie une référence au scénario du clip racine. Si un clip possède plusieurs niveaux, le scénario du clip racine se situe dans le niveau contenant le script en cours d'exécution. Par exemple, si un script de niveau 1 est évalué comme _root, _level1 est renvoyé.

Le fait de spécifier `_root` revient à utiliser la notation déconseillée, à barre oblique (`/`), pour spécifier un chemin absolu au sein du niveau actuel.

*Remarque :* Si un clip contenant _root est chargé dans un autre clip, _root fait référence au scénario du clip en cours de chargement et non pas au scénario qui contient _root. Si vous souhaitez vous assurer que _root fait référence au scénario du clip chargé, même si ce dernier a été chargé dans un autre clip, utilisez MovieClip._lockroot.

**Disponibilité :** Flash Player 5 ; ActionScript 1.0

### Paramètres

*`movieClip`*`:String` - Nom d'occurrence d'un clip.

*`action`*`:String` - Action ou méthode.

*`property`*`:String` - Propriété de l'objet MovieClip.

### Exemple

L'exemple suivant arrête le scénario du niveau contenant le script en cours d'exécution :

```
_root.stop();
```

L'exemple suivant suit les variables et les occurrences du domaine de `_root` :

```
for (prop in _root) {
 trace("_root."+prop+" = "+_root[prop]);
}
```

### Voir également

`_lockroot (propriété MovieClip._lockroot)`, `_parent, propriété`, `Fonction targetPath`

## scroll, propriété

*`textFieldVariableName`*`.scroll = x`

*Déconseillé*depuis Flash Player 5. Cette propriété a été déconseillée en faveur de`TextField.scroll`.

Contrôle l'affichage des informations dans un champ de texte associé à une variable. La propriété `scroll` définit l'emplacement à partir duquel le champ de texte commence à afficher le contenu ; une fois l'emplacement défini, Flash Player le met à jour lorsque l'utilisateur fait défiler le champ de texte. La propriété `scroll` est utile pour diriger les utilisateurs vers un paragraphe spécifique dans un long passage ou pour créer des champs de texte défilants. Cette propriété peut être récupérée et modifiée.

**Disponibilité :** Flash Player 4 ; ActionScript 1.0

### Exemple

Le code suivant est associé à un bouton Vers le haut qui fait défiler le champ de texte intitulé myText :

```
on (release) {
myText.scroll = myText.scroll + 1;
}
```

### Voir également

`maxscroll (propriété TextField.maxscroll)`, `scroll (propriété TextField.scroll)`

## _soundbuftime, propriété

`_soundbuftime`:*Number* = *integer*

Établit le nombre de secondes de son en diffusion continue à placer en mémoire tampon. La valeur par défaut est de 5 secondes.

**Disponibilité :** Flash Player 4 ; ActionScript 1.0

### Paramètres

*integer*:`Number` - Nombre de secondes précédant la diffusion en continu du fichier SWF.

### Exemple

L'exemple suivant diffuse un fichier MP3 en continu et place le son en mémoire tampon afin qu'il ne soit lu par l'utilisateur. Deux champs de texte dédiés à l'horloge et aux informations de débogage sont créés lors de l'exécution. La propriété `_soundbuftime` est définie de sorte à mettre le fichier MP3 en mémoire tampon pendant 10 secondes. Une nouvelle occurrence de l'objet Sound est créée pour le fichier MP3.

```
// create text fields to hold debug information.
this.createTextField("counter_txt", this.getNextHighestDepth(), 0, 0, 100,
  22);
this.createTextField("debug_txt", this.getNextHighestDepth(), 0, 20, 100,
  22);
// set the sound buffer to 10 seconds.
_soundbuftime = 10;
// create the new sound object instance.
var bg_sound:Sound = new Sound();
// load the MP3 sound file and set streaming to true.
bg_sound.loadSound("yourSound.mp3", true);
// function is triggered when the song finishes loading.
bg_sound.onLoad = function() {
```

```
 debug_txt.text = "sound loaded";
};
debug_txt.text = "sound init";
function updateCounter() {
 counter_txt.text++;
}
counter_txt.text = 0;
setInterval(updateCounter, 1000);
```

# this, propriété

```
this
```

Fait référence à un objet ou une occurrence de clip. Lorsqu'un script s'exécute, `this` référence l'occurrence de clip qui contient le script. Lorsqu'une méthode est appelée, `this` contient une référence à l'objet qui contient la méthode appelée.

Dans un gestionnaire d'événement `on()` associé à un bouton, `this` renvoie au scénario qui contient le bouton. Dans un gestionnaire d'événement `onClipEvent()` associé à un clip, `this` renvoie au scénario du clip.

Dans la mesure où `this` est évalué dans le contexte du script qui le contient, vous ne pouvez pas utiliser `this` pour faire référence à une variable définie dans un fichier de classe.

**Disponibilité :** Flash Player 5 ; ActionScript 1.0

## Exemple

Créez un fichier ActionsScript nommé ApplyThis.as et entrez le code suivant :

```
class ApplyThis {
 var str:String = "Defined in ApplyThis.as";
 function conctStr(x:String):String {
 return x+x;
 }
 function addStr():String {
 return str;
 }
}
```

Ensuite, dans un fichier FLA ou un autre fichier ActionScript, ajoutez le code suivant :

```
var obj:ApplyThis = new ApplyThis();
var abj:ApplyThis = new ApplyThis();
abj.str = "defined in FLA or AS";
trace(obj.addStr.call(abj, null)); //output: defined in FLA or AS
trace(obj.addStr.call(this, null)); //output: undefined
trace(obj.addStr.call(obj, null)); //output: Defined in applyThis.as
```

De même, pour appeler une fonction définie dans une classe dynamique, vous devez utiliser `this` pour appeler la fonction dans le domaine adéquat :

```
// incorrect version of Simple.as
/*
dynamic class Simple {
 function callfunc() {
 trace(func());
 }
}
*/
// correct version of Simple.as
dynamic class simple {
 function callfunc() {
 trace(this.func());
 }
}
```

Dans le fichier FLA ou un autre fichier ActionScript, ajoutez le code suivant :

```
var obj:Simple = new Simple();
obj.num = 0;
obj.func = function() {
 return true;
};
obj.callfunc();
// output: true
```

Le code ci-dessus fonctionne quand vous utilisez `this` dans la méthode callfunc(). Cependant, vous aurez une erreur de syntaxe si vous avez utilisé la mauvaise version de Simple.as, qui a été commentée dans l'exemple ci-dessus.

Dans l'exemple suivant, le mot-clé `this` fait référence à l'objet Circle :

```
function Circle(radius:Number):Void {
 this.radius = radius;
 this.area = Math.PI*Math.pow(radius, 2);
}
var myCircle = new Circle(4);
trace(myCircle.area);
```

Dans l'instruction suivante affectée à une image dans un clip, le mot-clé `this` fait référence au clip actuel.

```
// sets the alpha property of the current movie clip to 20
this._alpha = 20;
```

Dans l'instruction suivante dans un gestionnaire MovieClip.onPress, le mot-clé `this` fait référence au clip actuel :

```
this.square_mc.onPress = function() {
 startDrag(this);
};
this.square_mc.onRelease = function() {
 stopDrag();
};
```

### Voir également

`Gestionnaire on`, `Gestionnaire onClipEvent`

# Opérateurs

Les opérateurs symboliques sont des caractères qui spécifient comment combiner, comparer ou modifier les valeurs d'une expression.

## Récapitulatif des opérateurs

| Opérateur | Description |
|---|---|
| `+ (addition)` | Ajoute des expressions numériques ou concatène (combine) des chaînes. |
| `+= (addition assignment)` | Affecte à *`expression1`* la valeur de *`expression1 + expression2`*. |
| `[] (array access)` | Initialise un nouveau tableau ou tableau multidimensionnel avec les éléments spécifiés (*`a0`* , etc.) ou accède aux éléments dans un tableau. |
| `= (assignment)` | Affecte la valeur d'*`expression2`* (le paramètre de droite) à la variable, à l'élément de tableau ou à la propriété dans *`expression1`*. |
| `& (bitwise AND)` | Convertit *`expression1`* et *`expression2`* en entiers 32 bits non signés et applique une opération booléenne AND sur chaque bit des entiers entrés en tant que paramètres. |
| `&= (bitwise AND assignment)` | Affecte à *`expression1`* la valeur de *`expression1 & expression2`*. |
| `<< (bitwise left shift)` | Convertit *`expression1`* et *`expression2`* en entiers 32 bits et décale tous les bits d'*`expression1`* vers la gauche du nombre d'unités spécifié par l'entier résultant de la conversion d'*`expression2`*. |
| `<<= (bitwise left shift and assignment)` | Cet opérateur effectue un décalage vers la gauche au niveau du bit (`<<=`) et stocke ensuite le contenu dans *`expression1`*. |
| `~ (bitwise NOT)` | Connu également sous la forme de complément d'opérateur du un ou opérateur de complément au niveau du bit. |
| `\| (bitwise OR)` | Convertit *`expression1`* et *`expression2`* en entiers 32 bits non signés et renvoie un 1 pour chaque position de bit où les bits correspondants de *`expression1`* ou *`expression2`* ont la valeur 1. |
| `\|= (bitwise OR assignment)` | Affecte à *`expression1`* la valeur de *`expression1 \| expression2`*. |

| Opérateur | Description |
|---|---|
| `>> (bitwise right shift)` | Convertit *`expression1`* et *`expression2`* en entiers 32 bits et décale tous les bits d'*`expression1`* vers la droite du nombre d'unités spécifié par l'entier résultant de la conversion d'*`expression2`*. |
| `>>= (bitwise right shift and assignment)` | Cet opérateur effectue un décalage vers la droite au niveau du bit et stocke ensuite le contenu dans *`expression1`*. |
| `>>> (bitwise unsigned right shift)` | Identique à l'opérateur de décalage vers la droite (`>>` ) au niveau du bit à l'exception du fait qu'il ne préserve pas le signe de l'*`expression`* d'origine dans la mesure où les bits de gauche sont toujours remplacés par 0. Les nombres à virgule flottante sont convertis en entiers en supprimant tous les chiffres situés après la virgule. |
| `>>>= (bitwise unsigned right shift and assignment)` | Effectue un décalage vers la droite au niveau du bit non signé et stocke ensuite le contenu dans *`expression1`*. |
| `^ (bitwise XOR)` | Convertit *`expression1`* and *`expression2`* en entiers 32 bits non signés et renvoie un 1 pour chaque position de bit où les bits correspondants de *`expression1`* ou *`expression2`*, mais pas les deux, ont la valeur 1. |
| `^= (bitwise XOR assignment)` | Affecte à *`expression1`* la valeur de *`expression1 ^ expression2`*. |
| `/*..*/ (block comment delimiter)` | Démarque une ou plusieurs lignes de commentaires de script. |
| `, (comma)` | Évalue *`expression1`*, puis *`expression2`*, etc. |
| `add (concatenation (strings))` | *Déconseillé*depuis Flash Player 5. Macromedia recommande d'utiliser l'opérateur add (+) lorsque vous créez du contenu pour Flash Player 5 ou supérieur. Cet opérateur n'est pas pris en charge dans Flash Player 8 ou supérieur.<br>Concatène au moins deux chaînes. |
| `?: (conditional)` | Demande à Flash d'évaluer *`expression1`* et si la valeur d'*`expression1`* est `true`, la valeur d'*`expression2`* est renvoyée ; sinon, la valeur d'*`expression3`* est renvoyée. |
| `-- (decrement)` | Un opérateur unaire de pré et post-décrémentation qui soustrait 1 d'*`expression`*. |
| `/ (division)` | Divise *`expression1`* par *`expression2`*. |
| `/= (division assignment)` | Affecte à *`expression1`* la valeur de *`expression1 / expression2`*. |
| `. (dot)` | Permet de naviguer au sein des hiérarchies de clips pour accéder aux clips incorporés (enfants), aux variables ou aux propriétés. |

| Opérateur | Description |
|---|---|
| `== (equality)` | Vérifie si deux expressions sont égales. |
| `eq (equality (strings))` | *Déconseillé*depuis Flash Player 5. Cet opérateur a été déconseillé en faveur de l'opérateur `== (equality)`.<br>Renvoie `true` si la représentation de chaîne de *`expression1`* est égale à celle de *`expression2`*, sinon renvoie `false`. |
| `> (greater than)` | Compare deux expressions et détermine si *`expression1`* est supérieure à *`expression2`* ; dans l'affirmative, cet opérateur renvoie `true`. |
| `gt (greater than (strings))` | *Déconseillé*depuis Flash Player 5. Cet opérateur a été déconseillé en faveur de l'opérateur > (supérieur à).<br>Compare la représentation de chaîne de *`expression1`* avec celle de *`expression2`* et renvoie `true` si *`expression1`* est plus grand que *`expression2`*, sinon renvoie `false`. |
| `>= (greater than or equal to)` | Compare deux expressions et détermine si *`expression1`* est supérieure ou égale à *`expression2`* (`true`) ou si *`expression1`* est inférieure à *`expression2`* (`false`). |
| `ge (greater than or equal to (strings))` | *Déconseillé*depuis Flash Player 5. Cet opérateur a été déconseillé en faveur de l'opérateur >= (supérieur ou égal à).<br>Renvoie `true` si *`expression1`* est supérieure ou égale à *`expression2`*, sinon renvoie`false` . |
| `++ (increment)` | Un opérateur unaire de pré et post-incrémentation qui ajoute 1 à *`expression`* . |
| `!= (inequality)` | Recherche l'inverse de l'opérateur d'égalité (==). |
| `<> (inequality)` | *Déconseillé* depuis Flash Player 5. Cet opérateur est déconseillé. Macromedia recommande d'utiliser l'opérateur `!= (inequality)`.<br>Recherche l'inverse de l'opérateur d'égalité (==). |
| `instanceof` | Teste si `object` est une occurrence de `classConstructor` ou une sous-classe de `classConstructor`. |
| `< (less than)` | Compare deux expressions et détermine si *`expression1`* est inférieure à *`expression2`* ; dans l'affirmative, cet opérateur renvoie `true`. |
| `lt (less than (strings))` | *Déconseillé*depuis Flash Player 5. Cet opérateur a été déconseillé en faveur de l'opérateur < (inférieur à).<br>Renvoie true si l'expression1 est inférieure à l'expression2 ; false sinon. |
| `<= (less than or equal to)` | Compare deux expressions et détermine si *`expression1`* est inférieure ou égale à *`expression2`* ; dans l'affirmative, cet opérateur renvoie `true`. |

| Opérateur | Description |
|---|---|
| le (less than or equal to (strings)) | *Déconseillé*depuis Flash Player 5. Cet opérateur a été déconseillé dans Flash 5 en faveur de l'opérateur <= (inférieur ou égal à).<br>Renvoie true si *expression1* est inférieure ou égale à *expression2*, sinon renvoie false. |
| // (line comment delimiter) | Signale le début d'un commentaire de script. |
| && (logical AND) | Effectue une opération booléenne sur les valeurs des deux expressions. |
| and (logical AND) | *Déconseillé* depuis Flash Player 5. Macromedia recommande d'utiliser l'opérateur AND logique (&&).<br>Effectue une opération AND logique (&&) ) dans Flash Player 4. |
| ! (logical NOT) | Inverse la valeur booléenne d'une variable ou d'une expression. |
| not (logical NOT) | *Déconseillé*depuis Flash Player 5. Cet opérateur a été déconseillé en faveur de l'opérateur ! (logical NOT).<br>Effectue une opération NOT (!) logique dans Flash Player 4. |
| \|\| (logical OR) | Evalue l'*expression1* (l'expression située à gauche de l'opérateur) et renvoie true si cette expression est évaluée à true. |
| or (logical OR) | *Déconseillé*depuis Flash Player 5. Cet opérateur a été déconseillé en faveur de l'opérateur \|\| (logical OR).<br>Evalue *condition1* et *condition2*, si l'une des deux expressions est true, l'expression entière est true. |
| % (modulo) | Calcule le reste de *expression1* divisé par *expression2*. |
| %= (modulo assignment) | Affecte à *expression1* la valeur de *expression1 % expression2*. |
| * (multiplication) | Multiplie deux expressions numériques. |
| *= (multiplication assignment) | Affecte à *expression1* la valeur de *expression1 * expression2*. |
| new | Crée un objet, initialement anonyme, et appelle la fonction identifiée par le paramètre constructor. |
| ne (not equal (strings)) | *Déconseillé*depuis Flash Player 5. Cet opérateur a été déconseillé en faveur de l'opérateur != (inequality).<br>Renvoie true si *expression1* n'est pas égale à *expression2*, sinon renvoie false . |
| {} (object initializer) | Crée un objet et l'initialise avec les paires de propriétés spécifiées *name* et *value*. |

| Opérateur | Description |
|---|---|
| `() (parentheses)` | Effectue une opération de regroupement sur un ou plusieurs paramètres, évalue les expressions de façon séquentielle ou entoure un ou plusieurs paramètres et les transmet en tant que paramètres à une fonction en dehors des parenthèses. |
| `=== (strict equality)` | Teste l'égalité de deux expressions ; l'opérateur d'égalité stricte (===) se comporte de la même façon que l'opérateur d'égalité (==), à la différence que les types de données ne sont pas convertis. |
| `!== (strict inequality)` | Recherche l'inverse exact de l'opérateur d'égalité stricte ( === ). |
| `" (string delimiter)` | Lorsqu'ils entourent des caractères, les guillemets (") indiquent que ces caractères ont une valeur littérale et doivent être traités en tant que *chaîne* et non pas en tant que variable, valeur numérique ou tout autre élément ActionScript. |
| `- (subtraction)` | Utilisé pour la négation ou la soustraction. |
| `-= (subtraction assignment)` | Affecte à *`expression1`* la valeur de *`expression1 - expression2`*. |
| `: (type)` | Utilisé pour le typage strict des données ; cet opérateur spécifie le type de variable, le type de renvoi de la fonction ou le type de paramètre de la fonction. |
| `typeof` | L'opérateur `typeof` évalue l' `expression` et renvoie une chaîne spécifiant si l'expression est une valeur de type `String`, `MovieClip`, `Object`, `Function`, `Number`, ou `Boolean`. |
| `void` | L'opérateur `void` évalue une expression, puis supprime sa valeur, renvoyant `undefined`. |

# Opérateur d'addition +

*`expression1 + expression2`*

Ajoute des expressions numériques ou concatène (combine) des chaînes. Si l'une des expressions est une chaîne, toutes les autres expressions sont converties en chaîne et concaténées. Si les deux expressions sont des entiers, la somme est un entier. Si l'une ou les deux expressions sont des nombres à virgule flottante, la somme est un nombre à virgule flottante.

**Disponibilité :** Flash Player 4 ; ActionScript 1.0

## Opérandes

`expression1` - Nombre ou chaîne.

`expression2 : Number` - Nombre ou chaîne.

## Valeur renvoyée

`Object` - Chaîne, entier ou nombre à virgule flottante.

## Exemple

Utilisation 1 : L'exemple suivant concatène deux chaînes et affiche le résultat dans le panneau de sortie.

```
var name:String = "Cola";
var instrument:String = "Drums";
trace(name + " plays " + instrument); // output: Cola plays Drums
```

Utilisation 2 : Cette instruction additionne les entiers 2 et 3, puis affiche l'entier obtenu, 5, dans le panneau de sortie :

```
trace(2 + 3); // output: 5
```

Cette instruction additionne les nombres à virgule flottante 2,5 et 3,25 puis affiche le nombre obtenu, 5,75 dans le panneau de sortie :

```
trace(2.5 + 3.25); // output: 5.75
```

Utilisation 3 : Le type de données des variables associées aux champs de texte dynamique et de saisie est String. Dans l'exemple suivant, la variable `deposit` est un champ de texte de saisie sur la scène. Lorsque l'utilisateur a entré un nombre pour la variable deposit, le script tente d'additionner `deposit` à `oldBalance`. Toutefois, étant donné que le type de données de `deposit` est String, le script concatène les valeurs de variable (les associe pour former une chaîne) au lieu de les additionner.

```
var oldBalance:Number = 1345.23;
var currentBalance = deposit_txt.text + oldBalance;
trace(currentBalance);
```

Par exemple, si un utilisateur entre 475 dans le champ de texte deposit, l'instruction trace() envoie la valeur 4751345,23 vers le panneau de sortie. Pour y remédier, utilisez la fonction Number() pour convertir la chaîne en nombre de la manière suivante :

```
var oldBalance:Number = 1345.23;
var currentBalance:Number = Number(deposit_txt.text) + oldBalance;
trace(currentBalance);
```

L'exemple suivant montre que les sommes numériques à droite d'une expression de type String ne sont pas calculées :

```
var a:String = 3 + 10 + "asdf";
trace(a); // 13asdf
var b:String = "asdf" + 3 + 10;
trace(b); // asdf310
```

# Opérateur d'affectation de l'addition +=

*expression1* += *expression2*

Affecte à *expression1* la valeur de *expression1* + *expression2*. Par exemple, les deux instructions suivantes ont le même résultat :

```
x += y;
x = x + y;
```

Cette opérateur procède également à la concaténation de chaînes. Toutes les règles de l'opérateur d'addition (+) s'appliquent à l'opérateur d'affectation de l'addition (`(+=)` ).

**Disponibilité :** Flash Player 4 ; ActionScript 1.0

## Opérandes

`expression1 : Number` - Nombre ou chaîne.

`expression2 : Number` - Nombre ou chaîne.

## Valeur renvoyée

`Number` - Résultat de l'addition.

## Exemple

Utilisation 1 : Cet exemple utilise l'opérateur += associé à une expression de type String et envoie « My name is Gilbert » au panneau de sortie.

```
var x1:String = "My name is ";
x1 += "Gilbert";
trace(x1); // output: My name is Gilbert
```

Utilisation 2 : L'exemple suivant illustre une utilisation numérique de l'opérateur d'affectation de l'addition (+=) :

```
var x:Number = 5;
var y:Number = 10;
x += y;
trace(x); // output: 15
```

## Voir également

`Opérateur d'addition +`

# Opérateur d'accès au tableau []

*myArray* = [ *a0, a1,...aN* ]
*myArray*[ *i* ] = *value*
*myObject* [ *propertyName* ]

Initialise un nouveau tableau ou tableau multidimensionnel avec les éléments spécifiés (`a0`, etc.) ou accède aux éléments dans un tableau. L'opérateur d'accès au tableau permet de définir et extraire de façon dynamique une occurrence, une variable et des noms d'objet. Il permet également d'accéder aux propriétés d'objet.

Utilisation 1 : Un tableau est un objet dont les propriétés sont appelées des *éléments*, qui sont tous identifiés par des nombres constituant un *index*. Lorsque vous créez un tableau, vous entourez les éléments avec l'opérateur d'accès au tableau ([]) ou *crochets*). Un tableau peut regrouper différents types d'éléments. Par exemple, le tableau suivant, appelé `employee`, comporte trois éléments ; le premier est un nombre et les deux suivants sont des chaînes (entre guillemets) :

```
var employee:Array = [15, "Barbara", "Jay"];
```

Vous pouvez incorporer des crochets pour représenter les tableaux multi-dimensionnels. Vous pouvez incorporer les tableaux jusqu'à 256 niveaux. Le code suivant crée un tableau appelé `ticTacToe` comportant trois éléments correspondant tous à un tableau de trois éléments :

```
var ticTacToe:Array = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]; // Select Debug >
  List Variables in test mode
// to see a list of the array elements.
```

Utilisation 2 : Mettez l'index de chaque élément entre crochets ([]) pour y accéder directement. Vous pouvez ajouter un nouvel élément à un tableau ou bien modifier ou extraire la valeur d'un élément existant. Le premier index d'un tableau a toujours la valeur 0, comme indiqué dans l'exemple suivant :

```
var my_array:Array = new Array();
my_array[0] = 15;
my_array[1] = "Hello";
my_array[2] = true;
```

Vous pouvez utiliser des crochets ([]) pour ajouter un quatrième élément, comme indiqué dans l'exemple suivant :

```
my_array[3] = "George";
```

Vous pouvez utiliser les crochets ([]) pour accéder à un élément dans un tableau multidimensionnel. La première paire de crochets identifie l'élément dans le tableau d'origine, tandis que la deuxième identifie l'élément dans le tableau incorporé. Les lignes de code suivantes transmettent le chiffre 6 au panneau de sortie.

```
var ticTacToe:Array = [[1, 2, 3], [4, 5, 6], [7, 8, 9]];
trace(ticTacToe[1][2]);// output: 6
```

Utilisation 3 : Vous pouvez utiliser l'opérateur d'accès au tableau ([]) à la place de la fonction `eval()` pour définir et extraire de façon dynamique les valeurs de nom de clip ou toute propriété d'un objet. La ligne de code suivante transmet le chiffre 6 au panneau de sortie.

```
name["mc" + i] = "left_corner";
```

**Disponibilité :** Flash Player 4 ; ActionScript 1.0

## Opérandes

`myArray : Object` - *`myArray`* - Nom d'un tableau.

`a0, a1,...aN : Object` - *`a0,a1,...aN`* Eléments d'un tableau ; tout type natif ou occurrence d'objet, ce qui inclut les tableaux incorporés.

`i : Number` - *`i`* Index entier supérieur ou égal à 0.

`myObject : Object` - *`myObject`* - Nom d'un objet.

`propertyName : String` - *`propertyName`* - Chaîne qui nomme une propriété de l'objet.

## Valeur renvoyée

`Object` -

Utilisation 1 : Référence à un tableau.

Utilisation 2 : Une valeur du tableau ; soit un type natif, soit une occurrence d'objet (ce qui inclut une occurrence de tableau).

Utilisation 3 : Une propriété de l'objet ; soit un type natif, soit une occurrence d'objet (ce qui inclut l'occurrence de tableau).

## Exemple

L'exemple suivant illustre deux façons de créer un objet Array vide ; la première ligne utilise des crochets ([]) :

```
var my_array:Array = [];
var my_array:Array = new Array();
```

L'exemple suivant crée un tableau intitulé `employee_array` et utilise l'instruction trace() pour envoyer les éléments vers le panneau de sortie. À la quatrième ligne, un élément du tableau est modifié, et la cinquième ligne envoie le tableau qui vient d'être modifié vers le panneau de sortie :

```
var employee_array = ["Barbara", "George", "Mary"];
trace(employee_array); // output: Barbara,George,Mary
employee_array[2] = "Sam";
trace(employee_array); // output: Barbara,George,Sam
```

Dans l'exemple suivant, l'expression placée entre crochets (`"piece" + i`) est évaluée et le résultat obtenu est utilisé en tant que nom de la variable à récupérer dans le clip `my_mc`. Dans cet exemple, la variable `i` doit se trouver sur le même scénario que le bouton. Si la variable `i` est égale à 5, par exemple, la valeur de la variable `piece5` dans le clip `my_mc` s'affiche dans le panneau de sortie :

```
myBtn_btn.onRelease = function() {
 x = my_mc["piece"+i];
 trace(x);
};
```

Dans l'exemple suivant, l'expression placée entre crochets est évaluée et le résultat obtenu est utilisé en tant que nom de la variable à récupérer dans le clip `name_mc` :

```
name_mc["A" + i];
```

Si vous maîtrisez la syntaxe à barre oblique ActionScript de Flash 4, vous pouvez utiliser la fonction `eval()` pour obtenir le même résultat :

```
eval("name_mc.A" & i);
```

Vous pouvez utiliser le code ActionScript suivant pour passer en boucle sur tous les objets du domaine `_root` ce qui est particulièrement utile en vue du débogage :

```
for (i in _root) {
 trace(i+": "+_root[i]);
}
```

Vous pouvez également utiliser l'opérateur d'accès au tableau ([]) dans la partie gauche d'une instruction d'affectation pour définir de façon dynamique les noms d'objet, de variable et d'occurrence :

```
employee_array[2] = "Sam";
```

### Voir également

`Array`, `Object`, `Fonction eval`

## Opérateur d'affectation =

*`expression1 = expression2`*

Affecte la valeur d'*`expression2`* (le paramètre de droite) à la variable, à l'élément de tableau ou à la propriété dans *`expression1`*. L'affectation peut se faire par valeur ou par référence. L'affectation par valeur copie la valeur réelle d'*`expression2`* et la place dans *`expression1`*. L'affectation par valeur est utilisée lorsqu'une variable se voit affecter un nombre ou une chaîne de littéral. L'affectation par référence place une référence à *`expression2`* dans *`expression1`*. L'affectation par référence est généralement utilisée avec l'opérateur `new`. L'application de l'opérateur `new` crée un objet en mémoire. Une référence à l'emplacement de cet objet en mémoire est affectée à une variable.

**Disponibilité :** Flash Player 4 ; ActionScript 1.0

## Opérandes

`expression1 : Object` - Variable, élément de tableau ou propriété d'un objet.

`expression2 : Object` - Valeur de tout type.

## Valeur renvoyée

`Object` - Valeur affectée, *`expression2`* .

## Exemple

L'exemple suivant utilise l'affectation par valeur pour affecter la valeur de 5 à la variable `x`.

```
var x:Number = 5;
```

L'exemple suivant utilise l'affectation par valeur pour affecter la valeur « `hello` » à la variable `x` :

```
var x:String;
x = " hello ";
```

L'exemple suivant utilise l'affectation par référence pour créer la variable moonsOfJupiter, qui contient une référence au nouvel objet Array créé. L'affectation par valeur est ensuite utilisée pour copier la valeur « Callisto » dans le premier élément du tableau référencé par la variable `moonsOfJupiter` :

```
var moonsOfJupiter:Array = new Array();
moonsOfJupiter[0] = "Callisto";
```

L'exemple suivant utilise l'affectation par référence pour créer un objet et affecter une référence à cet objet à la variable `mercury`. L'affectation par valeur est ensuite utilisée pour affecter la valeur de `3030` à la propriété `diameter` de l'objet `mercury` :

```
var mercury:Object = new Object(); mercury.diameter = 3030; // in miles
trace (mercury.diameter); // output: 3030
```

L'exemple suivant s'articule autour de l'exemple précédent en créant une variable intitulée `merkur` (le mot allemand désignant le mercure) et en lui affectant la valeur de `mercury`. Deux variables faisant référence au même objet dans la mémoire sont ainsi créées, ce qui signifie que vous pouvez utiliser l'une ou l'autre pour accéder aux propriétés de cet objet. Nous pouvons ensuite modifier la propriété `diameter`pour utiliser les kilomètres au lieu des miles :

```
var merkur:Object = mercury;
merkur.diameter = 4878; // in kilometers
trace (mercury.diameter); // output: 4878
```

## Voir également

`Opérateur == (égalité)`

# & Opérateur AND au niveau du bit

*expression1* & *expression2*

Convertit *expression1* et *expression2* en entiers 32 bits non signés et applique une opération booléenne AND sur chaque bit des entiers entrés en tant que paramètres. Les nombres à virgule flottante sont convertis en entiers en supprimant les chiffres après la virgule. Le résultat est un nouvel entier de 32 bits.

Les entiers positifs sont convertis en valeur hexadécimale non signée dont la valeur maximale est de 4294967295 ou 0xFFFFFFFF. Les valeurs supérieures au maximum perdent leurs chiffres les plus importants lorsqu'elles sont converties, de façon à ce que la valeur demeure à 32 bits. Les nombres négatifs sont convertis en valeur hexadécimale non signée par l'intermédiaire de la notation complément à deux, la valeur minimale étant de -2147483648 ou 0x800000000. Les nombres inférieurs à cette valeur minimale sont convertis en complément à deux avec une plus grande précision et perdent également leurs chiffres les plus importants.

La valeur renvoyée est interprétée en tant que nombre à complément à deux avec un signe, ce qui signifie que la valeur renvoyée est un entier compris entre -2147483648 et 2147483647.

**Disponibilité :** Flash Player 5 ; ActionScript 1.0

## Opérandes

`expression1 : Number` - Nombre.

`expression2 : Number` - Nombre.

## Valeur renvoyée

`Number` - Résultat de l'opération au niveau du bit.

## Exemple

L'exemple suivant compare la représentation des nombres au niveau du bit et renvoie 1 uniquement si les deux bits ont la valeur 1 à la même position. Dans ce code ActionScript, vous ajoutez 13 (binaire 1101) et 11 (binaire 1011) et renvoyez 1 uniquement à la position où les deux nombres ont la valeur 1.

```
var insert:Number = 13;
var update:Number = 11;
trace(insert & update); // output : 9 (or 1001 binary)
```

Pour les nombres 13 et 11, le résultat est 9 car seules les première et dernière positions des deux nombres ont la valeur 1.

Les exemples suivants illustrent le comportement de la conversion de la valeur renvoyée :

```
trace(0xFFFFFFFF); // 4294967295
trace(0xFFFFFFFF & 0xFFFFFFFF); // -1
trace(0xFFFFFFFF & -1); // -1
trace(4294967295 & -1); // -1
trace(4294967295 & 4294967295); // -1
```

### Voir également

`&= Opérateur d'affectation AND au niveau du bit`, `Opérateur ^ (XOR au niveau du bit)`, `Opérateur ^= (affectation XOR au niveau du bit)`, `| Opérateur OR au niveau du bit`, `|= Opérateur d'affectation OR au niveau du bit`, `~ Opérateur NOT au niveau du bit`

## &= Opérateur d'affectation AND au niveau du bit

*`expression1 &= expression2`*

Affecte à *`expression1`* la valeur de *`expression1 & expression2`*. Par exemple, les deux expressions suivantes sont équivalentes :

```
x &= y;
x = x & y;
```

**Disponibilité :** Flash Player 5 ; ActionScript 1.0

### Opérandes

`expression1 : Number` - Nombre.

`expression2 : Number` - Nombre.

### Valeur renvoyée

`Number` - Valeur affectée, *`expression1 & expression2`*.

### Exemple

L'exemple suivant affecte la valeur `9` à `x` :

```
var x:Number = 15;
var y:Number = 9;
trace(x &= y); // output: 9
```

### Voir également

& Opérateur AND au niveau du bit, `Opérateur ^ (XOR au niveau du bit)`, `Opérateur ^= (affectation XOR au niveau du bit)`, `| Opérateur OR au niveau du bit`, `|= Opérateur d'affectation OR au niveau du bit`, `~ Opérateur NOT au niveau du bit`

# << Opérateur de décalage gauche au niveau du bit

*expression1 << expression2*

Convertit `expression1` et `expression2` en valeurs de nombres entiers 32 bits que vous pouvez appeler V1 et V2. Décale tous les bits de la valeur de V1 vers la gauche sur les positions de V2. Rejette les bits décalés à l'extrémité gauche de V1 par cette opération et insère des zéros aux emplacements de bits qui sont vides à droite. Le fait de décaler une valeur d'une unité vers la gauche revient à la multiplier par 2.

Les nombres à virgule flottante sont convertis en entiers en supprimant les chiffres après la virgule. Les entiers positifs sont convertis en valeur hexadécimale non signée dont la valeur maximale est de 4294967295 ou 0xFFFFFFFF. Les valeurs supérieures au maximum perdent leurs chiffres les plus importants lorsqu'elles sont converties, de façon à ce que la valeur demeure à 32 bits. Les nombres négatifs sont convertis en valeur hexadécimale non signée par l'intermédiaire de la notation complément à deux, la valeur minimale étant de -2147483648 ou 0x800000000. Les nombres inférieurs à cette valeur minimale sont convertis en complément à deux avec une plus grande précision et perdent leurs chiffres les plus importants.

La valeur renvoyée est interprétée en tant que nombre à complément à deux avec un signe, ce qui signifie que la valeur renvoyée sera un entier compris entre -2147483648 et 2147483647.

**Disponibilité :** Flash Player 5 ; ActionScript 1.0

## Opérandes

`expression1 : Number` - Nombre ou expression à décaler vers la gauche.

`expression2 : Number` - Nombre ou expression à convertir en entier compris entre 0 et 31.

## Valeur renvoyée

`Number` - Résultat de l'opération au niveau du bit.

## Exemple

Dans l'exemple suivant, l'entier 1 est décalé de 10 bits vers la gauche : `x = 1 << 10` Le résultat de cette opération est `x = 1024`. Ce résultat est dû au fait qu'un 1 décimal égale un 1 binaire, le 1 binaire décalé de 10 bits à gauche est 10000000000 en binaire, et 10000000000 en binaire est 1024 en décimal. Dans l'exemple suivant, l'entier 7 est décalé de 8 bits vers la gauche : `x = 7 << 8` Le résultat de cette opération est `x = 1792`. Ce résultat est dû au fait qu'un 7 décimal égale un 111 binaire, le 111 binaire décalé de 8 bits à gauche est 11100000000 en binaire, et 11100000000 en binaire est 1792 en décimal. Si vous suivez l'exemple suivant, vous remarquez que les bits ont été déplacés de deux espaces vers la gauche :

```
// 2 binary == 0010
// 8 binary == 1000
trace(2 << 2); // output: 8
```

### Voir également

>>= Opérateur de décalage droit au niveau du bit et d'affectation, >> Opérateur de décalage droit au niveau du bit, <<= Opérateur de décalage gauche au niveau du bit et d'affectation, >>> Opérateur de décalage droit non signé au niveau du bit, >>>= Opérateur de décalage droit non signé au niveau du bit et d'affectation

## <<= Opérateur de décalage gauche au niveau du bit et d'affectation

*expression1* <<= *expression2*

Cet opérateur effectue un décalage vers la gauche au niveau du bit (<<=) et stocke ensuite le contenu dans *expression1*. Les deux expressions suivantes sont équivalentes :

```
A <<= B;
A = (A << B)
```

**Disponibilité :** Flash Player 5 ; ActionScript 1.0

### Opérandes

`expression1 : Number` - Nombre ou expression à décaler vers la gauche.

`expression2 : Number` - Nombre ou expression à convertir en entier compris entre 0 et 31.

### Valeur renvoyée

`Number` - Résultat de l'opération au niveau du bit.

### Exemple

Dans l'exemple suivant, vous utilisez l'opérateur de décalage gauche au niveau du bit et d'affectation (<<=) pour décaler tous les bits d'un espace vers la gauche :

```
var x:Number = 4;
// shift all bits one slot to the left.
x <<= 1;
trace(x); // output: 8
// 4 decimal = 0100 binary
// 8 decimal = 1000 binary
```

### Voir également

`<< Opérateur de décalage gauche au niveau du bit`, `>>= Opérateur de décalage droit au niveau du bit et d'affectation`, `>> Opérateur de décalage droit au niveau du bit`

## ~ Opérateur NOT au niveau du bit

*`~expression`*

Connu également sous la forme de complément d'opérateur du un ou opérateur de complément au niveau du bit. Convertit l'*`expression`* en un entier signé de 32 bits, puis applique un complément à un au niveau du bit. Ainsi, tout bit 0 devient 1 et inversement. Le résultat est un nouvel entier signé de 32 bits.

Par exemple, la valeur hexadécimale 0x7777 est représentée de la façon suivante en binaire : 0111011101110111

La négation au niveau du bit de cette valeur, ~0x7777, renvoie : 1000100010001000

En hexadécimal, ceci se traduit par 0x8888. Par conséquent, ~0x7777 donne 0x8888.

L'utilisation la plus répandue des opérateurs au niveau du bit consiste à représenter les *bits indicateurs* (valeurs booléennes contractées sur 1 bit).

Les nombres à virgule flottante sont convertis en entiers en supprimant les chiffres après la virgule. Les entiers positifs sont convertis en valeur hexadécimale non signée dont la valeur maximale est de 4294967295 ou 0xFFFFFFFF. Les valeurs supérieures au maximum perdent leurs chiffres les plus importants lorsqu'elles sont converties, de façon à ce que la valeur demeure à 32 bits. Les nombres négatifs sont convertis en valeur hexadécimale non signée par l'intermédiaire de la notation complément à deux, la valeur minimale étant de -2147483648 ou 0x800000000. Les nombres inférieurs à cette valeur minimale sont convertis en complément à deux avec une plus grande précision et perdent leurs chiffres les plus importants.

La valeur renvoyée est interprétée en tant que nombre à complément à deux avec un signe, ce qui signifie que la valeur renvoyée est un entier compris entre -2147483648 et 2147483647.

**Disponibilité :** Flash Player 5 ; ActionScript 1.0

### Opérandes

`expression : Number` - Nombre.

### Valeur renvoyée

`Number` - Résultat de l'opération au niveau du bit.

### Exemple

L'exemple suivant décrit l'utilisation de l'opérateur NOT (-) au niveau du bit avec les bits indicateurs :

```
var ReadOnlyFlag:Number = 0x0001; // defines bit 0 as the read-only flag
var flags:Number = 0;
trace(flags);
/* To set the read-only flag in the flags variable,
 the following code uses the bitwise OR:
*/
flags |= ReadOnlyFlag;
trace(flags);
/* To clear the read-only flag in the flags variable,
 first construct a mask by using bitwise NOT on ReadOnlyFlag.
 In the mask, every bit is a 1 except for the read-only flag.
 Then, use bitwise AND with the mask to clear the read-only flag.
 The following code constructs the mask and performs the bitwise AND:
*/
flags &= ~ReadOnlyFlag;
trace(flags);
// output: 0 1 0
```

### Voir également

& Opérateur AND au niveau du bit, &= Opérateur d'affectation AND au niveau du bit, Opérateur ^ (XOR au niveau du bit), Opérateur ^= (affectation XOR au niveau du bit), | Opérateur OR au niveau du bit, |= Opérateur d'affectation OR au niveau du bit

## | Opérateur OR au niveau du bit

*expression1* | *expression2*

Convertit *expression1* et *expression2* en entiers 32 bits non signés et renvoie un 1 pour chaque position de bit où les bits correspondants de *expression1* ou *expression2* ont la valeur 1. Les nombres à virgule flottante sont convertis en entiers en supprimant tous les chiffres situés après la virgule. Le résultat est un nouvel entier de 32 bits.

Les entiers positifs sont convertis en valeur hexadécimale non signée dont la valeur maximale est de 4294967295 ou 0xFFFFFFFF. Les valeurs supérieures au maximum perdent leurs chiffres les plus importants lorsqu'elles sont converties, de façon à ce que la valeur demeure à 32 bits. Les nombres négatifs sont convertis en valeur hexadécimale non signée par l'intermédiaire de la notation complément à deux, la valeur minimale étant de -2147483648 ou 0x800000000. Les nombres inférieurs à cette valeur minimale sont convertis en complément à deux avec une plus grande précision et perdent leurs chiffres les plus importants.

La valeur renvoyée est interprétée en tant que nombre à complément à deux avec un signe, ce qui signifie que la valeur renvoyée sera un entier compris entre -2147483648 et 2147483647.

**Disponibilité :** Flash Player 5 ; ActionScript 1.0

### Opérandes

`expression1 : Number` - Nombre.

`expression2 : Number` - Nombre.

### Valeur renvoyée

`Number` - Résultat de l'opération au niveau du bit.

### Exemple

L'exemple suivant illustre une opération OR (|) au niveau du bit :

```
// 15 decimal = 1111 binary
var x:Number = 15;
// 9 decimal = 1001 binary
var y:Number = 9;
// 1111 | 1001 = 1111
trace(x | y); // returns 15 decimal (1111 binary)
```

Ne confondez pas l'opération unique | (OR au niveau du bit) avec l'opérateur || (OR logique).

### Voir également

& Opérateur AND au niveau du bit, `&= Opérateur d'affectation AND au niveau du bit`, `Opérateur ^ (XOR au niveau du bit)`, `Opérateur ^= (affectation XOR au niveau du bit)`, `|= Opérateur d'affectation OR au niveau du bit`, `~ Opérateur NOT au niveau du bit`

## |= Opérateur d'affectation OR au niveau du bit

*`expression1 |= expression2`*

Affecte à *`expression1`* la valeur de *`expression1 | expression2`*. Par exemple, les deux instructions suivantes sont équivalentes :

```
x |= y;
x = x | y;
```

**Disponibilité :** Flash Player 5 ; ActionScript 1.0

### Opérandes

`expression1 : Number` - Nombre ou variable.

`expression2 : Number` - Nombre ou variable.

### Valeur renvoyée

`Number` - Résultat de l'opération au niveau du bit.

### Exemple

L'exemple suivant utilise l'opérateur (`|=`) d'affectation OR au niveau du bit :

```
// 15 decimal = 1111 binary
var x:Number = 15;
// 9 decimal = 1001 binary
var y:Number = 9;
// 1111 |= 1001 = 1111
trace(x |= y); // returns 15 decimal (1111 binary)
```

### Voir également

& Opérateur AND au niveau du bit, `&= Opérateur d'affectation AND au niveau du bit`, `Opérateur ^ (XOR au niveau du bit)`, `Opérateur ^= (affectation XOR au niveau du bit)`, `| Opérateur OR au niveau du bit`, `|= Opérateur d'affectation OR au niveau du bit`, `~ Opérateur NOT au niveau du bit`

## >> Opérateur de décalage droit au niveau du bit

*`expression1 >> expression2`*

Convertit *`expression1`* et *`expression2`* en entiers 32 bits et décale tous les bits d'*`expression1`* vers la droite du nombre d'unités spécifié par l'entier résultant de la conversion d'*`expression2`*. Les bits décalés vers la droite sont supprimés. Pour préserver le signe de l'*`expression`* d'origine, les bits situés à gauche sont remplacés par des 0 si le bit le plus significatif (le bit le plus à gauche) d'*`expression1`* est 0, et par des 1 si le bit le plus significatif est 1. Le décalage d'une valeur d'une unité équivaut à une division par 2 et au rejet du reste.

Les nombres à virgule flottante sont convertis en entiers en supprimant les chiffres après la virgule. Les entiers positifs sont convertis en valeur hexadécimale non signée dont la valeur maximale est de 4294967295 ou 0xFFFFFFFF. Les valeurs supérieures au maximum perdent leurs chiffres les plus importants lorsqu'elles sont converties, de façon à ce que la valeur demeure à 32 bits. Les nombres négatifs sont convertis en valeur hexadécimale non signée par l'intermédiaire de la notation complément à deux, la valeur minimale étant de -2147483648 ou 0x800000000. Les nombres inférieurs à cette valeur minimale sont convertis en complément à deux avec une plus grande précision et perdent leurs chiffres les plus importants.

La valeur renvoyée est interprétée en tant que nombre à complément à deux avec un signe, ce qui signifie que la valeur renvoyée sera un entier compris entre -2147483648 et 2147483647.

**Disponibilité :** Flash Player 5 ; ActionScript 1.0

## Opérandes

`expression1 : Number` - Nombre ou expression à décaler vers la droite.

`expression2 : Number` - Nombre ou expression à convertir en entier compris entre 0 et 31.

## Valeur renvoyée

`Number` - Résultat de l'opération au niveau du bit.

## Exemple

L'exemple suivant convertit 65535 en entier 32 bits et le décale de 8 bits vers la droite :

```
var x:Number = 65535 >> 8;
trace(x); // outputs 255
```

L'exemple suivant affiche le résultat de l'exemple précédent :

```
var x:Number = 255;
```

Ceci est dû au fait que 65535 en décimal équivaut à 1111111111111111 en binaire (seize 1), 1111111111111111 en binaire décalé de 8 bits vers la droite représente 11111111 en binaire, et que 11111111 en binaire est égal à 255 en décimal. Le bit le plus significatif est 0 car il s'agit d'entiers 32 bits, le bit de remplissage est donc 0.

L'exemple suivant convertit -1 en entier 32 bits et le décale de 1 bit vers la droite :

```
var x:Number = -1 >> 1;
trace(x); // outputs -1
```

L'exemple suivant affiche le résultat de l'exemple précédent :

```
var x:Number = -1;
```

Ceci est dû au fait que -1 en décimal équivaut à 11111111111111111111111111111111 en binaire (trente-deux 1), le décalage de un bit vers la droite entraîne la suppression du bit le moins significatif (le bit le plus à droite) et le remplacement du bit le plus significatif par la valeur 1. Le résultat obtenu est 11111111111111111111111111111111 (trente-deux 1) en binaire, soit l'entier 32 bits -1.

### Voir également

`>>= Opérateur de décalage droit au niveau du bit et d'affectation`

## >>= Opérateur de décalage droit au niveau du bit et d'affectation

*`expression1 >>= expression2`*

Cet opérateur effectue un décalage vers la droite au niveau du bit et stocke ensuite le contenu dans *`expression1`*.

Les deux instructions suivantes sont équivalentes :

```
A >>= B;
A = (A >> B);
```

**Disponibilité :** Flash Player 5 ; ActionScript 1.0

### Opérandes

`expression1 : Number` - Nombre ou expression à décaler vers la droite.

`expression2 : Number` - Nombre ou expression à convertir en entier compris entre 0 et 31.

### Valeur renvoyée

`Number` - Résultat de l'opération au niveau du bit.

### Exemple

Le code commenté suivant utilise l'opérateur (>>=) de décalage droit au niveau du bit et d'affectation.

```
function convertToBinary(numberToConvert:Number):String {
 var result:String = "";
 for (var i = 0; i<32; i++) {
 // Extract least significant bit using bitwise AND
 var lsb:Number = numberToConvert & 1;
 // Add this bit to the result
 string result = (lsb ? "1" : "0")+result;
 // Shift numberToConvert right by one bit, to see next bit
 numberToConvert >>= 1;
```

```
 }
 return result;
}
trace(convertToBinary(479));
// Returns the string 00000000000000000000000111011111
// This string is the binary representation of the decimal
// number 479
```

### Voir également

`>> Opérateur de décalage droit au niveau du bit`

## >>> Opérateur de décalage droit non signé au niveau du bit

*`expression1 >>> expression2`*

Identique à l'opérateur de décalage vers la droite au niveau du bit (`>>`), à la différence qu'il ne préserve pas le signe de l'*`expression`* d'origine car les bits de gauche sont toujours remplacés par des 0.

Les nombres à virgule flottante sont convertis en entiers en supprimant les chiffres après la virgule. Les entiers positifs sont convertis en valeur hexadécimale non signée dont la valeur maximale est de 4294967295 ou 0xFFFFFFFF. Les valeurs supérieures au maximum perdent leurs chiffres les plus significatifs lorsqu'elles sont converties, de façon à ce que la valeur demeure à 32 bits. Les nombres négatifs sont convertis en valeur hexadécimale non signée par l'intermédiaire de la notation complément à deux, la valeur minimale étant de -2147483648 ou 0x800000000. Les nombres inférieurs à cette valeur minimale sont convertis en complément à deux avec une plus grande précision et perdent leurs chiffres les plus significatifs.

**Disponibilité :** Flash Player 5 ; ActionScript 1.0

### Opérandes

`expression1 : Number` - Nombre ou expression à décaler vers la droite.

`expression2 : Number` - Nombre ou expression à convertir en entier compris entre 0 et 31.

### Valeur renvoyée

`Number` - Résultat de l'opération au niveau du bit.

### Exemple

L'exemple suivant convertit -1 en entier 32 bits et le décale de 1 bit vers la droite :

```
var x:Number = -1 >>> 1;
```

```
trace(x); // output: 2147483647
```

Ceci est dû au fait que -1 en décimal équivaut à 11111111111111111111111111111111 en binaire (trente-deux 1), et que lorsque vous effectuez un décalage de 1 bit vers la droite (non signé), le bit le moins significatif (le plus à droite) est supprimé, et le bit le plus significatif (le plus à gauche) est remplacé par la valeur 0. Le résultat obtenu est 01111111111111111111111111111111 en binaire, soit l'entier 32 bits 2147483647.

### Voir également

`>>= Opérateur de décalage droit au niveau du bit et d'affectation`

## >>>= Opérateur de décalage droit non signé au niveau du bit et d'affectation

*`expression1 >>>= expression2`*

Effectue un décalage vers la droite au niveau du bit non signé et stocke ensuite le contenu dans *`expression1`*. Les deux instructions suivantes sont équivalentes :

```
A >>>= B;
A = (A >>> B);
```

**Disponibilité :** Flash Player 5 ; ActionScript 1.0

### Opérandes

`expression1 : Number` - Nombre ou expression à décaler vers la droite.

`expression2 : Number` - Nombre ou expression à convertir en entier compris entre 0 et 31.

### Valeur renvoyée

`Number` - Résultat de l'opération au niveau du bit.

### Exemple

### Voir également

`>>> Opérateur de décalage droit non signé au niveau du bit`, `>>= Opérateur de décalage droit au niveau du bit et d'affectation`

## Opérateur ^ (XOR au niveau du bit)

*`expression1 ^ expression2`*

Convertit *expression1* et *expression2* en entiers 32 bits non signés et renvoie un 1 pour chaque position de bit où les bits correspondants de *expression1* ou *expression2*, mais pas les deux, ont la valeur 1. Les nombres à virgule flottante sont convertis en entiers en supprimant tous les chiffres situés après la virgule. Le résultat est un nouvel entier de 32 bits.

Les entiers positifs sont convertis en valeur hexadécimale non signée dont la valeur maximale est de 4294967295 ou 0xFFFFFFFF. Les valeurs supérieures au maximum perdent leurs chiffres les plus significatifs lorsqu'elles sont converties, de façon à ce que la valeur demeure à 32 bits. Les nombres négatifs sont convertis en valeur hexadécimale non signée par l'intermédiaire de la notation complément à deux, la valeur minimale étant de -2147483648 ou 0x800000000. Les nombres inférieurs à cette valeur minimale sont convertis en complément à deux avec une plus grande précision et perdent leurs chiffres les plus significatifs.

La valeur renvoyée est interprétée en tant que nombre à complément à deux avec un signe, ce qui signifie que la valeur renvoyée sera un entier compris entre -2147483648 et 2147483647.

**Disponibilité :** Flash Player 5 ; ActionScript 1.0

## Opérandes

`expression1 : Number` - Nombre.

`expression2 : Number` - Nombre.

## Valeur renvoyée

`Number` - Résultat de l'opération au niveau du bit.

## Exemple

L'exemple suivant utilise l'opérateur XOR au niveau du bit sur les décimales 15 et 9 et affecte le résultat à la variable `x` :

```
// 15 decimal = 1111 binary
// 9 decimal = 1001 binary
var x:Number = 15 ^ 9;
trace(x);
// 1111 ^ 1001 = 0110
// returns 6 decimal (0110 binary)
```

## Voir également

& Opérateur AND au niveau du bit, `&= Opérateur d'affectation AND au niveau du bit`, `Opérateur ^= (affectation XOR au niveau du bit)`, `| Opérateur OR au niveau du bit`, `|= Opérateur d'affectation OR au niveau du bit`, `~ Opérateur NOT au niveau du bit`

# Opérateur ^= (affectation XOR au niveau du bit)

*expression1* `^=` *expression2*

Affecte à *expression1* la valeur de *expression1* `^` *expression2*. Par exemple, les deux instructions suivantes sont équivalentes :

```
x ^= y;
x = x ^ y;
```

**Disponibilité :** Flash Player 5 ; ActionScript 1.0

## Opérandes

`expression1 : Number` - Entiers et variables.

`expression2 : Number` - Entiers et variables.

## Valeur renvoyée

`Number` - Résultat de l'opération au niveau du bit.

## Exemple

L'exemple suivant illustre l'opération (^=) d'affectation XOR au niveau du bit :

```
// 15 decimal = 1111 binary
var x:Number = 15;
// 9 decimal = 1001 binary
var y:Number = 9;
trace(x ^= y); // returns 6 decimal (0110 binary)
```

## Voir également

& Opérateur AND au niveau du bit, `&= Opérateur d'affectation AND au niveau du bit`, `Opérateur ^ (XOR au niveau du bit)`, `| Opérateur OR au niveau du bit`, `|= Opérateur d'affectation OR au niveau du bit`, `~ Opérateur NOT au niveau du bit`

# Opérateur /*..*/ (séparateur de commentaires de bloc)

```
/* comment */
/* comment
comment */
```

Démarque une ou plusieurs lignes de commentaires de script. Tout caractère qui s'affiche entre la balise ouvrante de commentaires (`/*`) et la balise fermante (`*/`) est interprété en tant que commentaire et ignoré par l'interpréteur d'ActionScript. Préférez l'opérateur // (séparateur de commentaires) pour les commentaires sur une ligne. Retenez l'opérateur /* pour identifier les commentaires répartis sur plusieurs lignes. L'omission de la balise fermante (`*/`) renvoie un message d'erreur. Le fait d'incorporer plusieurs balises de commentaires les unes dans les autres renvoie également un message d'erreur. Ainsi, lorsque vous utilisez une balise ouvrante (/*), la première balise fermante (`*/`) termine ce commentaire, quel que soit le nombre de balises (/*) intercalées.

**Disponibilité :** Flash Player 5 ; ActionScript 1.0

### Opérandes

`comment` - Tout caractère.

### Exemple

Le script suivant utilise des séparateurs de commentaires au début du script :

```
/* records the X and Y positions of
the ball and bat movie clips */
var ballX:Number = ball_mc._x;
var ballY:Number = ball_mc._y;
var batX:Number = bat_mc._x;
var batY:Number = bat_mc._y;
```

La tentative d'incorporation suivante de plusieurs balises de commentaires les unes dans les autres renvoie un message d'erreur :

```
/* this is an attempt to nest comments.
/* But the first closing tag will be paired
with the first opening tag */
and this text will not be interpreted as a comment */
```

### Voir également

`Opérateur // (séparateur de commentaires sur une ligne)`

## Opérateur , (virgule)

*`(expression1 , expression2 [, expressionN... ])`*

Évalue *`expression1`*, puis *`expression2`*, etc. Cet opérateur est destiné principalement à l'instruction loop `for` et est souvent utilisé en conjonction avec l'opérateur parenthèses ().

**Disponibilité :** Flash Player 4 ; ActionScript 1.0

## Opérandes

`expression1 : Number` - Expression à évaluer.

`expression2 : Number` - Expression à évaluer.

`expressionN : Number` - Nombre quelconque d'expressions supplémentaires à évaluer.

## Valeur renvoyée

`Object` - Valeur d'*`expression1`*, puis d'*`expression2`*, etc.

## Exemple

L'exemple suivant utilise l'opérateur virgule (,) dans une boucle `for` :

```
for (i = 0, j = 0; i < 3 && j < 3; i++, j+=2) {
 trace("i = " + i + ", j = " + j);
}
// Output:
// i = 0, j = 0
// i = 1, j = 2
```

L'exemple suivant utilise l'opérateur virgule (,) sans l'opérateur parenthèses () et montre que l'opérateur virgule renvoie uniquement la valeur de la première expression sans l'opérateur parenthèses () :

```
var v:Number = 0;
v = 4, 5, 6;
trace(v); // output: 4
```

L'exemple suivant utilise l'opérateur virgule (,) en conjonction avec l'opérateur parenthèses () et montre que l'opérateur virgule renvoie la valeur de la dernière expression lorsqu'il est utilisé avec l'opérateur parenthèses () :

```
var v:Number = 0;
v = (4, 5, 6);
trace(v); // output: 6
```

L'exemple suivant utilise l'opérateur virgule (,) sans l'opérateur parenthèses () et montre que l'opérateur virgule évalue de manière séquentielle toutes les expressions mais renvoie uniquement la valeur de la première expression. La deuxième expression, `z++`, est évaluée et la valeur `z` est incrémentée de un.

```
var v:Number = 0;
var z:Number = 0;
v = v + 4 , z++, v + 6;
trace(v); // output: 4
trace(z); // output: 1
```

L'exemple suivant est identique au précédent à ceci près qu'il inclut l'opérateur parenthèses () et montre à nouveau que, lorsqu'il est utilisé conjointement avec l'opérateur parenthèses (), l'opérateur virgule (,) renvoie la valeur de la dernière expression de la série :

```
var v:Number = 0;
var z:Number = 0;
v = (v + 4, z++, v + 6);
trace(v); // output: 6
trace(z); // output: 1
```

### Voir également

Opérateur () (parenthèses)

## Opérateur de concaténation add (chaînes)

*string1* add *string2*

*Déconseillé*depuis Flash Player 5. Macromedia recommande d'utiliser l'opérateur add (+) lorsque vous créez du contenu pour Flash Player 5 ou version ultérieure. Cet opérateur n'est pas pris en charge dans Flash Player 8 ou supérieur.

Concatène au moins deux chaînes. L'opérateur d'ajout (+) remplace l'opérateur & de Flash 4 ; les fichiers Flash Player 4 qui utilisent l'opérateur & sont automatiquement convertis pour pouvoir utiliser l'opérateur d'ajout (+) en vue de la concaténation de chaînes lorsqu'ils sont importés dans l'environnement de programmation Flash 5 ou version ultérieure. Utilisez l'opérateur d'ajout (+) pour concaténer des chaînes lorsque vous créez du contenu pour Flash Player 4 ou ses versions antérieures.

**Disponibilité :** Flash Player 4 ; ActionScript 1.0

### Opérandes

`string1 : String` - Chaîne.

`string2 : String` - Chaîne.

### Valeur renvoyée

`String` - Chaîne concaténée.

### Voir également

Opérateur d'addition +

## ?: opérateur conditionnel

*expression1* ? *expression2* : *expression3*

Demande à Flash d'évaluer *expression1* et si la valeur d'*expression1* est `true`, la valeur d'*expression2* est renvoyée ; sinon, la valeur d'*expression3* est renvoyée.

**Disponibilité :** Flash Player 4 ; ActionScript 1.0

### Opérandes

`expression1 : Object` - Expression qui renvoie une valeur booléenne ; généralement une expression de comparaison, telle que `x < 5`.

`expression2 : Object` - Valeurs de tout type.

`expression3 : Object` - Valeurs de tout type.

### Valeur renvoyée

`Object` - Valeur de *`expression2`* ou *`expression3`*.

### Exemple

L'instruction suivante affecte la valeur de la variable `x` à la variable `z` car `expression1` renvoie `true` :

```
var x:Number = 5;
var y:Number = 10;
var z = (x < 6) ? x: y;
trace (z); // returns 5
```

L'exemple suivant illustre une instruction conditionnelle abrégée :

```
var timecode:String = (new Date().getHours() < 11) ? "AM" : "PM";
trace(timecode);
```

Cette même instruction conditionnelle peut également être écrite de manière non abrégée, comme indiqué dans l'exemple suivant :

```
if (new Date().getHours() < 11) {
 var timecode:String = "AM";
} else {
 var timecode:String = "PM";
} trace(timecode);
```

# Opérateur -- (décrément)

*`--expression`*
*`expression--`*

Un opérateur unaire de pré et post-décrémentation qui soustrait 1 d' *`expression`* . *`expression`* peut être une variable, un élément de tableau ou une propriété d'objet. La forme post-décrément de l'opérateur (*`--expression`*) soustrait 1 d' *`expression`* et renvoie le résultat. La forme pré-décrément de l'opérateur (*`expression--`*) soustrait 1 de l' *`expression`* et renvoie la valeur initiale d' *`expression`* (la valeur précédant la soustraction).

**Disponibilité :** Flash Player 4 ; ActionScript 1.0

### Opérandes

`expression : Number` - Nombre ou variable qui renvoie un nombre.

### Valeur renvoyée

`Number` - Résultat de la valeur décrémentée.

### Exemple

La forme pré-décrément de l'opérateur décrémente `x` pour obtenir 2 (`x - 1 = 2`) et renvoie le résultat dans `y` :

```
var x:Number = 3;
var y:Number = --x; //y is equal to 2
```

La forme post-décrément de l'opérateur décrémente `x` pour obtenir 2 (`x - 1 = 2`) et renvoie la valeur d'origine de `x`comme résultat `y` :

```
var x:Number = 3;
var y:Number = x--; //y is equal to 3
```

L'exemple suivant boucle de 10 à 1 et chaque itération de la boucle diminue la variable du compteur `i` de 1.

```
for (var i = 10; i>0; i--) {
 trace(i);
}
```

## Opérateur / (division)

*`expression1 / expression2`*

Divise *`expression1`* par *`expression2`*. Le résultat de l'opération de division est un nombre à virgule flottante comportant deux décimales.

**Disponibilité :** Flash Player 4 ; ActionScript 1.0

### Opérandes

`expression : Number` - Nombre ou variable évaluée sous forme de nombre.

### Valeur renvoyée

`Number` - Résultat, en virgule flottante, de l'opération.

### Exemple

L'instruction suivante divise la largeur et la hauteur actuelles de la scène, puis affiche le résultat dans le panneau de sortie.

```
trace(Stage.width/2);
```

```
trace(Stage.height/2);
```

Avec une largeur et une hauteur de scène de 550 x 400 par défaut, on obtient les valeurs 275 et 150.

### Voir également

Opérateur % (modulo)

## Opérateur /= (affectation de division)

*expression1 /= expression2*

Affecte à *expression1* la valeur de *expression1 / expression2*. Par exemple, les deux instructions suivantes sont équivalentes :

```
x /= y;
x = x / y;
```

**Disponibilité :** Flash Player 4 ; ActionScript 1.0

### Opérandes

`expression1 : Number` - Nombre ou variable évaluée sous forme de nombre.

`expression2 : Number` - Nombre ou variable évaluée sous forme de nombre

### Valeur renvoyée

`Number` - Nombre.

### Exemple

Le code suivant indique comment utiliser l'opérateur affectation de division (`/=`) avec des variables et des nombres :

```
var x:Number = 10;
var y:Number = 2;
x /= y; trace(x); // output: 5
```

### Voir également

Opérateur / (division)

## . Opérateur point (.)

*object.property_or_methodinstancename.variable*
*instancename.childinstanceinstancename.childinstance.variable*

Permet de naviguer au sein des hiérarchies de clips pour accéder aux clips incorporés (enfants), aux variables ou aux propriétés. L'opérateur point permet également de tester ou définir les propriétés d'un objet ou d'une classe de premier niveau, d'exécuter une méthode d'un objet ou d'une classe de premier niveau ou de créer une structure de données.

**Disponibilité :** Flash Player 4 ; ActionScript 1.0

## Opérandes

`object : Object` - Occurrence de classe. Cet objet peut être une occurrence de l'une des classes ActionScript intégrées ou d'une classe personnalisée. Ce paramètre figure toujours à gauche de l'opérateur point (.).

`property_or_method` - Nom d'une propriété ou d'une méthode associée à un objet. Toutes les méthodes et les propriétés valides pour les classes intégrées figurent dans les tableaux récapitulatifs des méthodes et des propriétés pour cette classe. Ce paramètre figure toujours à droite de l'opérateur point (.).

`instancename : MovieClip` - Nom d'occurrence d'un clip.`variable` - Le nom d'occurrence à gauche de l'opérateur point (.) peut également représenter une variable sur le scénario du clip.

`childinstance : MovieClip` - Occurrence de clip qui est un enfant d'un autre clip ou qui est incorporée dans ce dernier.

## Valeur renvoyée

`Object` Méthode, propriété ou clip nommés à droite du point.

## Exemple

L'exemple suivant identifie la valeur actuelle de la variable `hairColor` dans le clip `person_mc` :

```
person_mc.hairColor
```

L'environnement de programmation Flash 4 ne prenait pas en charge la syntaxe à point ; en revanche, les fichiers Flash MX 2004 publiés pour Flash Player 4 peuvent utiliser l'opérateur point. L'exemple précédent équivaut à la syntaxe Flash 4 (déconseillée) suivante :

```
/person_mc:hairColor
```

L'exemple suivant crée un nouveau clip dans le domaine `_root`. Ensuite, un champ texte est créé dans le clip intitulé `container_mc`. La propriété `autoSize` du champ de texte est définie sur `true`, puis renseignée avec la date du jour.

```
this.createEmptyMovieClip("container_mc", this.getNextHighestDepth());
this.container_mc.createTextField("date_txt", this.getNextHighestDepth(),
  0, 0, 100, 22);
this.container_mc.date_txt.autoSize = true;
```

```
this.container_mc.date_txt.text = new Date();
```

L'opérateur point (.) est utilisé lorsque vous ciblez des occurrences dans le fichier SWF et lorsque vous devez définir leurs propriétés et valeurs.

# Opérateur == (égalité)

*expression1 == expression2*

Vérifie si deux expressions sont égales. Le résultat est `true` lorsque les expressions sont égales.

La définition de l'égalité dépend du type de données du paramètre :

- Les nombres ou les valeurs booléennes sont considérés comme égaux lorsque leur valeur est identique.
- Les expressions de type String sont égales lorsqu'elles comportent le même nombre de caractères et que ces caractères sont identiques.
- Les variables représentant des objets, des tableaux et des fonctions sont comparées par référence. Deux variables sont égales lorsqu'elles font référence au même objet, au même tableau ou à la même fonction. Deux tableaux distincts ne sont jamais considérés comme égaux, même s'ils comportent le même nombre d'éléments.

Lorsque la comparaison porte sur la valeur, si *expression1* et *expression2* ont un type de donnée différent, ActionScript tente de convertir le type de données d'*expression2* pour le faire correspondre à celui d' *expression1*.

**Disponibilité :** Flash Player 5 ; ActionScript 1.0

## Opérandes

`expression1 : Object` - Nombre, chaîne, valeur booléenne, variable, objet, tableau ou fonction.

`expression2 : Object` - Nombre, chaîne, valeur booléenne, variable, objet, tableau ou fonction.

## Valeur renvoyée

`Boolean` - Résultat booléen de la comparaison.

## Exemple

L'exemple suivant utilise l'opérateur d'égalité (`==`) conjointement avec une instruction `if`:

```
var a:String = "David", b:String = "David";
if (a == b) {
 trace("David is David");
}
```

Les exemples suivants affichent les résultats des opérations qui comparent des types mixtes :

```
var x:Number = 5;
var y:String = "5";
trace(x == y); // output: true
var x:String = "5";
var y:String = "66";
trace(x == y); // output: false
var x:String = "chris";
var y:String = "steve";
trace(x == y); // output: false
```

Les exemples suivants affichent la comparaison par référence. Le premier exemple compare deux tableaux dont la longueur et les éléments sont identiques. L'opérateur d'égalité renvoie la valeur false pour ces deux tableaux. Bien que les tableaux semblent équivalents, la comparaison par référence exige qu'ils se réfèrent tous deux au même tableau. Le deuxième exemple crée la variable thirdArray qui pointe vers le même tableau que la variable firstArray. L'opérateur d'égalité renvoie la valeur true pour ces deux tableaux car les deux variables font référence au même tableau.

```
var firstArray:Array = new Array("one", "two", "three");
var secondArray:Array = new Array("one", "two", "three");
trace(firstArray == secondArray);
// will output false
// Arrays are only considered equal
// if the variables refer to the same array.
var thirdArray:Array = firstArray;
trace(firstArray == thirdArray); // will output true
```

### Voir également

`!` Opérateur NOT logique, Opérateur != (inégalité), Opérateur !== (inégalité stricte), Opérateur && (AND logique), Opérateur || (OR logique), Opérateur === (égalité stricte)

## Opérateur eq d'égalité (chaînes)

*expression1* `eq` *expression2*

*Déconseillé* depuis Flash Player 5. Cet opérateur a été déconseillé en faveur de l'opérateur == (equality).

Compare l'égalité de deux expressions et renvoie une valeur `true` si la chaîne représentant l'*expression1* est égale à la chaîne représentant l'*expression2*, `false` sinon.

**Disponibilité :** Flash Player 4 ; ActionScript 1.0

### Opérandes

`expression1 : Object` - Nombres, chaînes ou variables.

`expression2 : Object` - Nombres, chaînes ou variables.

### Valeur renvoyée

`Boolean` - Résultat de la comparaison.

### Voir également

`Opérateur == (égalité)`

## › Opérateur supérieur à

*`expression1 > expression2`*

Compare deux expressions et détermine si *`expression1`* est supérieure à *`expression2`* ; dans l'affirmative, cet opérateur renvoie `true`. Si *`expression1`* est inférieure ou égale à *`expression2`*, l'opérateur renvoie `false` (faux). Les expressions de type chaîne sont évaluées en fonction de l'ordre alphabétique ; toutes les lettres majuscules précèdent les lettres minuscules.

**Disponibilité :** Flash Player 4 ; ActionScript 1.0

### Opérandes

`expression1 : Object` - Nombre ou chaîne.

`expression2 : Object` - Nombre ou chaîne.

### Valeur renvoyée

`Boolean` - Résultat booléen de la comparaison.

### Exemple

Dans l'exemple suivant, l'opérateur supérieur à (>) est utilisé pour déterminer si la valeur du champ de texte `score_txt` est supérieure à 90 :

```
if (score_txt.text>90) {
 trace("Congratulations, you win!");
} else {
 trace("sorry, try again");
}
```

# Opérateur gt supérieur à (chaînes)

*expression1* gt *expression2*

*Déconseillé*depuis Flash Player 5. Cet opérateur a été déconseillé en faveur de l'opérateur > (supérieur à).

Compare la représentation de chaîne de *expression1* avec celle de *expression2* et renvoie true si *expression1* est supérieur à *expression2*, sinon renvoie false.

**Disponibilité :** Flash Player 4 ; ActionScript 1.0

## Opérandes

expression1 : Object - Nombres, chaînes ou variables.

expression2 : Object - Nombres, chaînes ou variables.

## Valeur renvoyée

Boolean - Résultat booléen de la comparaison.

## Voir également

> Opérateur supérieur à

# Opérateur >= (supérieur ou égal à)

*expression1* >= *expression2*

Compare deux expressions et détermine si *expression1* est supérieure ou égale à *expression2* (true) ou si *expression1* est inférieure à *expression2* (false).

**Disponibilité :** Flash Player 4 ; ActionScript 1.0

## Opérandes

expression1 : Object - Chaîne, entier ou nombre à virgule flottante.

expression2 : Object - Chaîne, entier ou nombre à virgule flottante.

## Valeur renvoyée

Boolean - Résultat booléen de la comparaison.

## Exemple

Dans l'exemple suivant, l'opérateur supérieur ou égal à (>=) est utilisé pour déterminer si l'heure est supérieure ou égale à 12 :

```
if (new Date().getHours() >= 12) {
```

```
 trace("good afternoon");
} else {
 trace("good morning");
}
```

# Opérateur ge supérieur ou égal à (chaînes)

*expression1* ge *expression2*

*Déconseillé*depuis Flash Player 5. Cet opérateur a été déconseillé en faveur de l'opérateur >= (supérieur ou égal à).

Compare la représentation de chaîne de *expression1* avec celle de *expression2* et renvoie `true` si *expression1* est supérieur ou égal à *expression2*, sinon renvoie `false` .

**Disponibilité :** Flash Player 4 ; ActionScript 1.0

## Opérandes

`expression1 : Object` - Nombres, chaînes ou variables.

`expression2 : Object` - Nombres, chaînes ou variables.

## Valeur renvoyée

`Boolean` - Résultat de la comparaison.

## Voir également

Opérateur >= (supérieur ou égal à)

# Opérateur ++ (incrément)

++*expression*
*expression*++

Un opérateur unaire de pré et post-incrémentation qui ajoute 1 à *expression* . *expression* peut être une variable, un élément de tableau ou une propriété d'objet. La forme pré-incrément de l'opérateur (++*expression*) ajoute 1 à *expression* et renvoie le résultat. La forme post-incrément de l'opérateur (*expression*++) ajoute 1 à *expression* et renvoie la valeur initiale d' *expression* (la valeur précédant l'addition).

```
The pre-increment form of the operator increments x to 2 (x + 1 = 2) and
returns the result as y:

var x:Number = 1;
var y:Number = ++x;
trace("x:"+x); //traces x:2
trace("y:"+y); //traces y:2
```

La forme post-incrément de l'opérateur incrémente `x` pour obtenir 2 (`x + 1 = 2`) et renvoie la valeur d'origine de `x`comme résultat `y` :

```
var x:Number = 1;
var y:Number = x++;
trace("x:"+x); //traces x:2
trace("y:"+y); //traces y:1
```

**Disponibilité :** Flash Player 4 ; ActionScript 1.0

## Opérandes

`expression : Number` - Nombre ou variable évaluée sous forme de nombre

## Valeur renvoyée

`Number` - Résultat de l'incrément.

## Exemple

L'exemple suivant utilise ++ en tant qu'opérateur de post-incrémentation pour qu'une boucle `while` s'exécute à cinq reprises :

```
var i:Number = 0;
while (i++ < 5) {
 trace("this is execution " + i);
}
/* output:
 this is execution 1
 this is execution 2
 this is execution 3
 this is execution 4
 this is execution 5
*/
```

L'exemple suivant utilise ++ en tant qu'opérateur de pré-incrémentation :

```
var a:Array = new Array();
var i:Number = 0;
while (i < 10) {
 a.push(++i);
}
trace(a.toString()); //traces: 1,2,3,4,5,6,7,8,9,10
```

Cet exemple utilise également ++ en tant qu'opérateur de pré-incrémentation.

```
var a:Array = [];
for (var i = 1; i <= 10; ++i) {
 a.push(i);
}
trace(a.toString()); //traces: 1,2,3,4,5,6,7,8,9,10
```

Ce script affiche le résultat suivant dans le panneau de sortie : `1,2,3,4,5,6,7,8,9,10`

L'exemple suivant utilise ++ en tant qu'opérateur de post-incrémentation dans une boucle `while` :

```
// using a while loop
var a:Array = new Array();
var i:Number = 0;
while (i < 10) {
 a.push(i++);
}
trace(a.toString()); //traces 0,1,2,3,4,5,6,7,8,9
```

L'exemple suivant utilise ++ en tant qu'opérateur de post-incrémentation dans une boucle `for` :

```
// using a for loop
var a:Array = new Array();
for (var i = 0; i < 10; i++) {
 a.push(i);
}
trace(a.toString()); //traces 0,1,2,3,4,5,6,7,8,9
```

Ce script affiche le résultat suivant dans le panneau de sortie :

```
0,1,2,3,4,5,6,7,8,9
```

## Opérateur != (inégalité)

*expression1* `!=` *expression2*

Recherche l'inverse de l'opérateur d'égalité (==). Si *expression1* est égale à *expression2* , le résultat est `false`. Comme pour l'opérateur d'égalité (==), la définition de l'égalité dépend des types de données comparés, comme illustré dans la liste suivante :

- Les valeurs booléennes, les nombres et les chaînes sont comparés en fonction de leur valeur.
- Les objets, les tableaux et les fonctions sont comparés par référence.
- Une variable est comparée par valeur ou par référence, en fonction de son type.

La comparaison par valeur, comme son nom l'indique signifie que deux expressions ont la même valeur. Par exemple, l'expression (2 + 3) est égale à l'expression (1 + 4) lorsque la comparaison porte sur la valeur.

La comparaison par référence signifie que deux expressions ne sont égales que si elles font toutes deux référence au même objet, tableau ou fonction. Les valeurs figurant dans l'objet, le tableau ou la fonction ne sont pas comparées.

Lorsque la comparaison porte sur la valeur, si *expression1* et *expression2* ont un type de donnée différent, ActionScript tente de convertir le type de données d' *expression2* pour le faire correspondre à celui d'*expression1*.

**Disponibilité :** Flash Player 5 ; ActionScript 1.0

## Opérandes

`expression1 : Object` - Nombre, chaîne, valeur booléenne, variable, objet, tableau ou fonction.

`expression2 : Object` - Nombre, chaîne, valeur booléenne, variable, objet, tableau ou fonction.

## Valeur renvoyée

`Boolean` - Résultat booléen de la comparaison.

## Exemple

L'exemple suivant affiche le résultat de l'opérateur d'inégalité (`!=`) :

```
trace(5 != 8); // returns true
trace(5 != 5) //returns false
```

L'exemple suivant illustre l'utilisation de l'opérateur d'inégalité (`!=`) dans une instruction `if` :

```
var a:String = "David";
var b:String = "Fool";
if (a != b) {
 trace("David is not a fool");
}
```

L'exemple suivant illustre la comparaison par référence avec deux fonctions :

```
var a:Function = function() { trace("foo"); };
var b:Function = function() { trace("foo"); };
a(); // foo
b(); // foo
trace(a != b); // true
a = b;
a(); // foo
b(); // foo
trace(a != b); // false
// trace statement output: foo foo true foo foo false
```

L'exemple suivant illustre la comparaison par référence avec deux tableaux :

```
var a:Array = [ 1, 2, 3 ];
var b:Array = [ 1, 2, 3 ];
trace(a); // 1, 2, 3
trace(b); // 1, 2, 3
trace(a!=b); // true
```

```
a = b;
trace(a); // 1, 2, 3
trace(b); // 1, 2, 3
trace(a != b); // false
// trace statement output: 1,2,3 1,2,3 true 1,2,3 1,2,3 false
```

### Voir également

`! Opérateur NOT logique`, `Opérateur !== (inégalité stricte)`, `Opérateur && (AND logique)`, `Opérateur || (OR logique)`, `Opérateur == (égalité)`, `Opérateur === (égalité stricte)`

## Opérateur d'inégalité <>

*`expression1 <> expression2`*

*Déconseillé* depuis Flash Player 5. Cet opérateur est déconseillé. Macromedia recommande d'utiliser l'opérateur `!= (inequality)`.

Recherche l'inverse de l'opérateur d'égalité (==). Si *`expression1`* est égale à *`expression2`*, le résultat est `false`. Comme pour l'opérateur d'égalité (==), la définition de l'égalité dépend des types de données comparés :

- Les valeurs booléennes, les nombres et les chaînes sont comparés en fonction de leur valeur.
- Les objets, les tableaux et les fonctions sont comparés par référence.
- Les variables sont comparées par valeur ou par référence, en fonction de leur type.

**Disponibilité :** Flash Player 2 ; ActionScript 1.0

### Opérandes

`expression1 : Object` - Nombre, chaîne, valeur booléenne, variable, objet, tableau ou fonction.

`expression2 : Object` - Nombre, chaîne, valeur booléenne, variable, objet, tableau ou fonction.

### Valeur renvoyée

`Boolean` - Résultat booléen de la comparaison.

### Voir également

`Opérateur != (inégalité)`

# Opérateur instanceof

*object* instanceof *classConstructor*

Teste si `object` est une occurrence de `classConstructor` ou une sous-classe de `classConstructor`. L'opérateur `instanceof` ne convertit pas les types primitifs en enveloppes. Par exemple, le code suivant renvoie `true` :

```
new String("Hello") instanceof String;
```

Tandis que le code suivant renvoie false :

```
"Hello" instanceof String;
```

**Disponibilité :** Flash Player 6 ; ActionScript 1.0

## Opérandes

`object : Object` - Objet ActionScript.

`classConstructor : Function` - Référence à une fonction constructeur ActionScript, telle que `String` ou `Date`.

## Valeur renvoyée

`Boolean` - Si `object` est une occurence ou une sous-classe de `classConstructor`, `instanceof` renvoie`true`, sinon il renvoie `false`. De même, `_global instanceof Object` renvoie `false`.

## Voir également

`Opérateur typeof`

# Opérateur < (inférieur à)

*expression1* < *expression2*

Compare deux expressions et détermine si *expression1* est inférieure à *expression2* ; dans l'affirmative, cet opérateur renvoie `true`. Si *expression1* est supérieure ou égale à *expression2*, l'opérateur renvoie `false`. Les expressions de type chaîne sont évaluées en fonction de l'ordre alphabétique ; toutes les lettres majuscules précèdent les lettres minuscules.

**Disponibilité :** Flash Player 4 ; ActionScript 1.0

## Opérandes

`expression1 : Number` - Nombre ou chaîne.

`expression2 : Number` - Nombre ou chaîne.

### Valeur renvoyée

`Boolean` - Résultat booléen de la comparaison.

### Exemple

Les exemples suivants renvoient des valeurs `true` et `false` pour les comparaisons numériques et de type chaîne :

```
trace(3 < 10); // true
trace(10 < 3); // false
trace("Allen" < "Jack"); // true
trace("Jack" < "Allen"); //false
trace("11" < "3"); // true
trace("11" < 3); // false (numeric comparison)
trace("C" < "abc"); // true
trace("A" < "a"); // true
```

## Opérateur lt inférieur à (chaînes)

*expression1* `lt` *expression2*

*Déconseillé*depuis Flash Player 5. Cet opérateur a été déconseillé en faveur de l'opérateur `<` (inférieur à).

Compare *expression1* à *expression2* et renvoie `true` si *expression1* est inférieur à *expression2*, `false` dans les autre cas.

**Disponibilité :** Flash Player 4 ; ActionScript 1.0

### Opérandes

`expression1 : Object` - Nombres, chaînes ou variables.

`expression2 : Object` - Nombres, chaînes ou variables.

### Valeur renvoyée

`Boolean` - Résultat de la comparaison.

### Voir également

`Opérateur < (inférieur à)`

## Opérateur <= (inférieur ou égal à)

*expression1* `<=` *expression2*

Compare deux expressions et détermine si *expression1* est inférieure ou égale à *expression2* ; dans l'affirmative, cet opérateur renvoie `true`. Si *expression1* est supérieure à *expression2*, l'opérateur renvoie `false`. Les expressions de type chaîne sont évaluées en fonction de l'ordre alphabétique ; toutes les lettres majuscules précèdent les lettres minuscules.

**Disponibilité :** Flash Player 4 ; ActionScript 1.0

### Opérandes

`expression1 : Object` - Nombre ou chaîne.

`expression2 : Object` - Nombre ou chaîne.

### Valeur renvoyée

`Boolean` - Résultat booléen de la comparaison.

### Exemple

Les exemples suivants renvoient des valeurs `true` et `false` pour les comparaisons numériques et de type chaîne :

```
trace(5 <= 10); // true
trace(2 <= 2); // true
trace(10 <= 3); // false
trace("Allen" <= "Jack"); // true
trace("Jack" <= "Allen"); // false
trace("11" <= "3"); // true
trace("11" <= 3); // false (numeric comparison)
trace("C" <= "abc"); // true
trace("A" <= a); // true
```

## Opérateur le inférieur ou égal à (chaînes)

*expression1* `le` *expression2*

*Déconseillé*depuis Flash Player 5. Cet opérateur a été déconseillé dans Flash 5 en faveur de l'opérateur <= (inférieur ou égal à).

Compare *expression1* à *expression2* et renvoie `true` si *expression1* est inférieure ou égale à *expression2*, `false` dans les autre cas.

**Disponibilité :** Flash Player 4 ; ActionScript 1.0

### Opérandes

`expression1 : Object` - Nombres, chaînes ou variables.

`expression2 : Object` - Nombres, chaînes ou variables.

### Valeur renvoyée

`Boolean` - Résultat de la comparaison.

### Voir également

`Opérateur <= (inférieur ou égal à)`

## Opérateur // (séparateur de commentaires sur une ligne)

*`// comment`*

Signale le début d'un commentaire de script. Tout caractère qui s'affiche entre le séparateur de commentaires (`//`) et le caractère de fin de ligne est interprété en tant que commentaire et ignoré par l'interpréteur d'ActionScript.

**Disponibilité :** Flash Player 1,0 ; ActionScript 1.0

### Opérandes

`comment` - Tout caractère.

### Exemple

Le script suivant utilise des séparateurs de commentaires pour identifier les première, troisième, cinquième et septième lignes en tant que commentaires :

```
// record the X position of the ball movie clip
var ballX:Number = ball_mc._x;
// record the Y position of the ball movie clip
var ballY:Number = ball_mc._y;
// record the X position of the bat movie clip
var batX:Number = bat_mc._x;
// record the Y position of the ball movie clip
var batY:Number = bat_mc._y;
```

### Voir également

`Opérateur /*..*/ (séparateur de commentaires de bloc)`

## Opérateur && (AND logique)

*`expression1 && expression2`*

Effectue une opération booléenne sur les valeurs des deux expressions. Si *expression1* et *expression2* ont toutes deux la valeur `true`, `true` est renvoyé, sinon `false` est renvoyé.

| Expression | Renvoie |
|---|---|
| `true&&true` | true |
| `true&&false` | false |
| `false&&false` | false |
| `false&&true` | false |

**Disponibilité :** Flash Player 4 ; ActionScript 1.0

## Opérandes

`expression1 : Number` - Valeur booléenne ou expression qui se convertit en valeur booléenne.

`expression2 : Number` - Valeur booléenne ou une expression qui se convertit en valeur booléenne.

## Valeur renvoyée

`Boolean` - Résultat booléen de l'opération logique.

## Exemple

L'exemple suivant utilise l'opérateur AND logique (`&&`) pour effectuer un test permettant de déterminer si un joueur a gagné la partie. Les variables `turns` et `score` sont mises à jour lorsqu'un joueur prend la main ou marque des points au cours de la partie. Le script affiche le texte « You Win the Game ! » dans le panneau de sortie lorsque le score du joueur atteint au moins la valeur 75 pour 3 parties jouées ou moins.

```
var turns:Number = 2;
var score:Number = 77;
if ((turns <= 3) && (score >= 75)) {
 trace("You Win the Game!");
} else {
 trace("Try Again!");
}
// output: You Win the Game!
```

## Voir également

`! Opérateur NOT logique`, `Opérateur != (inégalité)`, `Opérateur !== (inégalité stricte)`, `Opérateur || (OR logique)`, `Opérateur == (égalité)`, `Opérateur === (égalité stricte)`

# Opérateur AND (and logique)

*condition1* `and` *condition2*

*Deconseillé* depuis Flash Player 5. Macromedia recommande d'utiliser l'opérateur AND logique (`&&`).

Effectue une opération AND logique (`&&`) dans Flash Player 4. Si les deux expressions renvoient `true`, l'expression toute entière a la valeur `true`.

**Disponibilité :** Flash Player 4 ; ActionScript 1.0

## Opérandes

`condition1 : Boolean` - *condition1*, *condition2* Conditions ou expressions renvoyant `true` ou `false`.

`condition2 : Boolean` - *condition1*, *condition2* Conditions ou expressions renvoyant `true` ou `false`.

## Valeur renvoyée

`Boolean` - Résultat booléen de l'opération logique.

## Voir également

`Opérateur && (AND logique)`

# ! Opérateur NOT logique

`!` *expression*

Inverse la valeur booléenne d'une variable ou d'une expression. Si *expression* est une variable dont la valeur absolue ou convertie est `true`, la valeur de `!` *expression* est `false`. Si l'expression `x && y` renvoie `false`, l'expression `!(x && y)` renvoie `true`.

Les expressions suivantes illustrent le résultat de l'utilisation de l'opérateur logique NON (!) :

`!` *true* renvoie `false`, `!` *false* renvoie `true`

**Disponibilité :** Flash Player 4 ; ActionScript 1.0

## Opérandes

`expression : Boolean` - Expression ou variable qui renvoie une valeur booléenne.

## Valeur renvoyée

`Boolean` - Résultat booléen de l'opération logique.

### Exemple

Dans l'exemple suivant, la variable `happy` est définie sur `false`. La condition `if` évalue la condition `!happy`, et si elle est `true`, l'instruction trace() envoie une chaîne vers le panneau de sortie.

```
var happy:Boolean = false;
if (!happy) {
 trace("don't worry, be happy"); //traces don't worry, be happy
}
```

L'instruction trace parce que `!false` égale `true`.

### Voir également

`Opérateur != (inégalité)`, `Opérateur !== (inégalité stricte)`, `Opérateur && (AND logique)`, `Opérateur || (OR logique)`, `Opérateur == (égalité)`, `Opérateur === (égalité stricte)`

## Opérateur NOT non logique

`not` *`expression`*

*Déconseillé*depuis Flash Player 5. Cet opérateur a été déconseillé en faveur de l'opérateur `!` `(logical NOT)`.

Effectue une opération NOT logique (!) dans Flash Player 4.

**Disponibilité :** Flash Player 4 ; ActionScript 1.0

### Opérandes

`expression : Object` - Variable ou autre expression qui se convertit en valeur booléenne.

### Valeur renvoyée

`Boolean` - Résultat de l'opération logique.

### Voir également

`! Opérateur NOT logique`

## Opérateur || (OR logique)

*`expression1`* `||` *`expression2`*

Evalue *expression1* (l'expression située à gauche de l'opérateur) et renvoie `true` si cette expression est vraie. Si *expression1* renvoie `false`, *expression2* (l'expression située à droite de l'opérateur) est évaluée. Si *expression2* renvoie `false`, le résultat final est `false`. Sinon, le résultat est `true`.

Si vous utilisez un appel de fonction en tant qu'*expression2*, la fonction ne sera pas exécutée par cet appel si *expression1* renvoie true.

Le résultat est `true`true si l'une des expressions, voire les deux, renvoie(nt) `true`. Le résultat est `false` si et uniquement si les deux expressions renvoient `false`. Vous pouvez utiliser l'opérateur OR logique avec autant d'opérandes que nécessaire. Si l'un des opérandes renvoie `true`, le résultat est `true`.

**Disponibilité :** Flash Player 4 ; ActionScript 1.0

## Opérandes

`expression1 : Number` - Valeur booléenne ou une expression qui se convertit en valeur booléenne.

`expression2 : Number` - Valeur booléenne ou une expression qui se convertit en valeur booléenne.

## Valeur renvoyée

`Boolean` - Résultat de l'opération logique.

## Exemple

L'exemple suivant utilise l'opérateur OR logique (`||`) dans une instruction `if`. La deuxième expression renvoie `true`, par conséquent, le résultat final est `true` :

```
var x:Number = 10;
var y:Number = 250;
var start:Boolean = false;
if ((x > 25) || (y > 200) || (start)) {
 trace("the logical OR test passed"); // output: the logical OR test passed
}
```

Le message « the logical OR test passed » apparaît car l'une des conditions de l'instruction `if` est true (`y>200`). Bien que les deux autres expressions renvoient `false`, tant qu'une condition renvoie `true`, le bloc `if` s'exécute .

L'exemple suivant illustre la façon dont des résultats inattendus peuvent être obtenus si vous utilisez un appel de fonction en tant qu'*expression2*. Si l'expression située à gauche de l'opérateur renvoie `true`, ce résultat est renvoyé sans évaluer l'expression située à droite (la fonction `fx2()` n'est pas appelée).

```
function fx1():Boolean {
```

```
 trace("fx1 called");
 return true;
}
function fx2():Boolean {
 trace("fx2 called");
 return true;
}
if (fx1() || fx2()) {
 trace("IF statement entered");
}
```

Les informations suivantes apparaissent dans le panneau de sortie : fonction fx1 appelée, instruction IF entrée

### Voir également

`! Opérateur NOT logique`, `Opérateur != (inégalité)`, `Opérateur !== (inégalité stricte)`, `Opérateur && (AND logique)`, `Opérateur == (égalité)`, `Opérateur === (égalité stricte)`

## Opérateur OR ou logique

*condition1* `or` *condition2*

*Déconseillé*depuis Flash Player 5. Cet opérateur a été déconseillé en faveur de l'opérateur `||` `(logical OR)`.

Evalue *condition1* et *condition2*, si l'une des deux expressions est `true`, l'expression entière est `true`.

**Disponibilité :** Flash Player 4 ; ActionScript 1.0

### Opérandes

`condition1 : Boolean` - Expression qui prend pour valeur `true` ou `false`.

`condition2 : Boolean` - Expression qui prend pour valeur `true` ou `false`.

### Valeur renvoyée

`Boolean` - Résultat de l'opération logique.

### Voir également

`Opérateur || (OR logique)`, `| Opérateur OR au niveau du bit`

## Opérateur % (modulo)

*expression1* `%` *expression2*

Calcule le reste de *expression1* divisé par *expression2*. Si l'un des paramètres d'*expression* n'est pas numérique, l'opérateur modulo (%) tente de le convertir en nombre. *expression* peut être un nombre ou une chaîne à convertir en valeur numérique.

Le signe du résultat de l'opération modulo correspond au signe du dividende (le premier nombre). Par exemple, `-4 % 3` et `-4 % -3` renvoient tous deux `-1`.

**Disponibilité :** Flash Player 4 ; ActionScript 1.0

### Opérandes

`expression1 : Number` - Nombre ou expression évaluée sous forme de nombre.

`expression2 : Number` - Nombre ou expression évaluée sous forme de nombre.

### Valeur renvoyée

`Number` - Résultat de l'opération arithmétique.

### Exemple

L'exemple numérique suivant utilise l'opérateur modulo (%) :

```
trace(12%5); // traces 2
trace(4.3%2.1); // traces 0.0999999999999996
trace(4%4); // traces 0
```

La première instruction trace renvoie 2, plutôt que 12/5 ou 2,4 car l'opérateur modulo (%) renvoie uniquement le reste. La deuxième instruction trace renvoie 0,0999999999999996 au lieu de la valeur 0,1 attendue en raison des limites d'exactitude des nombres à virgule flottante inhérentes au calcul binaire.

### Voir également

`Opérateur / (division)`, `round (méthode Math.round)`

## Opérateur %= (affectation modulo)

*`expression1 %= expression2`*

Affecte à *expression1* la valeur de *expression1 % expression2*. Les deux instructions suivantes sont équivalentes :

```
x %= y;
x = x % y;
```

**Disponibilité :** Flash Player 4 ; ActionScript 1.0

### Opérandes

`expression1 : Number` - Nombre ou expression évaluée sous forme de nombre.

`expression2 : Number` - Nombre ou expression évaluée sous forme de nombre.

### Valeur renvoyée

`Number` - Résultat de l'opération arithmétique.

### Exemple

L'exemple suivant affecte la valeur `4` à la variable `x` :

```
var x:Number = 14;
var y:Number = 5;
trace(x = y); // output: 4
```

### Voir également

`Opérateur % (modulo)`

## Opérateur * (multiplication)

*`expression1 * expression2`*

Multiplie deux expressions numériques. Lorsque les deux expressions sont des entiers, le produit est un entier. Lorsque l'une ou les deux expressions sont des nombres à virgule flottante, le produit est un nombre à virgule flottante.

**Disponibilité :** Flash Player 4 ; ActionScript 1.0

### Opérandes

`expression1 : Number` - Nombre ou expression évaluée sous forme de nombre.

`expression2 : Number` - Nombre ou expression évaluée sous forme de nombre.

### Valeur renvoyée

`Number` - Entier ou nombre à virgule flottante.

### Exemple

Utilisation 1 : L'instruction suivante multiplie les entiers 2 et 3 :

```
trace(2*3); // output: 6
```

Le résultat est 6 qui correspond à un entier. Utilisation 2 : Cette instruction multiplie les nombres à virgule flottante 2,0 et 3,1416 :

```
trace(2.0 * 3.1416); // output: 6.2832
```

Le résultat est 6,2832 qui correspond à un nombre à virgule flottante.

# Opérateur *= (affectation de multiplication)

*expression1 *= expression2*

Affecte à *expression1* la valeur de *expression1 * expression2*. Par exemple, les deux expressions suivantes sont équivalentes :

```
x *= y;
x = x * y
```

**Disponibilité :** Flash Player 4 ; ActionScript 1.0

## Opérandes

`expression1 : Number` - Nombre ou expression évaluée sous forme de nombre.

`expression2 : Number` - Nombre ou expression évaluée sous forme de nombre.

## Valeur renvoyée

`Number` - Valeur de *expression1 * expression2*. Si une expression ne peut pas être convertie en valeur numérique, elle renvoie `NaN` (non numérique).

## Exemple

Utilisation 1 : L'exemple suivant affecte la valeur 50 à la variable `x` :

```
var x:Number = 5;
var y:Number = 10;
trace(x *= y); // output: 50
```

Utilisation 2 : Les deuxième et troisième lignes de l'exemple suivant calculent les expressions situées à droite du signe égal et affectent les résultats à `x` et `y` :

```
var i:Number = 5;
var x:Number = 4 - 6;
var y:Number = i + 2;
trace(x *= y); // output: -14
```

## Voir également

`Opérateur * (multiplication)`

# Opérateur new

`new` *`constructor()`*

Crée un objet, initialement anonyme, et appelle la fonction identifiée par le paramètre `constructor`. L'opérateur `new` transmet à la fonction les paramètres facultatifs placés entre parenthèses, ainsi que le nouvel objet créé, référencé à l'aide du mot-clé `this`. La fonction `constructor` peut ensuite utiliser `this` pour définir les variables de l'objet.

**Disponibilité :** Flash Player 5 ; ActionScript 1.0

### Opérandes

`constructor : Object` - Fonction suivie des paramètres facultatifs placés entre parenthèses. La fonction correspond généralement au nom du type d'objet (par exemple, `Array`, `Number`, ou `Object`) à construire.

### Exemple

L'exemple suivant crée la fonction `Book()`, puis utilise l'opérateur `new` pour créer les objets `book1` et `book2`.

```
function Book(name, price){
 this.name = name;
 this.price = price;
}

book1 = new Book("Confederacy of Dunces", 19.95);
book2 = new Book("The Floating Opera", 10.95);
```

L'exemple suivant utilise l'opérateur `new` pour créer un objet `Array` incluant 18 éléments :

```
golfCourse_array = new Array(18);
```

### Voir également

`Opérateur d'accès au tableau []`, `Opérateur {} (initialiseur d'objet)`

## Opérateur ne n'est pas égal à (chaînes)

*`expression1`* `ne` *`expression2`*

*Déconseillé*depuis Flash Player 5. Cet opérateur a été déconseillé en faveur de l'opérateur `!= (inequality)`.

Compare *`expression1`* à *`expression2`* et renvoie `true` si *`expression1`* n'est pas égal à *`expression2`*, `false` dans les autre cas.

**Disponibilité :** Flash Player 4 ; ActionScript 1.0

### Opérandes

`expression1 : Object` - Nombres, chaînes ou variables.

`expression2 : Object` - Nombres, chaînes ou variables.

### Valeur renvoyée

`Boolean` - Renvoie true si expression1 n'est pas égal à expression2 ; false sinon.

### Voir également

Opérateur != (inégalité)

## Opérateur {} (initialiseur d'objet)

*object* = { *name1 : value1 , name2 : value2 ,... nameN : valueN* }
{*expression1; [...expressionN]*}

Crée un objet et l'initialise avec les paires de propriétés spécifiées *name* et *value*. L'utilisation de cet opérateur a le même effet que la syntaxe `new Object` et le fait de compléter des paires de propriétés avec l'opérateur d'affectation. Le prototype du nouvel objet est génériquement appelé Object.

Cet opérateur est également utilisé pour marquer des blocs de code contigus associés aux instructions de contrôle du flux (`for`, `while`, `if`, `else`, `switch`) et aux fonctions.

**Disponibilité :** Flash Player 5 ; ActionScript 1.0

### Opérandes

`object : Object` - Oobjet à créer. *name1,2,...N* - Noms des propriétés. *value1,2,...N* - Valeurs correspondantes pour chaque propriété *name*.

### Valeur renvoyée

`Object` -

Utilisation 1 : Un objet Object.

Utilisation 2 : Rien, sauf lorsqu'une fonction renvoie une instruction `return` explicite, auquel cas le type renvoyé est spécifié lors de l'implémentation de la fonction.

### Exemple

La première ligne du code suivant crée un objet vide à l'aide de l'opérateur ({}) initialiseur d'objet ; la deuxième ligne crée un nouvel objet à l'aide d'une fonction constructeur :

```
var object:Object = {};
var object:Object = new Object();
```

L'exemple suivant crée un objet `account` et initialise les propriétés `name`, `address`, `city`, `state`, `zip`, et `balance` avec les valeurs suivantes :

```
var account:Object = {name:"Macromedia, Inc.", address:"600 Townsend
  Street", city:"San Francisco", state:"California", zip:"94103",
  balance:"1000"};
for (i in account) {
 trace("account." + i + " = " + account[i]);
}
```

L'exemple suivant indique comment imbriquer un tableau et des initialiseurs d'objet :

```
var person:Object = {name:"Gina Vechio", children:["Ruby", "Chickie",
  "Puppa"]};
```

L'exemple suivant utilise les informations de l'exemple précédent et permet d'obtenir le même résultat à l'aide des fonctions constructeur :

```
var person:Object = new Object();
person.name = "Gina Vechio";
person.children = new Array();
person.children[0] = "Ruby";
person.children[1] = "Chickie";
person.children[2] = "Puppa";
```

L'exemple ActionScript précédent peut également être écrit au format suivant :

```
var person:Object = new Object();
person.name = "Gina Vechio";
person.children = new Array("Ruby", "Chickie", "Puppa");
```

### Voir également

`Object`

## Opérateur () (parenthèses)

`(`*expression1 [, expression2]*`)`
`(` *expression1, expression2* `)`
*function* `(` *parameter1,..., parameterN* `)`

Effectue une opération de regroupement sur un ou plusieurs paramètres, évalue les expressions de façon séquentielle ou entoure un ou plusieurs paramètres et les transmet en tant que paramètres à une fonction en dehors des parenthèses.

Utilisation 1 : Contrôle l'ordre suivant lequel les opérateurs s'exécutent dans l'expression. Les parenthèses remplacent la séquence normale et entraînent l'évaluation des expressions entre parenthèses en premier. Lorsque les parenthèses sont imbriquées, le contenu entre les parenthèses de plus bas niveau est évalué en premier.

Utilisation 2 : Évalue une série d'expressions, séparées par des virgules, dans la séquence et renvoie le résultat de l'expression finale.

Utilisation 3 : Entoure un ou plusieurs paramètres et les transmet en tant que paramètres à la fonction située en dehors des parenthèses.

**Disponibilité :** Flash Player 4 ; ActionScript 1.0

### Opérandes

`expression1 : Object` - Nombres, chaînes, variables ou texte.

`expression2 : Object` - Nombres, chaînes, variables ou texte.

`function : Function` - Fonction à exécuter sur le contenu des parenthèses.

`parameter1...parameterN : Object` - Série de paramètres à exécuter avant de transmettre les résultats sous forme de paramètres à la fonction située en-dehors des parenthèses.

### Exemple

Utilisation 1 : Les instructions suivantes illustrent l'utilisation de parenthèses afin de contrôler l'ordre dans lequel les expressions sont exécutées (la valeur de chaque expression apparaît dans le panneau de sortie) :

```
trace((2 + 3)*(4 + 5)); // Output: 45
trace((2 + 3) * (4 + 5)); // Output: 45trace(2 + (3 * (4 + 5))); // //
  writes 29
trace(2 + (3 * (4 + 5))); // Output: 29trace(2+(3*4)+5); // writes 19
trace(2 + (3 * 4) + 5); // Output: 19
```

Utilisation 2 : L'exemple suivant évalue la fonction `foo()`, puis la fonction `bar()` et renvoie le résultat de l'expression `a + b` :

```
var a:Number = 1;
var b:Number = 2;
function foo() { a += b; }
function bar() { b *= 10; }
trace((foo(), bar(), a + b)); // outputs 23
```

Utilisation 3 : L'exemple suivant illustre l'utilisation des parenthèses avec des fonctions :

```
var today:Date = new Date();
trace(today.getFullYear()); // traces current year
function traceParameter(param):Void { trace(param); }
traceParameter(2 * 2); //traces 4
```

### Voir également

Instruction with

## Opérateur === (égalité stricte)

*`expression1 === expression2`*

Teste l'égalité de deux expressions ; l'opérateur d'égalité stricte (===) se comporte de la même façon que l'opérateur d'égalité (==), à la différence que les types de données ne sont pas convertis. Le résultat est `true` lorsque les deux expressions sont égales, types de données inclus.

La définition de l'égalité dépend du type de données du paramètre :

- Les nombres ou les valeurs booléennes sont considérés comme égaux lorsque leur valeur est identique.
- Les expressions de type String sont égales lorsqu'elles comportent le même nombre de caractères et que ces caractères sont identiques.
- Les variables représentant des objets, des tableaux et des fonctions sont comparées par référence. Deux variables sont égales lorsqu'elles font référence au même objet, au même tableau ou à la même fonction. Deux tableaux distincts ne sont jamais considérés comme égaux, même s'ils comportent le même nombre d'éléments.

**Disponibilité :** Flash Player 6 ; ActionScript 1.0

## Opérandes

`expression1 : Object` - Nombre, chaîne, valeur booléenne, variable, objet, tableau ou fonction.

`expression2 : Object` - Nombre, chaîne, valeur booléenne, variable, objet, tableau ou fonction.

## Valeur renvoyée

`Boolean` - Résultat booléen de la comparaison.

## Exemple

Les commentaires inclus dans le code suivant affichent la valeur renvoyée des opérations qui utilisent les opérateurs d'égalité et d'égalité stricte :

```
// Both return true because no conversion is done
var string1:String = "5";
var string2:String = "5";
trace(string1 == string2); // true
trace(string1 === string2); // true
// Automatic data typing in this example converts 5 to "5"
var string1:String = "5";
var num:Number = 5;
trace(string1 == num); // true
trace(string1 === num); // false
// Automatic data typing in this example converts true to "1"
var string1:String = "1";
var bool1:Boolean = true;
trace(string1 == bool1); // true
trace(string1 === bool1); // false
// Automatic data typing in this example converts false to "0"
var string1:String = "0";
var bool2:Boolean = false;
trace(string1 == bool2); // true
trace(string1 === bool2); // false
```

Les exemples suivants illustrent la façon dont l'opérateur d'égalité stricte traite les références de variables différemment des variables incluant des valeurs littérales. C'est l'une des raisons pour laquelle il convient d'utiliser de façon systématique des littéraux de chaîne et d'éviter d'utiliser l'opérateur `new` avec la classe String.

```
// Create a string variable using a literal value
var str:String = "asdf";
// Create a variable that is a reference
var stringRef:String = new String("asdf");
// The equality operator does not distinguish among literals, variables,
// and references
trace(stringRef == "asdf"); // true
trace(stringRef == str); // true
trace("asdf" == str); // true
// The strict equality operator considers variables that are references
// distinct from literals and variables
trace(stringRef === "asdf"); // false
trace(stringRef === str); // false
```

### Voir également

`! Opérateur NOT logique`, `Opérateur != (inégalité)`, `Opérateur !== (inégalité stricte)`, `Opérateur && (AND logique)`, `Opérateur || (OR logique)`, `Opérateur == (égalité)`

## Opérateur !== (inégalité stricte)

*`expression1`* `!==` *`expression2`*

Recherche l'inverse exact de l'opérateur d'égalité stricte (`===` ). L'opérateur d'inégalité stricte opère de la même façon que l'opérateur d'inégalité, à la différence que le type de données n'est pas converti.

Si *`expression1`* est égal à *`expression2`*, et que leurs types de données sont égaux, le résultat est `false` Comme pour l'opérateur d'égalité stricte (===), la définition de l'égalité dépend des types de données comparés, comme illustré dans la liste suivante :

- Les valeurs booléennes, les nombres et les chaînes sont comparés en fonction de leur valeur.
- Les objets, les tableaux et les fonctions sont comparés par référence.
- Une variable est comparée par valeur ou par référence, en fonction de son type.

**Disponibilité :** Flash Player 6 ; ActionScript 1.0

### Opérandes

`expression1 : Object` - Nombre, chaîne, valeur booléenne, variable, objet, tableau ou fonction.

`expression2 : Object` - Nombre, chaîne, valeur booléenne, variable, objet, tableau ou fonction.

### Valeur renvoyée

`Boolean` - Résultat booléen de la comparaison.

### Exemple

Les commentaires inclus dans le code suivant affichent la valeur renvoyée des opérations qui utilisent les opérateurs d'égalité (==), d'égalité stricte (===) et d'inégalité stricte (!==):

```
var s1:String = "5";
var s2:String = "5";
var s3:String = "Hello";
var n:Number = 5;
var b:Boolean = true;
trace(s1 == s2); // true
trace(s1 == s3); // false
trace(s1 == n); // true
trace(s1 == b); // false
trace(s1 === s2); // true
trace(s1 === s3); // false
trace(s1 === n); // false
trace(s1 === b); // false
trace(s1 !== s2); // false
trace(s1 !== s3); // true
trace(s1 !== n); // true
trace(s1 !== b); // true
```

### Voir également

`! Opérateur NOT logique`, `Opérateur != (inégalité)`, `Opérateur && (AND logique)`, `Opérateur || (OR logique)`, `Opérateur == (égalité)`, `Opérateur === (égalité stricte)`

## Opérateur " (séparateur de chaîne)

`"text"`

Lorsqu'ils entourent des caractères, les guillemets (") indiquent que ces caractères ont une valeur littérale et doivent être traités en tant que *chaîne* et non pas en tant que variable, valeur numérique ou tout autre élément ActionScript.

**Disponibilité :** Flash Player 4 ; ActionScript 1.0

### Opérandes

`text : String` - Séquence de zéros ou de plusieurs caractères.

### Exemple

L'exemple suivant utilise des guillemets (") pour indiquer que la valeur de la variable *yourGuess* est la chaîne littérale `"Prince Edward Island"`, et non pas le nom d'une variable. La valeur `province` est une variable, et non un littéral ; pour déterminer la valeur de `province`, la valeur de *yourGuess* doit être déterminée.

```
var yourGuess:String = "Prince Edward Island";
submit_btn.onRelease = function() { trace(yourGuess); };
// displays Prince Edward Island
```

### Voir également

`String`, `Fonction String`

## Opérateur - (soustraction)

`(Negation) -`*expression*
`(Subtraction)` *expression1 - expression2*

Utilisé pour la négation ou la soustraction.

Utilisation 1 : Lorsque cet opérateur est utilisé pour la négation, il inverse le signe de l'expression numérique *`expression`*. Utilisation 2 : Lorsqu'il est utilisé pour la soustraction, il effectue une soustraction arithmétique sur deux expressions numériques, en soustrayant *`expression2`* de *`expression1`*. Lorsque les deux expressions sont des entiers, la différence est un entier. Lorsque l'une ou les deux expressions sont des nombres à virgule flottante, la différence est un nombre à virgule flottante.

**Disponibilité :** Flash Player 4 ; ActionScript 1.0

### Opérandes

`expression1 : Number` - Nombre ou expression évaluée sous forme de nombre.

`expression2 : Number` - Nombre ou expression évaluée sous forme de nombre.

### Valeur renvoyée

`Number` - Entier ou nombre à virgule flottante.

### Exemple

Utilisation 1 : L'instruction suivante inverse le signe de l'expression 2 + 3 :

```
trace(-(2+3)); // output: -5
```

Utilisation 2 : L'instruction suivante soustrait l'entier 2 de l'entier 5 :

```
trace(5-2); // output: 3
```

Le résultat est 3 qui correspond à un entier. Utilisation 3 : L'instruction suivante soustrait le nombre à virgule flottante 1,5 du nombre à virgule flottante 3,25 :

```
trace(3.25-1.5); // output: 1.75
```

Le résultat est 1,75 qui correspond à un nombre à virgule flottante.

## Opérateur -= (affectation de soustraction)

*expression1* `-=` *expression2*

Affecte à *expression1* la valeur de *expression1* `-` *expression2*. Par exemple, les deux instructions suivantes sont équivalentes : `x -= y ;x = x - y;`

Les expressions de type String doivent être converties en nombres. Sinon, `NaN` (non numérique) est renvoyé.

**Disponibilité :** Flash Player 4 ; ActionScript 1.0

### Opérandes

`expression1 : Number` - Nombre ou expression évaluée sous forme de nombre.

`expression2 : Number` - Nombre ou expression évaluée sous forme de nombre.

### Valeur renvoyée

`Number` - Résultat de l'opération arithmétique.

### Exemple

L'exemple suivant utilise l'opérateur (`-=`) d'affectation de soustraction pour soustraire 10 de 5 et affecte le résultat à la variable `x` :

```
var x:Number = 5;
var y:Number = 10;
x -= y; trace(x); // output: -5
```

L'exemple suivant indique comment convertir des chaînes en nombres :

```
var x:String = "5";
var y:String = "10";
x -= y; trace(x); // output: -5
```

### Voir également

`Opérateur - (soustraction)`

## : Opérateur

*[ modifiers ] var variableName* : *type*
*function functionName ()* : *type { ... }*
*function functionName* ( *parameter1*:*type , ... , parameterN*:*type* ) *[ :type ]{ ... }*

Utilisé pour le typage strict des données ; cet opérateur spécifie le type de variable, le type de renvoi de la fonction ou le type de paramètre de la fonction. Lorsqu'il est utilisé dans une déclaration une affectation de variable, cet opérateur spécifie le type de variable. Lorsqu'il fait partie d'une déclaration ou une définition de fonction, cet opérateur spécifie le type de renvoi de la fonction. Lorsqu'il est utilisé avec un paramètre de fonction dans une définition de fonction, cet opérateur spécifie le type de variable attendu pour ce paramètre.

Un type est une fonction de compilation uniquement. Tous les types sont vérifiés lors de la compilation et des erreurs sont générées en cas d'incompatibilité. Les incompatibilités peuvent se produire pendant les opérations d'affectation, les appels de fonction et les ruptures de référence des membres de classe avec l'opérateur (.). Pour éviter les erreurs liées aux incompatibilités, appliquez le typage strict des données.

Les types utilisables incluent tous les types d'objet, les classes et les interfaces natifs que vous avez définis, ainsi que Function et Void. Les types natifs reconnus sont Boolean, Number et String. Toutes les classes intégrées sont également prises en charge en tant que types natifs.

**Disponibilité :** Flash Player 6 ; ActionScript 1.0

### Opérandes

`variableName : Object` - Identificateur pour une variable. *`type`* - Type de données natif, nom de classe que vous avez défini ou nom d'interface. *`functionName`* - Identificateur pour une fonction. *`parameter`* - Identificateur pour un paramètre de fonction.

### Exemple

Utilisation 1 : L'exemple suivant déclare une variable publique intitulée `userName` de type String et lui affecte une chaîne vide :

```
var userName:String = "";
```

Utilisation 2 : L'exemple suivant indique comment spécifier le type de paramètre d'une fonction en définissant une fonction intitulée `randomInt()` qui prend un paramètre intitulé `integer` de type Number :

```
function randomInt(integer:Number):Number {
 return Math.round(Math.random()*integer);
```

```
}
trace(randomInt(8));
```

Utilisation 3 : L'exemple suivant définit une fonction intitulée s`squareRoot()` qui prend un paramètre intitulé `val` de type Number et renvoie la racine carrée de `val`, également de type Number :

```
function squareRoot(val:Number):Number {
 return Math.sqrt(val);
}
trace(squareRoot(121));
```

### Voir également

`Instruction var`, `Instruction function`

## Opérateur typeof

`typeof(`*`expression`*`)`

Evalue l' `expression` et renvoie une chaîne spécifiant si l'expression est une valeur de type `String`, `MovieClip`, `Object`, `Function`, `Number`, ou `Boolean`.

**Disponibilité :** Flash Player 5 ; ActionScript 1.0

### Opérandes

`expression : Object` - Chaîne, clip, bouton, objet ou fonction.

### Valeur renvoyée

`String` - Représentation sous forme de `String` du type d'`expression`. Le tableau suivant affiche les résultats de l'opérateur `typeof` pour chaque type d'`expression`.

| Type d'expression | Résultat |
|---|---|
| String | `string` |
| Movie clip | `movieclip` |
| Button | `object` |
| Text field | `object` |
| Number | `number` |
| Boolean | `boolean` |
| Object | `object` |
| Function | `function` |

### Voir également

`Opérateur instanceof`

## Opérateur void

`void` *`expression`*

L'opérateur `void` évalue une expression, puis supprime sa valeur, renvoyant `undefined`. L'opérateur `void` est souvent utilisé dans les comparaisons incluant l'opérateur == pour tester les valeurs non définies.

**Disponibilité :** Flash Player 5 ; ActionScript 1.0

### Opérandes

`expression : Object` - Expression à évaluer.

# Instructions

Les instructions sont des éléments de langage qui effectuent ou spécifient une action. Par exemple, l'instruction return renvoie un résultat sous forme de valeur de la fonction dans laquelle il s'exécute. L'instruction if évalue une condition pour déterminer l'action suivante à exécuter. L'instruction switch crée une structure arborescente pour les instructions ActionScript.

### Récapitulatif des instructions

| Instruction | Description |
|---|---|
| `break` | Apparaît au sein d'une boucle (`for` , `for..in`, `do..while` ou `while`) ou dans un bloc d'instructions associées à un cas donné au sein d'une instruction `switch`. |
| `case` | Définit une condition pour l'instruction `switch`. |
| `class` | Définit une classe personnalisée, ce qui permet de créer des occurrences des objets qui partagent les méthodes et les propriétés que vous définissez. |
| `continue` | Ignore toutes les instructions restantes dans la boucle imbriquée de plus bas niveau et passe à l'itération suivante, comme si le contrôle avait été transmis à la fin de la boucle normalement. |
| `default` | Définit le cas par défaut d'une instruction `switch`. |
| `delete` | Détruit la référence d'objet spécifiée par le paramètre *`reference`* et renvoie `true` si la référence est supprimée correctement ; `false` sinon. |

| Instruction | Description |
|---|---|
| `do..while` | Semblable à une boucle `while`, à la différence que les instructions sont exécutées une fois avant l'évaluation initiale de la condition. |
| `dynamic` | Spécifie que les objets basés sur la classe spécifiée peuvent ajouter des propriétés dynamiques et y accéder pendant l'exécution. |
| `else` | Spécifie les instructions à exécuter si la condition incluse dans l'instruction `if`renvoie `false`. |
| `else if` | Évalue une condition et spécifie les instructions à exécuter si la condition incluse dans l'instruction `if` initiale renvoie `false`. |
| `extends` | Définit une classe qui est une sous-classe d'une autre classe, cette dernière formant la superclasse. |
| `for` | Évalue l'expression *`init`* (initialiser) une fois, puis amorce une séquence de bouclage. |
| `for..in` | Répète en boucle les propriétés d'un objet ou d'éléments de tableau, puis exécute l'instruction *`statement`* pour chaque propriété ou élément. |
| `function` | Comprend un ensemble d'instructions que vous définissez pour effectuer une certaine tâche. |
| `get` | Autorise la *lecture* de propriétés associées aux objets sur la base des classes que vous avez définies dans les fichiers de classe externes. |
| `if` | Évalue une condition pour déterminer l'action suivante d'un fichier SWF. |
| `implements` | Spécifie qu'une classe doit définir toutes les méthodes déclarées dans l'interface (ou les interfaces) en cours d'implémentation. |
| `import` | Permet d'accéder aux classes sans spécifier leur nom complet, avec qualificatifs. |
| `interface` | Définit une interface. |
| `intrinsic` | Autorise la vérification des types lors de la compilation des classes définies précédemment. |
| `private` | Spécifie qu'une variable ou une fonction est disponible uniquement pour la classe qui la déclare ou la définit, ou pour les sous-classes de cette classe. |
| `public` | Spécifie qu'une variable ou une fonction est disponible à tout appelant. |
| `return` | Spécifie la valeur renvoyée par une fonction. |
| `set` | Autorise la définition implicite de propriétés associées aux objets sur la base des classes que vous avez définies dans les fichiers de classe externes. |
| `set variable` | Associe une valeur à une variable. |

| Instruction | Description |
|---|---|
| `static` | Spécifie qu'une variable ou une fonction n'est créée qu'une fois par classe et non pas créée dans chaque objet en fonction de cette classe. |
| `super` | Invoque la version superclass d'une méthode ou d'un constructeur. |
| `switch` | Crée une structure arborescente pour les instructions ActionScript. |
| `throw` | Génère ou *renvoie* une erreur qui peut être traitée ou *interceptée* par un bloc de code `catch{}`. |
| `try..catch..finally` | Entoure un bloc de code dans lequel une erreur peut se produire et être traitée. |
| `var` | Permet de déclarer des variables locales ou de scénario. |
| `while` | Évalue une condition. Si cette condition renvoie `true`, exécute une instruction ou une série d'instructions avant de suivre la boucle et d'évaluer de nouveau la condition. |
| `with` | Permet de spécifier un objet (tel qu'un clip) avec le paramètre *`object`* et évalue les expressions et les actions au sein de cet objet avec le paramètre *`statement(s)`*. |

# Instruction break

`break`

Apparaît au sein d'une boucle (`for` , `for..in`, `do..while` ou `while`) ou dans un bloc d'instructions associées à un cas donné au sein d'une instruction `switch`. Lorsqu'elle est utilisée dans une boucle, l'instruction `break` force Flash à ignorer le reste du corps de la boucle, arrête l'action de la boucle et exécute l'instruction suivant l'instruction de bouclage. Lors de l'utilisation dans le cadre d'une instruction `switch`, l'instruction `break` force Flash à ignorer le reste des instructions de ce bloc `case` et passe à la première instruction suivant l'instruction `switch` qui l'encadre.

Dans les boucles incorporées, l'instruction `break` ignore uniquement le reste de la boucle immédiate, sans sortir de la série de boucles incorporées. Pour sortir d'une série de boucles incorporées, voir `try..catch..finally`.

**Disponibilité :** Flash Player 4 ; ActionScript 1.0

## Exemple

L'exemple suivant utilise l'instruction `break` pour fermer une boucle sans fin :

```
var i:Number = 0;
while (true) {
 trace(i);
```

```
 if (i >= 10) {
 break; // this will terminate/exit the loop
 }
 i++;
}
```

ce qui permet de suivre les informations suivantes :

```
0
1
2
3
4
5
6
7
8
9
10
```

### Voir également

Instruction for

## Instruction case

```
case expression : statement(s)
```

Définit une condition pour l'instruction `switch`. Si le paramètre *`expression`* est égal au paramètre *`expression`* de l'instruction `switch` en appliquant l'égalité stricte (===), Flash Player exécute les instructions du paramètre *`statement(s)`* jusqu'à ce qu'il détecte une instruction `break` ou la fin d'une instruction `switch`.

Si vous utilisez l'instruction `case` en dehors d'une instruction `switch`, ceci produit une erreur et le script ne se compile pas.

*Remarque :* Vous devez toujours compléter le paramètre *`statement(s)`* par une instruction break. Si vous omettez `break statement` dans le paramètre *`statement(s)`* , l'exécution continue avec l'instruction `case` suivante au lieu de sortir de l'instruction `switch`.

**Disponibilité :** Flash Player 4 ; ActionScript 1.0

### Paramètres

*`expression`*`:String` - Toute expression.

### Exemple

L'exemple suivant définit les conditions de l'instruction switch `thisMonth`. Si `thisMonth` équivaut à l'expression de l'instruction case, l'instruction s'exécute.

```
var thisMonth:Number = new Date().getMonth();
switch (thisMonth) {
 case 0 :
 trace("January");
 break;
 case 1 :
 trace("February");
 break;
 case 5 :
 case 6 :
 case 7 :
 trace("Some summer month");
 break;
 case 8 :
 trace("September");
 break;
 default :
 trace("some other month");
}
```

### Voir également

`Instruction break`

## Instruction class

`[dynamic] class` *className* [ extends *superClass* ] [ implements *interfaceName*[, *interfaceName*`...` ] ] {

```
 // class definition here
}
```

Définit une classe personnalisée, ce qui permet de créer des occurrences des objets qui partagent les méthodes et les propriétés que vous définissez. Par exemple, si vous développez un système de suivi de factures, vous pouvez créer une classe invoice (facturation) qui définit toutes les méthodes et propriétés communes à l'ensemble des factures. Vous pouvez alors utiliser la commande `new invoice()` pour créer des objets facture.

Le nom de la classe doit correspondre au nom du fichier externe qui contient cette classe. Le nom du fichier externe doit être identique au nom de la classe auquel vient s'ajouter l'extension .as. Par exemple, si vous nommez une classe Stagiaire, le fichier qui définit la classe doit s'appeler Stagiaire.as.

Si une classe appartient à un package, la déclaration de classe doit utiliser le nom de classe entièrement qualifié de la forme base.sub1.sub2.MyClass. De même, le fichier AS de la classe doit être stocké avec son chemin dans une structure d'adresse reflétant la structure du package, telle que base/sub1/sub2/MyClass.as. Si une définition de classe est de forme « class MyClass », elle est dans le package par défaut et le fichier MyClass.as doit se trouver au niveau supérieur d'une adresse dans le chemin.

De ce fait, il est recommandé de planifier votre structure de répertoires avant de commencer la création de classes. En effet, si vous décidez de déplacer les fichiers de classe après leur création, vous devrez modifier les instructions de déclaration de classe pour indiquer leur nouvel emplacement.

Vous ne pouvez pas incorporer des définitions de classe. En d'autres termes, vous ne pouvez pas définir de classes supplémentaires dans une définition de classe.

Pour indiquer que des objets peuvent ajouter des propriétés dynamiques pendant la période d'exécution et y accéder, faites précéder l'instruction class par un mot-clé `dynamic`. Pour déclarer qu'une classe implémente une interface, utilisez le mot-clé `implements`. Pour créer des sous-classes d'une classe, utilisez le mot-clé `extends`. (Une classe ne peut étendre qu'une seule autre classe, mais peut implémenter plusieurs interfaces.) Vous pouvez utiliser `implements` et `extends` au sein d'une instruction unique. Les exemples suivants présentent des exemples type des mots-clés `implements` et `extends` :

```
class C implements Interface_i, Interface_j // OK
class C extends Class_d implements Interface_i, Interface_j // OK
class C extends Class_d, Class_e // not OK
```

**Disponibilité :** Flash Player 6 ; ActionScript 2,0

## Paramètres

*`className`*`:String` - Nom de la classe avec tous ses attributs.

## Exemple

L'exemple suivant crée une classe intitulée Plant. Le constructeur Plant réclame deux paramètres.

```
// Filename Plant.as
class Plant {
 // Define property names and types
 var leafType:String;
 var bloomSeason:String;
 // Following line is constructor
 // because it has the same name as the class
 function Plant(param_leafType:String, param_bloomSeason:String) {
 // Assign passed values to properties when new Plant object is created
 this.leafType = param_leafType;
```

```
 this.bloomSeason = param_bloomSeason;
 }
 // Create methods to return property values, because best practice
 // recommends against directly referencing a property of a class
 function getLeafType():String {
 return leafType;
 }
 function getBloomSeason():String {
 return bloomSeason;
 }
}
```

Dans un fichier de script externe ou dans le panneau Actions, utilisez l'opérateur `new` pour créer un objet Plant.

```
var pineTree:Plant = new Plant("Evergreen", "N/A");
// Confirm parameters were passed correctly
trace(pineTree.getLeafType());
trace(pineTree.getBloomSeason());
```

L'exemple suivant crée une classe intitulée ImageLoader. Le constructeur `ImageLoader` réclame trois paramètres.

```
// Filename ImageLoader.as
class ImageLoader extends MovieClip {
 function ImageLoader(image:String, target_mc:MovieClip, init:Object) {
 var listenerObject:Object = new Object();
 listenerObject.onLoadInit = function(target) {
 for (var i in init) {
 target[i] = init[i];
 }
 };
 var JPEG_mcl:MovieClipLoader = new MovieClipLoader();
 JPEG_mcl.addListener(listenerObject);
 JPEG_mcl.loadClip(image, target_mc);
 }
}
```

Dans un fichier de script externe ou dans le panneau Actions, utilisez l'opérateur `new` pour créer un objet ImageLoader.

```
var jakob_mc:MovieClip = this.createEmptyMovieClip("jakob_mc",
  this.getNextHighestDepth());
var jakob:ImageLoader = new ImageLoader("http://www.helpexamples.com/flash/
  images/image1.jpg", jakob_mc, {_x:10, _y:10, _alpha:70, _rotation:-5});
```

## Voir également

`Instruction dynamic`

# Instruction continue

`continue`

Ignore toutes les instructions restantes dans la boucle imbriquée de plus bas niveau et passe à l'itération suivante, comme si le contrôle avait été transmis à la fin de la boucle normalement. Elle n'a aucun effet en dehors d'une boucle.

**Disponibilité :** Flash Player 4 ; ActionScript 1.0

## Exemple

Dans la boucle `while` suivante, l'instruction `continue` force l'interpréteur Flash à ignorer le reste du corps de la boucle et à passer au début de la boucle, où la condition est testée :

```
trace("example 1");
var i:Number = 0;
while (i < 10) {
 if (i % 3 == 0) {
 i++;
 continue;
 }
 trace(i);
 i++;
}
```

Dans la boucle `do..while` suivante, l'instruction `continue` force l'interpréteur Flash à ignorer le reste du corps de la boucle et à passer au début de la boucle, où la condition est testée :

```
trace("example 2");
var i:Number = 0;
do {
 if (i % 3 == 0) {
 i++;
 continue;
 }
 trace(i);
 i++;
}
while (i < 10);
```

Dans une boucle `for`, l'instruction `continue` force l'interpréteur Flash à ignorer le reste du corps de la boucle. Dans l'exemple suivant, si le modulo i 3 est égal à 0, l'instruction `trace(i)` est ignorée :

```
trace("example 3");
for (var i = 0; i < 10; i++) {
 if (i % 3 == 0) {
 continue;
 }
```

```
 trace(i);
}
```

Dans la boucle `for..in` suivante, l'instruction `continue` force l'interpréteur Flash à ignorer le reste du corps de la boucle et à passer de nouveau au début de la boucle, où la valeur suivante de l'énumération est traitée :

```
for (i in _root) {
 if (i == "$version") {
 continue;
 }
 trace(i);
}
```

### Voir également

## Instruction default

`default: statements`

Définit le cas par défaut d'une instruction `switch`. Les instructions s'exécutent si le paramètre *`expression`* de l'instruction `switch` n'est pas égal (en appliquant l'opération d'égalité stricte [===]) à l'un des paramètres *`expression`* qui suivent les mots-clés `case` d'une instruction `switch` donnée.

L'instruction `switch` ne doit pas nécessairement inclure l'instruction case par `default`. L'instruction case par `default` ne doit pas nécessairement figurer en fin de liste. Si vous utilisez l'instruction `default` en dehors d'une instruction `switch`, ceci produit une erreur et le script ne se compile pas.

**Disponibilité :** Flash Player 6 ; ActionScript 1.0

### Paramètres

*`statements`*`:String` - Toute instruction.

### Exemple

Dans l'exemple suivant, l'expression A n'est pas égale aux expressions B ou D, donc l'instruction suivant le mot clé `default` est exécutée et l'instruction `trace()` est envoyée vers le panneau de sortie.

```
var dayOfWeek:Number = new Date().getDay();
switch (dayOfWeek) {
 case 1 :
 trace("Monday");
 break;
 case 2 :
```

```
 trace("Tuesday");
 break;
 case 3 :
 trace("Wednesday");
 break;
 case 4 :
 trace("Thursday");
 break;
 case 5 :
 trace("Friday");
 break;
 default :
 trace("Weekend");
}
```

### Voir également

`Instruction switch`

## Instruction delete

`delete` *`reference`*

Détruit la référence d'objet spécifiée par le paramètre *`reference`* et renvoie `true` si la référence est supprimée correctement ; `false` sinon. Cet opérateur permet de libérer la mémoire utilisée par les scripts. Vous pouvez utiliser l'opérateur `delete` pour supprimer des références à des objets. Une fois toutes les références à un objet supprimées, Flash Player supprime cet objet et libère la mémoire qu'il utilise.

Bien que `delete` soit un opérateur, il est généralement utilisé en tant qu'instruction, comme indiqué dans l'exemple suivant :

```
delete x;
```

L'opérateur `delete` peut échouer et renvoyer `false` si le paramètre *`reference`* n'existe pas ou ne peut pas être supprimé. L'instruction `var` ne vous permet pas de supprimer d'objets et de propriétés prédéfinis, ni de variables déclarées au sein d'une fonction. Vous ne pouvez pas utiliser l'opérateur `delete` pour supprimer des clips.

**Disponibilité :** Flash Player 5 ; ActionScript 1.0

### Valeur renvoyée

`Boolean` - Valeur booléenne.

### Paramètres

*`reference`*`:Object` - Nom de la variable ou de l'objet à éliminer.

## Exemple

Utilisation 1 : L'exemple suivant crée un objet, l'utilise, puis le supprime lorsqu'il n'est plus requis :

```
var account:Object = new Object();
account.name = "Jon";
account.balance = 10000;
trace(account.name); //output: Jon
delete account;
trace(account.name); //output: undefined
```

Utilisation 2 : L'exemple suivant supprime une propriété d'un objet :

```
// create the new object "account"
var account:Object = new Object();
// assign property name to the account
account.name = "Jon";
// delete the property
delete account.name;
```

Utilisation 3 : L'exemple suivant supprime une propriété d'objet :

```
var my_array:Array = new Array();
my_array[0] = "abc"; // my_array.length == 1
my_array[1] = "def"; // my_array.length == 2
my_array[2] = "ghi"; // my_array.length == 3
// my_array[2] is deleted, but Array.length is not changed
delete my_array[2];
trace(my_array.length); // output: 3
trace(my_array); // output: abc,def,undefined
```

Utilisation 4 : L'exemple suivant illustre le comportement de l'instruction `delete` sur des références à un objet :

```
var ref1:Object = new Object();
ref1.name = "Jody";
// copy the reference variable into a new variable
// and delete ref1
ref2 = ref1;
delete ref1;
trace("ref1.name "+ref1.name); //output: ref1.name undefined
trace("ref2.name "+ref2.name); //output: ref2.name Jody
```

Si `ref1` n'avait pas été copié dans `ref2`, l'objet aurait été supprimé au moment de la suppression de `ref1` car il ne contient aucune référence. Si vous supprimez `ref2`, il n'existe aucune référence à l'objet ; celui-ci sera détruit et la mémoire qu'il utilisait devient disponible.

## Voir également

`Instruction var`

# Instruction do..while

`do { statement(s) } while (condition)`

Semblable à une boucle `while`, à la différence que les instructions sont exécutées une fois avant l'évaluation initiale de la condition. Par conséquent, les instructions ne sont exécutées que si la condition renvoie `true`.

La boucle d`do..while` permet de s'assurer que le code de la boucle s'exécute au moins une fois. Bien que ceci puisse également se faire avec une boucle `while` en plaçant une copie des instructions à exécuter avant le début de la boucle `while`, de nombreux programmeurs trouvent les boucles `do..while` plus faciles à lire.

Si la condition renvoie toujours `true`, la boucle `do..while` est infinie. Si vous activez une boucle infinie, vous subirez des problèmes au niveau de Flash Player et recevrez un message d'avertissement, voire subirez un arrêt du lecteur. Dans la mesure du possible, utilisez une boucle `for` si vous connaissez le nombre de répétitions de la boucle. Bien que les boucles `for` soient plus faciles à lire et déboguer, elles ne sont pas totalement interchangeables avec les boucles `do..while`.

**Disponibilité :** Flash Player 4 ; ActionScript 1.0

## Paramètres

*`condition`*`:Boolean` - Condition à évaluer. Les instructions *`statement(s)`* à l'intérieur du bloc de code `do` sont exécutées tant que le paramètre *`condition`* renvoie `true` .

## Exemple

L'exemple suivant utilise une boucle `do..while` afin de déterminer si une condition a la valeur `true`, et suit `myVar` jusqu'à ce que la valeur de `myVar` soit supérieure à 5. Lorsque la valeur de `myVar` est supérieure à 5, la boucle se termine.

```
var myVar:Number = 0;
do {
 trace(myVar);
 myVar++;
}
while (myVar < 5);
/* output:
0
1
2
3
4
*/
```

### Voir également

`Instruction break`

## Instruction dynamic

```
dynamic class className [ extends superClass ] [ implements interfaceName[,
  interfaceName... ] ] {
 // class definition here
}
```

Spécifie que les objets basés sur la classe spécifiée peuvent ajouter des propriétés dynamiques et y accéder pendant l'exécution.

La vérification du type des classes dynamiques est moins stricte que pour les classes non dynamiques, dans la mesure où les membres sollicités au sein de la définition de classe et dans les occurrences de classe ne sont pas comparées à celles qui sont définies dans le domaine de la classe. Les fonctions des membres de la classe, cependant, peuvent toujours faire l'objet d'une vérification du type de renvoi ou de paramètre. Ce comportement est particulièrement utile lorsque vous travaillez avec des objets MovieClip, dans la mesure où il existe de nombreuses façons d'ajouter de façon dynamique des propriétés et des objets à un clip, telles que `MovieClip.createEmptyMovieClip()` et `MovieClip.createTextField()`.

Les sous-classes des classes dynamiques sont également des classes dynamiques.

Spécifiez bien le type lors de la déclaration d'un objet, comme ci-dessous :

```
var x:MyClass = new MyClass();
```

Si vous *ne spécifiez pas* le type lors de la déclaration d'un objet (comme ci-dessous), l'objet est alors considéré comme dynamique :

```
var x = new MyClass();
```

**Disponibilité :** Flash Player 6 ; ActionScript 2,0

### Exemple

Dans l'exemple suivant, la classe `Person2` n'a pas encore été définie comme étant dynamique ; par conséquent, l'appel d'une fonction non déclarée sur celle-ci génère une erreur lors de la compilation :

```
class Person2 {
 var name:String;
 var age:Number;
 function Person2(param_name:String, param_age:Number) {
 trace ("anything");
 this.name = param_name;
 this.age = param_age;
 }
}
```

Dans un fichier FLA ou AS qui se trouve dans le même répertoire, ajoutez le code ActionScript suivant à l'image 1 sur le scénario :

```
// Before dynamic is added
var craig:Person2 = new Person2("Craiggers", 32);
for (i in craig) {
 trace("craig." + i + " = " + craig[i]);
}
/* output:
craig.age = 32
craig.name = Craiggers */
```

Si vous ajoutez une fonction non déclarée, `dance`, une erreur est générée, comme indiqué dans l'exemple suivant :

```
trace("");
craig.dance = true;
for (i in craig) {
 trace("craig." + i + " = " + craig[i]);
}
/* output: **Error** Scene=Scene 1, layer=Layer 1, frame=1:Line 14: There is
   no property with the name 'dance'. craig.dance = true; Total ActionScript
   Errors: 1 Reported Errors: 1 */
```

Ajoutez le mot-clé `dynamic` à la classe Person2, de manière à ce que la première ligne s'affiche comme suit :

```
dynamic class Person2 {
```

Testez le code de nouveau ; vous obtenez le code suivant :

```
craig.dance = true craig.age = 32 craig.name = Craiggers
```

### Voir également

`Instruction class`

## Instruction else

```
if (condition){
statement(s);
} else {
statement(s);
}
```

Spécifie les instructions à exécuter si la condition incluse dans l'instruction `if`renvoie `false`. Les accolades (`{}`), qui servent normalement à entourer le bloc d'instructions que l'instruction `else` doit exécuter, peuvent être omises si une seule instruction doit s'exécuter.

**Disponibilité :** Flash Player 4 ; ActionScript 1.0

### Paramètres

*condition*:Boolean - Expression qui prend pour valeur `true` ou `false`.

### Exemple

Dans l'exemple suivant, la condition `else` est utilisée afin de vérifier si la variable `age_txt` est supérieure ou inférieure à 18 :

```
if (age_txt.text>=18) {
 trace("welcome, user");
}
else {
 trace("sorry, junior");
 userObject.minor = true;
 userObject.accessAllowed = false;
}
```

Dans l'exemple suivant, les accolades (`{}`) ne sont pas nécessaires car une seule instruction suit l'instruction `else` :

```
if (age_txt.text>18) { trace("welcome, user"); } else trace("sorry,
  junior");
```

### Voir également

`Instruction if`

## Instruction else if

`if(`*condition*`) {`
*statement(s)*;
`} else if(`*condition*`) {`
*statement(s)*;
`}`

Évalue une condition et spécifie les instructions à exécuter si la condition incluse dans l'instruction `if` initiale renvoie `false`. Lorsque la condition `else if` renvoie `true`, l'interpréteur Flash exécute les instructions qui suivent la condition entre accolades (`{}`). Si la condition `else if` est à `false`, Flash ignore les instructions entre accolades et exécute les instructions qui suivent ces accolades.

Utilisez l'instruction `else if` pour créer des arborescences logiques dans vos scripts. En présence de plusieurs branches, envisagez l'utilisation d'une instruction `switch`.

**Disponibilité :** Flash Player 4 ; ActionScript 1.0

### Paramètres

*condition*:Boolean - Expression qui prend pour valeur `true` ou `false`.

### Exemple

L'exemple suivant utilise des instructions `else if` pour comparer `score_txt` à une valeur spécifiée :

```
if (score_txt.text>90) {
 trace("A");
}
else if (score_txt.text>75) {
 trace("B");
}
else if (score_txt.text>60) {
 trace("C");
}
else {
 trace("F");
}
```

### Voir également

`Instruction if`

## Instruction extends

`class` *className* extends *otherClassName* `{}`
`interface` *interfaceName* extends *otherInterfaceName* `{}`

Définit une classe qui est une sous-classe d'une autre classe, cette dernière formant la superclasse. La sous-classe hérite de toutes les méthodes, propriétés, fonctions, etc. qui sont définies dans la superclasse.

Les interfaces peuvent également être développées avec le mot clé `extends`. Une interface qui développe une autre interface reprend toutes les déclarations de méthode de l'interface d'origine.

**Disponibilité :** Flash Player 6 ; ActionScript 2,0

### Paramètres

*`className`*`:String` - Nom de la classe en cours de définition.

### Exemple

Dans l'exemple suivant, la classe Car étend la classe Vehicle de manière à ce que toutes ses méthodes, propriétés et fonctions soient héritées. Si votre script instancie un objet Car, les méthodes de la classe Car et de la classe Vehicle peuvent être utilisées.

L'exemple suivant affiche le contenu d'un fichier intitulé Vehicle.as, qui définit la classe Vehicle :

```
class Vehicle {
 var numDoors:Number;
 var color:String;
 function Vehicle(param_numDoors:Number, param_color:String) {
 this.numDoors = param_numDoors;
 this.color = param_color;
 }
 function start():Void {
 trace("[Vehicle] start");
 }
 function stop():Void {
 trace("[Vehicle] stop");
 }
 function reverse():Void {
 trace("[Vehicle] reverse");
 }
}
```

L'exemple suivant affiche un deuxième fichier AS, intitulé Car.as, dans le même répertoire. Cette classe étend la classe Vehicle, la modifiant de trois façons. D'abord, la classe Car ajoute une variable `fullSizeSpare` afin de déterminer si, oui ou non, l'objet car est doté d'un pneu de secours de taille normale. Ensuite, elle ajoute une nouvelle méthode spécifique aux voitures, `activateCarAlarm()`, permettant d'activer l'alarme antivol de la voiture. Enfin, elle remplace la fonction `stop()` pour spécifier que la classe Car utilise un système de frein antiblocage pour s'arrêter.

```
class Car extends Vehicle {
 var fullSizeSpare:Boolean;
 function Car(param_numDoors:Number, param_color:String,
  param_fullSizeSpare:Boolean) {
 this.numDoors = param_numDoors;
 this.color = param_color;
 this.fullSizeSpare = param_fullSizeSpare;
 }
 function activateCarAlarm():Void {
 trace("[Car] activateCarAlarm");
 }
 function stop():Void {
 trace("[Car] stop with anti-lock brakes");
 }
}
```

L'exemple suivant instancie un objet Car, appelle une méthode définie dans la classe Vehicle (`start()`), puis celle remplacée par la classe Car (`stop()`) ; il appelle enfin une méthode de la classe Car (`activateCarAlarm()`) :

```
var myNewCar:Car = new Car(2, "Red", true);
myNewCar.start(); // output: [Vehicle] start
myNewCar.stop(); // output: [Car] stop with anti-lock brakes
myNewCar.activateCarAlarm(); // output: [Car] activateCarAlarm
```

Une sous-classe de la classe Vehicle peut également être écrite à l'aide du mot-clé `super` que la sous-classe peut utiliser pour accéder aux propriétés et méthodes de la superclasse. L'exemple suivant affiche un troisième fichier AS, intitulé Truck.as, une fois encore dans le même répertoire. La classe Truck utilise le mot-clé `super` dans le constructeur et, de nouveau, dans la fonction `reverse()` remplacée.

```
class Truck extends Vehicle {
 var numWheels:Number;
 function Truck(param_numDoors:Number, param_color:String,
  param_numWheels:Number) {
 super(param_numDoors, param_color);
 this.numWheels = param_numWheels;
 }
 function reverse():Void {
 beep();
 super.reverse();
 }
 function beep():Void {
 trace("[Truck] make beeping sound");
 }
}
```

L'exemple suivant instancie un objet Truck, appelle une méthode remplacée par la classe Truck (`reverse()`), puis une méthode définie dans la classe Vehicle (`stop()`) :

```
var myTruck:Truck = new Truck(2, "White", 18);
myTruck.reverse(); // output: [Truck] make beeping sound [Vehicle] reverse
myTruck.stop(); // output: [Vehicle] stop
```

### Voir également

`Instruction class`

## Instruction for

`for(`*init*; *condition*; *next*`) {`
*statement(s);*
`}`

Évalue l'expression *`init`* (initialiser) une fois, puis amorce une séquence de bouclage. La séquence de bouclage commence par évaluer l'expression *`condition`*. Si l'expression *`condition`* renvoie true, l'instruction *`statement`* s'exécute et l'expression *`next`* est évaluée. La séquence de bouclage reprend par l'évaluation de l'expression *`condition`*.

Les accolades (`{}`), qui servent normalement à entourer le bloc d'instructions que l'instruction `for` doit exécuter, peuvent être omises si une seule instruction doit s'exécuter.

**Disponibilité :** Flash Player 5 ; ActionScript 1.0

### Paramètres

*init* - Expression à évaluer avant d'amorcer la séquence de bouclage ; généralement une expression d'affectation. Ce paramètre autorise également une instruction `var`.

### Exemple

L'exemple suivant utilise l'instruction `for` pour ajouter les éléments dans un tableau :

```
var my_array:Array = new Array();
for (var i:Number = 0; i < 10; i++) {
 my_array[i] = (i + 5) * 10;
}
trace(my_array); // output: 50,60,70,80,90,100,110,120,130,140
```

L'exemple suivant utilise l'instruction `for` pour effectuer la même action à plusieurs reprises. Dans le code, la boucle `for` ajoute les nombres de 1 à 100.

```
var sum:Number = 0;
for (var i:Number = 1; i <= 100; i++) {
 sum += i;
}
trace(sum); // output: 5050
```

L'exemple suivant montre que les accolades ({}) ne sont pas nécessaires si une seule instruction s'exécute :

```
var sum:Number = 0;
for (var i:Number = 1; i <= 100; i++)
 sum += i;
trace(sum); // output: 5050
```

### Voir également

Opérateur ++ (incrément)

## Instruction for..in

`for (`*variableIterant* in *object*`) {`
*statement(s);*
`}`

Répète en boucle les propriétés d'un objet ou d'éléments de tableau, puis exécute l'instruction *statement* pour chaque propriété ou élément. Les méthodes d'un objet ne sont pas énumérées par l'action `for..in`.

Certaines propriétés ne peuvent pas être énumérées par l'action `for..in`. Par exemple, les propriétés de clip, telles que `_x` et `_y`, ne sont pas énumérées. Dans les fichiers de classe externes, les membres statiques ne peuvent pas être énumérés, contrairement aux membres d'occurrences.

L'instruction `for..in` itère sur des propriétés des objets du chaînage de prototype de l'objet itéré. Les propriétés de l'objet sont énumérés en premier, puis les propriétés de son prototype immédiat, puis les propriétés du prototype du prototype, etc. L'instruction `for..in` n'énumère pas le même nom de propriété deux fois. Si l'objet `child` comporte un prototype `parent` et que tous deux contiennent la propriété `prop`, l'instruction `for..in` appelée pour `child` énumère les propriétés `prop` de `child`, mais ignore celles de `parent`.

Les accolades (`{}`), qui servent normalement à entourer le bloc d'instructions que l'instruction `for..in` doit exécuter, peuvent être omises si une seule instruction doit s'exécuter.

Si vous écrivez une boucle `for..in` dans un fichier de classe (un fichier externe AS), les membres d'instance ne seront plus disponibles pour la boucle, contrairement aux membres statiques. Cependant, si vous écrivez une boucle `for..in` dans un fichier FLA pour une occurrence de la classe, les membres de l'occurrence restent disponibles, contrairement aux membres statiques.

**Disponibilité :** Flash Player 5 ; ActionScript 1.0

## Paramètres

*`variableIterant`*`:String` - Nom d'une variable devant servir d'itérant, référençant chaque propriété d'un objet ou d'un élément dans un tableau.

## Exemple

L'exemple suivant utilise une boucle `for..in` sur les propriétés d'un objet :

```
var myObject:Object = {firstName:"Tara", age:27, city:"San Francisco"};
for (var prop in myObject) {
 trace("myObject."+prop+" = "+myObject[prop]);
}
//output
myObject.firstName = Tara
myObject.age = 27
myObject.city = San Francisco
```

L'exemple suivant utilise une boucle `for..in` sur les éléments d'un tableau :

```
var myArray:Array = new Array("one", "two", "three");
for (var index in myArray)
 trace("myArray["+index+"] = " + myArray[index]);
// output:
myArray[2] = three
myArray[1] = two
myArray[0] = one
```

L'exemple suivant utilise l'opérateur `typeof` conjointement avec `for..in` pour faire une itération sur un type d'enfant particulier :

```
for (var name in this) {
```

```
 if (typeof (this[name]) == "movieclip") {
 trace("I have a movie clip child named "+name);
 }
}
```

*Remarque :* Si vous disposez de plusieurs clips, le code obtenu inclut leurs noms d'occurrence.

L'exemple suivant énumère les enfants d'un clip et les envoie à l'image 2 de leurs scénarios respectifs. Le clip `RadioButtonGroup` est un parent ayant trois enfants : `_RedRadioButton_`, `_GreenRadioButton_`, et `_BlueRadioButton_`.

```
for (var name in RadioButtonGroup) { RadioButtonGroup[name].gotoAndStop(2);
  }
```

## Instruction function

```
Usage 1: (Declares a named function.)
function functionname([parameter0, parameter1,...parameterN]){
statement(s)
}
Usage 2: (Declares an anonymous function and returns a reference to it.)
function ([parameter0, parameter1,...parameterN]){
statement(s)
}
```

Comprend un ensemble d'instructions que vous définissez pour effectuer une certaine tâche. Vous pouvez définir une fonction à un emplacement et l'*appeler* à partir de différents scripts dans un fichier SWF. Lorsque vous définissez une fonction, vous pouvez également spécifier des paramètres pour la fonction. Les paramètres sont des espaces réservés pour les valeurs sur lesquelles la fonction opère. Vous pouvez passer différents paramètres à une fonction lors de chaque appel, de façon à pouvoir utiliser une fonction dans différentes situations.

Utilisez l'instruction `return` dans le paramètre `statement(s)` d'une fonction pour que cette dernière génère ou *renvoie* une valeur.

Vous pouvez utiliser cette instruction pour définir une `function` ayant les paramètres spécifiés `functionname`, `parameters`, et `statement(s)`. Lorsqu'un script appelle une fonction, les instructions figurant dans la définition de la fonction s'exécute. Les références anticipées sont autorisées. Dans un script, une fonction peut être déclarée après son appel. Une définition de fonction remplace toute définition précédente de la même fonction. Vous pouvez utiliser cette syntaxe dans toutes les circonstances où une instruction est autorisée.

Vous pouvez également utiliser cette instruction pour créer une fonction anonyme et lui renvoyer une référence. Cette syntaxe est utilisée dans des expressions et est particulièrement utile pour l'installation des méthodes dans les objets.

Pour bénéficier de fonctionnalités supplémentaires, vous pouvez utiliser l'objet `arguments` dans votre définition de fonction. Certaines utilisations communes de l'objet `arguments` créent une fonction qui accepte un nombre variable de paramètres et créent une fonction anonyme récursive.

**Disponibilité :** Flash Player 5 ; ActionScript 1.0

### Valeur renvoyée

`String` : Utilisation 1 : Le formulaire de déclaration ne doit rien renvoyer. Utilisation 2 : référence à la fonction anonyme.

### Paramètres

*`functionname`*`:String` - Nom de la fonction déclarée.

### Exemple

L'exemple suivant définit la fonction `sqr` qui accepte un paramètre et renvoie la valeur `Math.pow(x, 2)` du paramètre :

```
function sqr(x:Number) {
 return Math.pow(x, 2);
}
var y:Number = sqr(3);
trace(y); // output: 9
```

Si la fonction est définie et utilisée dans le même script, la définition de fonction peut apparaître lorsque vous l'avez utilisée :

```
var y:Number = sqr(3);
trace(y); // output: 9
function sqr(x:Number) {
 return Math.pow(x, 2);
}
```

La fonction suivante crée un objet LoadVars et charge params.txt dans le fichier SWF. Si le chargement du fichier réussit, `variables loaded` est renvoyé :

```
var myLV:LoadVars = new LoadVars();
myLV.load("params.txt");
myLV.onLoad = function(success:Boolean) {
 trace("variables loaded");
}
```

## Instruction get

```
function get property () {
 // your statements here
}
```

Autorise la *lecture* de propriétés associées aux objets sur la base des classes que vous avez définies dans les fichiers de classe externes. L'utilisation de méthodes get implicites permet d'accéder aux propriétés des objets sans accéder à la propriété directement. Les méthodes get/set implicites sont des abréviations syntaxiques de la méthode `Object.addProperty()` dans ActionScript 1.0.

**Disponibilité :** Flash Player 6 ; ActionScript 2,0

## Paramètres

*`property`*`:String` - Mot que vous utilisez pour faire référence à la propriété qui obtient l'accès ; cette valeur doit être identique à la valeur utilisée dans la commande `set` correspondante.

## Exemple

Dans l'exemple suivant, vous définissez une classe Team. La classe Team inclut les méthodes get/set qui vous permettent de récupérer et de définir les propriétés au sein de la classe :

```
class Team {
 var teamName:String;
 var teamCode:String;
 var teamPlayers:Array = new Array();
 function Team(param_name:String, param_code:String) {
 this.teamName = param_name;
 this.teamCode = param_code;
 }
 function get name():String {
 return this.teamName;
 }
 function set name(param_name:String):Void {
 this.teamName = param_name;
 }
}
```

Entrez le code ActionScript suivant dans une image du scénario :

```
var giants:Team = new Team("San Fran", "SFO");
trace(giants.name);
giants.name = "San Francisco";
trace(giants.name);
/* output:
San Fran San Francisco */
```

Lorsque vous appliquez une instruction trace à giants.name, vous utilisez la méthode `get` pour renvoyer la valeur de la propriété.

## Voir également

`addProperty (méthode Object.addProperty)`

# Instruction if

```
if(condition) {
statement(s);
}
```

Évalue une condition pour déterminer l'action suivante d'un fichier SWF. Lorsque cette condition est `true`, Flash exécute les instructions qui suivent la condition entre accolades (`{}`). Si la condition est `false`, Flash ignore les instructions entre accolades et exécute les instructions qui suivent ces accolades. Utilisez l'instruction `if` en conjonction avec les instructions `else` et `else if` pour introduire une arborescence logique dans vos scripts.

Les accolades (`{}`), qui servent normalement à entourer le bloc d'instructions que l'instruction `if` doit exécuter, peuvent être omises si une seule instruction doit s'exécuter.

**Disponibilité :** Flash Player 4 ; ActionScript 1.0

## Paramètres

*condition*:`Boolean` - Expression qui prend pour valeur `true` ou `false`.

## Exemple

Dans l'exemple suivant, la condition placée entre parenthèses évalue le nom `name` de la variable afin de déterminer s'il a la valeur littérale `"Erica"`. Si tel est le cas, la fonction `play()` placée entre accolades s'exécute.

```
if(name == "Erica"){
 play();
}
```

L'exemple suivant utilise une instruction `if` pour évaluer le temps nécessaire à un utilisateur pour cliquer sur l'occurrence `submit_btn` d'un fichier SWF. Si l'utilisateur clique sur le bouton plus de 10 secondes après le début de la lecture du fichier SWF, la condition renvoie `true` et le message placé entre accolades (`{}`) apparaît dans un champ de texte créé lors de l'exécution (via `createTextField()`). Si l'utilisateur clique sur le bouton moins de 10 secondes après le début de la lecture du fichier SWF, la condition renvoie `false` et un message différent apparaît.

```
this.createTextField("message_txt", this.getNextHighestDepth, 0, 0, 100,
  22);
message_txt.autoSize = true;
var startTime:Number = getTimer();
this.submit_btn.onRelease = function() {
 var difference:Number = (getTimer() - startTime) / 1000;
 if (difference > 10) {
 this._parent.message_txt.text = "Not very speedy, you took "+difference+"
  seconds.";
 }
```

```
 else {
 this._parent.message_txt.text = "Very good, you hit the button in
  "+difference+" seconds.";
 }
};
```

### Voir également

`Instruction else`

## Instruction implements

*myClass* implements *interface01* [, *interface02* , ...]

Spécifie qu'une classe doit définir toutes les méthodes déclarées dans l'interface (ou les interfaces) en cours d'implémentation.

**Disponibilité :** Flash Player 6 ; ActionScript 2.0

### Exemple

Voir la section`interface`.

### Voir également

`Instruction class`

## Instruction import

`import` *className*
`import` *packageName*`.*`

Permet d'accéder aux classes sans spécifier leur nom complet, avec qualificatifs. Par exemple, si vous souhaitez utiliser une classe personnalisée, telle que macr.util.users.UserClass, dans un script, vous devez y faire référence avec son nom suivi de tous ses attributs ou l'importer. Si vous l'importez, vous pouvez y faire référence avec le nom de classe :

```
// before importing
var myUser:macr.util.users.UserClass = new macr.util.users.UserClass();
// after importing
import macr.util.users.UserClass;
var myUser:UserClass = new UserClass();
```

Lorsque le package contient plusieurs fichiers de classe (*working_directory* `/macr/utils/users`) auxquels vous devez accéder, vous pouvez les importer tous dans une instruction unique, comme indiqué dans l'exemple suivant :

```
import macr.util.users.*;
```

Vous devez émettre l'instruction `import` avant de tenter d'accéder à la classe importée sans spécifier l'ensemble du nom.

Si vous importez une classe, mais ne l'utilisez pas dans votre script, cette dernière n'est pas exportée avec le fichier SWF. Ceci signifie que vous pouvez importer des packages volumineux sans vous soucier de la taille du fichier SWF. Le pseudo-code binaire associé à une classe n'est inclus dans un fichier SWF que si cette classe est véritablement utilisée.

L'instruction `import` s'applique uniquement au script courant (image ou objet) dans lequel elle est appelée. Par exemple, supposons que vous deviez importer l'ensemble des classes du package macr.util dans l'image 1 d'un document Flash. Dans cette image, vous pouvez faire référence aux classes de ce package par leur nom simple :

```
// On Frame 1 of a FLA:
import macr.util.*;
var myFoo:foo = new foo();
```

Dans un autre script d'image, cependant, vous devez faire référence aux classes de ce package par leur nom suivi de tous leurs attributs (`var myFoo:foo = new macr.util.foo();`) ou ajouter une instruction `import` à l'image qui importe les classes dans ce package.

**Disponibilité :** Flash Player 6 ; ActionScript 2.0

### Paramètres

*`className`*`:String` - Nom qualifié d'une classe définie dans un fichier de classe externe.

### Exemple

## Instruction interface

`interface` *InterfaceName* [extends *InterfaceName* `] {}`

Définit une interface. Une interface est similaire à une classe. Les différences fondamentales sont regroupées ci-dessous :

- Les interfaces contiennent uniquement les déclarations des méthodes, pas leur implémentation. Ainsi, toute classe qui implémente une interface doit fournir une implémentation pour chaque méthode déclarée dans l'interface.
- Seuls les membres publics sont autorisés dans la définition d'une interface. Les instances et les membres de classe ne sont pas permis.
- Les instructions `get` et `set` ne sont pas autorisées dans les définitions d'interface.

**Disponibilité :** Flash Player 6 ; ActionScript 2.0

## Exemple

L'exemple suivant présente plusieurs façons de définir et d'implémenter des interfaces :

```
(in top-level package .as files Ia, B, C, Ib, D, Ic, E)
// filename Ia.as
interface Ia {
 function k():Number; // method declaration only
 function n(x:Number):Number; // without implementation
}
// filename B.as
class B implements Ia {
 function k():Number {
 return 25;
 }
 function n(x:Number):Number {
 return x + 5;
 }
} // external script or Actions panel // script file
var mvar:B = new B();
trace(mvar.k()); // 25
trace(mvar.n(7)); // 12
// filename c.as
class C implements Ia {
 function k():Number {
 return 25;
 }
} // error: class must implement all interface methods
// filename Ib.as
interface Ib {
 function o():Void;
}
class D implements Ia, Ib {
 function k():Number {
 return 15;
 }
 function n(x:Number):Number {
 return x * x;
 }
 function o():Void {
 trace("o");
 }
} // external script or Actions panel // script file
mvar = new D();
trace(mvar.k()); // 15
trace(mvar.n(7)); // 49
trace(mvar.o()); // "o"
interface Ic extends Ia {
 function p():Void;
}
class E implements Ib, Ic {
```

```
 function k():Number {
 return 25;
 }
 function n(x:Number):Number {
 return x + 5;
 }
 function o():Void {
 trace("o");
 }
 function p():Void {
 trace("p");
 }
}
```

### Voir également

`Instruction class`

## instruction intrinsic

```
intrinsic class className [extends superClass] [implements interfaceName [,
  interfaceName...] ] {
 //class definition here
}
```

Autorise la vérification des types lors de la compilation des classes définies précédemment. Flash utilise des déclarations de classe intrinsèques pour permettre la vérification des types de classes intégrées tels que `Array`, `Object`, et `String` lors de la compilation. Ce mot-clé indique au compilateur qu'aucune implémentation de fonction n'est requise et qu'il n'est pas nécessaire de générer un pseudo-code binaire pour celle-ci.

Le mot-clé `intrinsic` peut également être utilisé conjointement avec des déclarations de variable et de fonction. Flash utilise ce mot-clé pour permettre la vérification des types des fonctions et des propriétés globales lors de la compilation.

Le mot-clé `intrinsic` a été spécialement créé pour permettre la vérification des types de classes et objets intégrés, ainsi que des variables et des fonctions lors de la compilation. Ce mot-clé n'est pas destiné à un usage général mais peut s'avérer utile pour les développeurs qui cherchent à autoriser la vérification des types lors de la compilation à l'aide de classes définies précédemment, notamment ci celles-ci sont définies via ActionScript 1.0.

Ce mot-clé n'est pris en charge que lorsqu'il est utilisé dans des fichiers de script externes, et non pas dans les scripts écrits dans le panneau Actions.

**Disponibilité :** Flash Player 6 ; ActionScript 2.0

## Exemple

L'exemple suivant indique comment activer la vérification de fichiers lors de la compilation pour une classe ActionScript 1.0 définie précédemment. Le code génère une erreur de compilation car l'appel `myCircle.setRadius()` envoie une valeur de type `String` en tant que paramètre au lieu d'une valeur de type `Number`. Vous pouvez éviter cette erreur en modifiant le paramètre pour le définir sur une valeur de type `Number` (par exemple, en changeant `"10"` par `10`).

```
// The following code must be placed in a file named Circle.as
// that resides within your classpath:
intrinsic class Circle {
 var radius:Number;
 function Circle(radius:Number);
 function getArea():Number;
 function getDiameter():Number;
 function setRadius(param_radius:Number):Number;
}

// This ActionScript 1.0 class definition may be placed in your FLA file.
// Circle class is defined using ActionScript 1.0
function Circle(radius) {
 this.radius = radius;
 this.getArea = function(){
 return Math.PI*this.radius*this.radius;
 };
 this.getDiameter = function() {
 return 2*this.radius;
 };
 this.setRadius = function(param_radius) {
 this.radius = param_radius;
 }
}

// ActionScript 2.0 code that uses the Circle class
var myCircle:Circle = new Circle(5);
trace(myCircle.getArea());
trace(myCircle.getDiameter());
myCircle.setRadius("10");
trace(myCircle.radius);
trace(myCircle.getArea());
trace(myCircle.getDiameter());
```

## Voir également

`Instruction class`

# Instruction private

```
class someClassName{
 private var name;
 private function name() {
 // your statements here
 }
}
```

Spécifie qu'une variable ou une fonction est disponible uniquement pour la classe qui la déclare ou la définit, ou pour les sous-classes de cette classe. Par défaut, une variable ou une fonction est disponible à tout appelant. Utilisez ce mot-clé si vous devez restreindre l'accès à une variable ou une fonction. Ce mot-clé est voulu comme une aide au développement du logiciel afin de faciliter de bonnes méthodes de codage telles que l'encapsulation, et non comme un mécanisme de sécurité permettant de dissimuler ou sécuriser les ddonnées sensibles. Il n'empêche pas obligatoirement l'accès à une variable lors de l'exécution.

Ce mot-clé est réservé aux définitions de classe et ne permet pas de créer des définitions d'interface.

**Disponibilité :** Flash Player 6 ; ActionScript 2.0

## Paramètres

*name*:String - Nom de la variable ou de la fonction à spécifier en tant que privée.

## Exemple

L'exemple suivant montre comment restreindre l'accès aux variables ou fonctions par l'utilisation du mot-clé `private`. Créez un nouveau fichier AS intitulé Alpha.as :

```
class Alpha {
 private var privateProperty = "visible only within class and subclasses";
 public var publicProperty = "visible everywhere";
}
```

Dans le même répertoire qu'Alpha.as, créez un nouveau fichier AS nommé Beta.as qui contiendra le code suivant :

```
class Beta extends Alpha {
 function Beta() {
 trace("privateProperty is " + privateProperty);
 }
}
```

Comme le montre le code suivant, le constructeur de la classe Beta peut accéder à la propriété `privateProperty` qui est héritée de la classe Alpha :

```
var myBeta:Beta = new Beta(); // Output: privateProperty is visible only
  within class and subclasses
```

Des essais pour accéder à la variable `privateProperty` en dehors de la classe Alpha ou d'une classe héritant de la classe Alpha entraînent une erreur. Le code suivant, qui est hors de toute classe, entraîne une erreur :

```
trace(myBeta.privateProperty); // Error
```

### Voir également

`Instruction public`

## Instruction public

```
class someClassName{
 public var name;
 public function name() {
 // your statements here
 }
}
```

Spécifie qu'une variable ou une fonction est disponible à tout appelant. Dans la mesure où les variables et les fonctions sont publiques par défaut, ce mot-clé est utilisé surtout pour des raisons de style. Par exemple, vous pouvez l'utiliser pour des raisons de cohérence dans un bloc de code qui contient également des variables privées ou statiques.

**Disponibilité :** Flash Player 6 ; ActionScript 2.0

### Paramètres

*name*`:String` - Nom de la variable ou de la fonction à spécifier en tant que publique.

### Exemple

L'exemple suivant indique comment utiliser des variables publiques dans un fichier de classe. Créez un nouveau fichier de classe intitulé User.as et entrez le code suivant :

```
class User {
 public var age:Number;
 public var name:String;
}
```

Créez ensuite un nouveau fichier FLA ou AS dans le même répertoire, puis entrez le code ActionScript suivant dans l'image 1 du scénario :

```
import User;
var jimmy:User = new User();
jimmy.age = 27;
jimmy.name = "jimmy";
```

Si vous convertissez l'une des variables publiques de la classe User en variable privée, une erreur est générée lorsque vous tentez d'accéder à la propriété.

### Voir également

`Instruction private`

## return, instruction

`return[`*`expression`*`]`

Spécifie la valeur renvoyée par une fonction. L'instruction `return` évalue *`expression`* et renvoie un résultat sous forme de valeur de la fonction dans laquelle elle s'exécute. L'instruction `return` transfère immédiatement l'exécution à la fonction appelante. Si l'instruction `return` est utilisée seule, elle renvoie `undefined.` (non défini).

Vous ne pouvez pas renvoyer des valeurs multiples. En effet, seule la dernière valeur est renvoyée. Dans l'exemple suivant, la valeur `c` est renvoyée :

```
return a, b, c ;
```

Si vous devez renvoyer des valeurs multiples, utilisez un tableau ou un objet.

**Disponibilité :** Flash Player 5 ; ActionScript 1.0

### Valeur renvoyée

`String` - Paramètre *`expression`* évalué, si disponible.

### Paramètres

*`expression`* - Chaîne, nombre, valeur booléenne, tableau ou objet à évaluer et renvoyer sous forme de valeur de la fonction. Ce paramètre est facultatif.

### Exemple

L'exemple suivant utilise l'instruction `return` qui figure dans le corps de la fonction `sum()` pour renvoyer la valeur ajoutée des trois paramètres. La ligne de code suivante appelle `sum()` et affecte la valeur renvoyée à la variable `newValue`.

```
function sum(a:Number, b:Number, c:Number):Number {
 return (a + b + c);
}
var newValue:Number = sum(4, 32, 78);
trace(newValue); // output: 114
```

### Voir également

`Instruction function`

# Instruction set

```
function set property(varName) {
 // your statements here
}
```

Autorise la définition implicite de propriétés associées aux objets sur la base des classes que vous avez définies dans les fichiers de classe externes. L'utilisation de méthodes set implicites permet de modifier la valeur de la propriété d'un objet sans accéder directement à cette propriété. Les méthodes get/set implicites sont des abréviations syntaxiques de la méthode `Object.addProperty()` dans ActionScript 1.0.

**Disponibilité :** Flash Player 6 ; ActionScript 2.0

## Paramètres

*property*:`String` - Mot faisant référence à la propriété cible de `set` ; cette valeur doit être identique à la valeur utilisée par la commande `get` correspondante.

## Exemple

L'exemple suivant crée une classe Login qui montre comment utiliser le mot-clé `set` pour définir des variables privées :

```
class Login {
 private var loginUserName:String;
 private var loginPassword:String;
 public function Login(param_username:String, param_password:String) {
 this.loginUserName = param_username;
 this.loginPassword = param_password;
 }
 public function get username():String {
 return this.loginUserName;
 }
 public function set username(param_username:String):Void {
 this.loginUserName = param_username;
 }
 public function set password(param_password:String):Void {
 this.loginPassword = param_password;
 }
}
```

Dans un fichier FLA ou AS qui se trouve dans le même répertoire que le fichier Login.as, entrez le code ActionScript suivant dans l'image 1 du scénario :

```
var gus:Login = new Login("Gus", "Smith");
trace(gus.username); // output: Gus
gus.username = "Rupert";
trace(gus.username); // output: Rupert
```

Dans l'exemple suivant, la fonction `get` s'exécute lorsque la valeur est tracée. La fonction `set` se déclenche uniquement lorsque vous lui transmettez une valeur, comme indiqué sur la ligne :

```
gus.username = "Rupert";
```

### Voir également

`Instruction get`

## Instruction set variable

`set("variableString",expression)`

Associe une valeur à une variable. Une *variable* est un conteneur qui stocke des données. Le conteneur reste toujours le même, c'est le contenu qui peut varier. La modification de la valeur d'une variable pendant la lecture du fichier SWF permet d'enregistrer les informations relatives aux actions de l'utilisateur, d'enregistrer les valeurs modifiées pendant la lecture du fichier SWF ou d'évaluer si une condition est `true` ou `false`.

Les variables peuvent recouvrir tous les types de données, tels que String, Number, Boolean, Object ou MovieClip. Le scénario de chaque fichier SWF et clip comporte son propre jeu de variables, et chaque variable dispose de sa propre valeur, indépendamment des variables des autres scénarios.

Le typage strict des données n'est pas pris en charge dans une instruction `set`. Si vous utilisez cette instruction pour définir une variable sur une valeur dont le type de données diffère du type associé à cette variable dans un fichier de classe, aucune erreur de compilation n'est générée.

Il est important de noter que le paramètre *`variableString`* est une chaîne et non pas un nom de variable. Si vous transmettez une variable existante en tant que premier paramètre à `set()` sans le placer entre guillemets (""), la variable est évaluée avant que la valeur d'*`expression`* ne lui soit affectée. Par exemple, si vous créez une variable de type chaîne appelée `myVariable` et lui affectez la valeur « Tuesday » sans mettre cette dernière entre guillemets, vous créez une nouvelle variable appelée `Tuesday` et contenant la valeur normalement destinée à `myVariable` :

```
var myVariable:String = "Tuesday";
set (myVariable, "Saturday");
trace(myVariable); // outputs Tuesday
trace(Tuesday); // outputs Saturday
```

Pour remédier à cette situation, incluez les guillemets ("") :

```
set ("myVariable", "Saturday");
trace(myVariable); //outputs Saturday
```

**Disponibilité :** Flash Player 4 ; ActionScript 1.0

### Paramètres

*variableString*:String - Chaîne nommant la variable devant contenir la valeur du paramètre *expression*.

### Exemple

Dans l'exemple suivant, vous affectez une valeur à une variable. Vous affectez la valeur "Jakob" à la variable name.

```
set("name", "Jakob");
trace(name);
```

Le code suivant boucle à trois reprises et crée trois nouvelles variables intitulées caption0, caption1, et caption2 :

```
for (var i = 0; i < 3; i++) {
 set("caption" + i, "this is caption " + i);
}
trace(caption0);
trace(caption1);
trace(caption2);
```

### Voir également

Instruction var

## Instruction static

```
class someClassName{
 static var name;
 static function name() {
 // your statements here
 }
}
```

Spécifie qu'une variable ou une fonction n'est créée qu'une fois par classe et non pas créée dans chaque objet en fonction de cette classe.

Vous pouvez accéder à un membre de classe statique sans créer une occurrence de sa classe en utilisant la syntaxe someClassName.name. Si vous créez une occurence de la classe, vous pouvez également accéder à un membre statique en utilisant l'occurence, mais uniquement par le biais d'une fonction non statique qui accéde au membre statique.

Ce mot-clé est réservé aux définitions de classe et ne permet pas de créer des définitions d'interface.

**Disponibilité :** Flash Player 6 ; ActionScript 2.0

### Paramètres

*name*:String - Nom de la variable ou de la fonction à spécifier en tant que statique.

### Exemple

L'exemple suivant présente l'utilisation du mot-clé `static` pour créer un compteur chargé de suivre le nombre d'occurrences de la classe créées. La variable `numInstances` étant statique, elle ne sera créée qu'une fois pour l'ensemble de la classe, pas pour chaque occurrence. Créez un nouveau fichier AS intitulé Users.as et entrez le code suivant :

```
class Users {
 private static var numInstances:Number = 0;
 function Users() {
 numInstances++;
 }
 static function get instances():Number {
 return numInstances;
 }
}
```

Créez un document FLA ou AS dans le même répertoire, puis entrez le code ActionScript suivant dans l'image 1 du scénario :

```
trace(Users.instances);
var user1:Users = new Users();
trace(Users.instances);
var user2:Users = new Users();
trace(Users.instances);
```

### Voir également

`Instruction private`

## Instruction super

`super.`*method*`([`*arg1, ..., argN*`])`
`super([`*arg1, ..., argN*`])`

Le premier style de syntaxe peut être utilisé dans le corps d'une méthode d'objet pour appeler la version superclass d'une méthode et peut transmettre des paramètres en option `(arg1 ... argN)` à la méthode superclass. Cet opérateur permet non seulement de créer des méthodes de sous-classe qui ajoutent des comportements supplémentaires aux méthodes superclass, mais encore d'exécuter leur comportement d'origine.

Le deuxième style de syntaxe peut s'utiliser dans le corps d'une fonction constructeur pour appeler la version superclass de cette fonction et peut lui transférer des paramètres en option. Ceci permet non seulement de créer une sous-classe qui procède à une initialisation supplémentaire, mais encore d'appeler la fonction constructeur superclass pour initialiser la superclasse.

**Disponibilité :** Flash Player 6 ; ActionScript 1.0

### Valeur renvoyée

Les deux formes appellent une fonction. Cette fonction peut renvoyer toutes sortes de valeur.

### Paramètres

*method*:Function - Méthode à appeler dans la superclasse.

*argN* - Paramètres facultatifs qui sont transmis à la version superclass de la méthode (syntaxe 1) ou à la fonction constructeur de la superclasse (syntaxe 2).

## Instruction switch

```
switch (expression){
caseClause:
 [defaultClause:]
}
```

Crée une structure arborescente pour les instructions ActionScript. Comme pour l'instruction `if`, l'instruction `switch` teste une condition et exécute des instructions si cette condition renvoie la valeur `true`. Toutes les instructions switch doivent inclure un cas par défaut. Ce cas doit inclure une instruction break pour prévenir les erreurs fall-through en cas d'ajout d'un autre cas. Lorsqu'un cas subit une erreur fall-through, il ne comporte pas d'instruction break.

**Disponibilité :** Flash Player 4 ; ActionScript 1.0

### Paramètres

*expression* - Toute expression.

### Exemple

Dans l'exemple suivant, si le paramètre `String.fromCharCode(Key.getAscii())` évalue `A`, l'instruction `trace()` qui suit `case "A"` s'exécute; si le paramètre évalue `a`, l'instruction `trace()` qui suit `case "a"` s'exécute; etc. Si aucune expression `case` ne correspond au paramètre `String.fromCharCode(Key.getAscii())`, l'instruction `trace()` suivant le mot-clé `default` s'exécute.

```
var listenerObj:Object = new Object();
```

```
listenerObj.onKeyDown = function() {
 switch (String.fromCharCode(Key.getAscii())) {
 case "A" :
 trace("you pressed A");
 break;
 case "a" :
 trace("you pressed a");
 break;
 case "E" :
 case "e" :
 trace("you pressed E or e");
 break;
 case "I" :
 case "i" :
 trace("you pressed I or i");
 break;
 default :
 trace("you pressed some other key");
 break;
 }
};
Key.addListener(listenerObj);
```

### Voir également

Opérateur === (égalité stricte)

## Instruction throw

```
throw expression
```

Génère ou *renvoie* une erreur qui peut être traitée ou *interceptée* par un bloc de code `catch{}`. Si aucune exception n'est interceptée par le bloc `catch`, la chaîne représentant la valeur renvoyée s'affiche dans le panneau de sortie.

De manière générale, le système renvoie des occurrences de la classe Error ou de ses sous-classes (voir la section Exemple).

**Disponibilité :** Flash Player 7 ; ActionScript 1.0

### Paramètres

*expression*`:Object` - Expression ou objet ActionScript.

### Exemple

Dans cet exemple, une fonction intitulée `checkEmail()` vérifie si la chaîne qui lui est transmise est une adresse électronique correctement formatée. Si la chaîne ne contient pas le symbole @, la fonction renvoie une erreur.

```
function checkEmail(email:String) {
 if (email.indexOf("@") == -1) {
 throw new Error("Invalid email address");
 }
}
checkEmail("someuser_theirdomain.com");
```

Le code suivant appelle ensuite la fonction `checkEmail()` dans un bloc de code `try`. Si la chaîne `email_txt` ne contient pas une adresse de messagerie valide, le message d'erreur apparaît dans un champ de texte (`error_txt`).

```
try {
 checkEmail("Joe Smith");
}
catch (e) {
 error_txt.text = e.toString();
}
```

Dans l'exemple suivant, une sous-classe de la classe Error est renvoyée. La fonction `checkEmail()` est modifiée pour renvoyer une occurrence de cette sous-classe.

```
// Define Error subclass InvalidEmailError // In InvalidEmailError.as:
  class InvalidEmailAddress extends Error { var message = "Invalid email
  address."; }
```

Dans un fichier FLA ou AS, entrez le code ActionScript suivant dans l'image 1 du scénario :

```
import InvalidEmailAddress;
function checkEmail(email:String) {
 if (email.indexOf("@") == -1) {
 throw new InvalidEmailAddress();
 }
}
try {
 checkEmail("Joe Smith");
}
catch (e) {
 this.createTextField("error_txt", this.getNextHighestDepth(), 0, 0, 100,
  22);
 error_txt.autoSize = true;
 error_txt.text = e.toString();
}
```

### Voir également

`Erreur`

## Instruction try..catch..finally

```
try {
// ... try block ...
} finally {
```

```
// ... finally block ...
}
try {
// ... try block ...
} catch(error [:ErrorType1]) {
// ... catch block ...
} [catch(error[:ErrorTypeN]) {
// ... catch block ...
}] [finally {
// ... finally block ...
}]
```

Entoure un bloc de code dans lequel une erreur peut se produire et être traitée. Si du code figurant dans le bloc `try` renvoie une erreur (avec l'instruction `throw`), le contrôle passe au bloc `catch`, s'il existe, puis au bloc `finally`, s'il existe. Le bloc `finally`s'exécute toujours, qu'une erreur ait été renvoyée ou non. Si le code figurant dans le bloc `try` ne renvoie pas d'erreur (ce qui signifie que le bloc `try` se termine normalement), le code du bloc `finally` est toujours exécuté. Le bloc `finally` s'exécute même si le bloc `try` se termine par une instruction `return`.

Un bloc `try` doit être suivi par un bloc `catch`, un bloc `finally` ou les deux. Un bloc `try` peut comporter plusieurs blocs `catch` mais un seul bloc `finally`. Vous pouvez incorporer plusieurs blocs `try` et créer autant de niveaux que nécessaire.

Le paramètre *`error`* spécifié dans un gestionnaire `catch` doit être un simple identifiant tel que `e`, `theException` ou `x`. La variable d'un gestionnaire `catch` peut également être typée. Lorsqu'elles sont utilisées en conjonction avec plusieurs blocs `catch`, les erreurs typées permettent d'intercepter plusieurs types d'erreur à partir d'un bloc `try` unique.

Si l'exception renvoyée est un objet, le type correspond lorsque l'objet renvoyé constitue une sous-classe du type spécifié. Si une erreur de type spécifique est renvoyée, le bloc `catch` qui traite l'erreur correspondante s'exécute. Si l'exception renvoyée n'est pas du type spécifié, le bloc `catch` ne s'exécute pas et l'exception est renvoyée automatiquement du bloc `try`, à destination du gestionnaire `catch` correspondant.

Si une erreur est renvoyée au sein d'une fonction et si cette fonction n'inclut pas de gestionnaire `catch`, l'interpréteur ActionScript quitte alors cette fonction, ainsi que toute fonction appelante, jusqu'à ce qu'il détecte un bloc `catch`. Pendant ce processus, les gestionnaires `finally` sont appelés à tous les niveaux.

**Disponibilité :** Flash Player 7 ; ActionScript 1.0

## Paramètres

*`error`*`:Object` - Expression renvoyée par une instruction `throw`, en général une instance de la classe Error ou l'une de ses sous-classes.

## Exemple

L'exemple suivant indique comment créer une instruction `try..finally`. Étant donné que l'exécution du code dans le bloc `finally` est garantie, ce code est généralement utilisé pour effectuer le nettoyage nécessaire après l'exécution d'un bloc `try`. Dans l'exemple suivant, `setInterval()` appelle une fonction toutes les 1000 millisecondes (1 seconde). Si une erreur se produit, elle est renvoyée et interceptée par le bloc `catch`. Le bloc finally est toujours exécuté, qu'une erreur se produise ou non. Étant donné que la méthode `setInterval()` est utilisée, `clearInterval()` doit être placé dans le bloc `finally` afin de s'assurer que l'intervalle est supprimé de la mémoire.

```
myFunction = function () {
 trace("this is myFunction");
};
try {
 myInterval = setInterval(this, "myFunction", 1000);
 throw new Error("my error");
}
catch (myError:Error) {
 trace("error caught: "+myError);
}
finally {
 clearInterval(myInterval);
 trace("error is cleared");
}
```

Dans l'exemple suivant, le bloc `finally` est utilisé pour supprimer un objet ActionScript, qu'une erreur se soit produite ou non. Créez un nouveau fichier AS intitulé Account.as.

```
class Account {
 var balance:Number = 1000;
 function getAccountInfo():Number {
 return (Math.round(Math.random() * 10) % 2);
 }
}
```

Dans le répertoire du fichier Account.as, créez un nouveau document AS ou FLA et entrez le code ActionScript suivant dans l'image 1 du scénario :

```
import Account;
var account:Account = new Account();
try {
 var returnVal = account.getAccountInfo();
 if (returnVal != 0) {
 throw new Error("Error getting account information.");
 }
}
finally {
 if (account != null) {
 delete account;
 }
```

```
}
```

L'exemple ci-dessous illustre une instruction `try..catch`. Le code inclus dans le bloc `try` est exécuté. Si une exception est renvoyée par du code inclus dans le bloc `try`, le contrôle passe au bloc `catch` qui affiche le message d'erreur dans un champ de texte à l'aide de la méthode `Error.toString()`.

Dans le répertoire du fichier Account.as, créez un nouveau document FLA et entrez le code ActionScript suivant dans l'image 1 du scénario :

```
import Account;
var account:Account = new Account();
try {
 var returnVal = account.getAccountInfo();
 if (returnVal != 0) {
 throw new Error("Error getting account information.");
 }
 trace("success");
}
catch (e) {
 this.createTextField("status_txt", this.getNextHighestDepth(), 0, 0, 100,
  22);
 status_txt.autoSize = true;
 status_txt.text = e.toString();
}
```

L'exemple suivant présente un bloc de code `try` en conjonction avec plusieurs blocs de code typés `catch`. Selon le type d'erreur qui s'est produite, le bloc de code `try` renvoie un type d'objet différent. Dans ce cas, `myRecordSet` est une occurrence d'une classe (hypothétique) intitulée RecordSet dont la méthode `sortRows()` peut renvoyer deux types d'erreurs, RecordSetException et MalformedRecord.

Dans l'exemple suivant, les objets RecordSetException et MalformedRecord sont des sous-classes de la classe Error. Chacune d'entre elles est définie dans son propre fichier de classe AS.

```
// In RecordSetException.as:
class RecordSetException extends Error {
 var message = "Record set exception occurred.";
}
// In MalformedRecord.as:
class MalformedRecord extends Error {
 var message = "Malformed record exception occurred.";
}
```

Dans la méthode `sortRows()` de la classe RecordSet, l'un des objets d'erreur définis précédemment est renvoyé, en fonction du type d'exception rencontré. L'exemple suivant illustre l'aspect éventuel de ce code :

```
class RecordSet {
 function sortRows() {
 var returnVal:Number = randomNum();
```

```
 if (returnVal == 1) {
 throw new RecordSetException();
 }
 else if (returnVal == 2) {
 throw new MalformedRecord();
 }
 }
 function randomNum():Number {
 return Math.round(Math.random() * 10) % 3;
 }
}
```

Enfin, dans un autre fichier AS ou script FLA, le code suivant appelle la méthode `sortRows()` sur une occurrence de la classe RecordSet. Il définit les blocs `catch` de chaque type d'erreur renvoyé par `sortRows()`

```
import RecordSet;
var myRecordSet:RecordSet = new RecordSet();
try {
 myRecordSet.sortRows();
 trace("everything is fine");
}
catch (e:RecordSetException) {
 trace(e.toString());
}
catch (e:MalformedRecord) {
 trace(e.toString());
}
```

### Voir également

`Erreur`

## Instruction var

`var` *`variableName`* `[=` *`value1`*`][...,`*`variableNameN`*`[=`*`valueN`*`]]`

Permet de déclarer des variables locales. Si vous déclarez des variables dans une fonction, ces variables sont locales. Elles sont définies pour la fonction et expirent à la fin de l'appel de fonction. De façon plus précise, une variable définie avec `var` est une variable locale pour le bloc de code qui la contient. Les blocs de code sont signalés par des accolades ({}).

Si vous déclarez des variables en dehors d'une fonction, ces variables restent disponibles tout au long du scénario contenant l'instruction.

Vous ne pouvez pas déclarer une variable dont le domaine est limité à un autre objet en tant que variable locale.

```
my_array.length = 25; // ok
var my_array.length = 25; // syntax error
```

Lorsque vous utilisez `var`, vous pouvez typer la variable de façon stricte.

Vous pouvez déclarer plusieurs variables dans une instruction, en séparant les déclarations par des virgules (bien que cette syntaxe puisse réduire la clarté du code) :

```
var first:String = "Bart", middle:String = "J.", last:String = "Bartleby";
```

*Remarque :* Vous devez également utiliser `var` lorsque vous déclarez des propriétés au sein de définitions de classe dans les scripts externes. Les fichiers de classe prennent également en charge des domaines de variables publics, privés et statiques.

**Disponibilité :** Flash Player 5 ; ActionScript 1.0

### Paramètres

*`variableName`*`:String` - Identificateur.

### Exemple

Le script ActionScript suivant crée un nouveau tableau contenant des noms de produits. `Array.push` ajoute un élément à la fin du tableau. Si vous souhaitez utiliser le typage strict, vous devez impérativement utiliser le `var` var. Si le mot-clé `var` ne précède pas `product_array`, des erreurs se produisent lorsque vous tentez d'utiliser le typage strict.

```
var product_array:Array = new Array("MX 2004", "Studio", "Dreamweaver",
  "Flash", "ColdFusion", "Contribute", "Breeze");
product_array.push("Flex");
trace(product_array);
// output: MX
  2004,Studio,Dreamweaver,Flash,ColdFusion,Contribute,Breeze,Flex
```

## Instruction while

```
while(condition) {
statement(s);
}
```

Évalue une condition. Si cette condition renvoie `true`, exécute une instruction ou une série d'instructions avant de suivre la boucle et d'évaluer de nouveau la condition. Lorsque la condition renvoie `false`, l'instruction ou la série d'instructions est ignorée et la boucle se termine.

L'instruction `while` exécute les séries d'instructions suivantes. Toute répétition des étapes 1 à 4 constitue une *itération* de la boucle. La condition *`condition`* est testée de nouveau au début de chaque itération, comme indiqué dans les étapes suivantes :

- L'expression *`condition`* est évaluée.
- Si *`condition`* renvoie `true` ou une valeur convertie en valeur booléenne `true`, telle qu'un nombre différent de zéro, passez à l'étape 3. Sinon, l'instruction `while` se termine et l'exécution reprend au niveau de l'instruction qui suit la boucle `while`.
- Exécutez le bloc d'instructions *`statement(s)`*.
- Passez à l'étape 1.

Les boucles permettent d'exécuter une action tant que la valeur de la variable de décompte est inférieure à la valeur spécifiée. A la fin de chaque boucle, le compteur est incrémenté jusqu'à ce qu'il atteigne la valeur maximale spécifiée. A ce stade, *`condition`* n'a plus la valeur `true` et la boucle se termine.

Les accolades (`{}`), qui servent normalement à entourer le bloc d'instructions que l'instruction `while` doit exécuter, peuvent être omises si une seule instruction doit s'exécuter.

**Disponibilité :** Flash Player 4 ; ActionScript 1.0

## Paramètres

*`condition`*`:Boolean` - Expression à évaluer pour savoir si sa valeur est `true` ou `false`.

## Exemple

Dans l'exemple suivant, l'instruction `while` est utilisée pour tester une expression. Lorsque la valeur de `i` est inférieure à 20, la valeur de `i` est tracée. Lorsque la valeur de la condition n'est plus `true`, la boucle s'arrête.

```
var i:Number = 0;
while (i < 20) {
 trace(i);
 i += 3;
}
```

Le résultat suivant s'affiche dans le panneau de sortie.

```
0
3
6
9
12
15
18
```

### Voir également

`Instruction continue`

## Instruction with

```
with (object:Object) {
statement(s);
}
```

Permet de spécifier un objet (tel qu'un clip) avec le paramètre *`object`* et évalue les expressions et les actions au sein de cet objet avec le paramètre *`statement(s)`*. Ceci évite d'avoir à écrire plusieurs fois le nom de l'objet ou son chemin.

Le paramètre *`object`* forme alors le contexte de lecture des propriétés, variables et fonctions du paramètre *`statement(s)`*. Par exemple, si *`object`* est `my_array`, et que deux des propriétés spécifiées sont `length` et `concat`, ces propriétées sont automatiquement lues comme `my_array.length` et `my_array.concat`. Un autre exemple, si *`object`* est `state.california`, toutes les actions et instructions contenues dans l'instruction `with` sont appelées de l'intérieur de l'occurence `california`.

Pour déterminer la valeur d'un identificateur dans le paramètre *`statement(s)`*, ActionScript commence au début de la chaîne de domaine spécifiée par *`object`* et recherche l'identificateur à tous les niveaux de la chaîne de domaine, selon un ordre spécifique.

La chaîne de domaine utilisée par l'instruction `with` pour résoudre les identificateurs commence par le premier élément dans la liste suivante et se poursuit jusqu'au dernier :

- L'objet spécifié dans le paramètre *`object`* dans l'instruction `with` de plus bas niveau.
- L'objet spécifié dans le paramètre *`object`* de l'instruction `with` de plus haut niveau.
- Objet Activation. (Un objet temporaire qui est créé automatiquement lorsqu'une fonction est appelée et contient les variables locales appelées par la fonction.)
- Le clip qui contient le script en cours d'exécution.
- L'objet Global (objets intégrés tels que Math et String).

Pour définir une variable dans une instruction `with`, vous devez avoir déclaré cette variable en-dehors de l'instruction `with` ou vous devez entrer le chemin complet du scénario cible de la variable. Si vous définissez une variable dans une instruction `with` sans la déclarer, l'instruction `with` recherche la valeur en fonction de la chaîne de domaine. Si la variable n'existe pas, la nouvelle valeur est définie sur le scénario ayant servi à appeler l'instruction `with`.

Vous pouvez utiliser des chemins directs pour éviter `with()`. Si les chemins sont longs et difficiles à taper, créez une variable locale et enregistrez le chemin dans cette dernière. Vous pourrez alors le réutiliser dans votre code, comme indiqué dans le code ActionScript suivant.

```
var shortcut = this._parent._parent.name_txt; shortcut.text = "Hank";
  shortcut.autoSize = true;
```

**Disponibilité :** Flash Player 5 ; ActionScript 1.0

## Paramètres

*object*:Object - Occurrence de l'objet ActionScript ou du clip.

## Exemple

L'exemple suivant définit les propriétés _x et _y de l'occurrence someOther_mc, puis indique à someOther_mc de se rendre à l'image 3 et de s'arrêter.

```
with (someOther_mc) {
 _x = 50;
 _y = 100;
 gotoAndStop(3);
}
```

Le fragment de code suivant indique comment écrire le code qui précède sans instruction with.

```
someOther_mc._x = 50;
someOther_mc._y = 100;
someOther_mc.gotoAndStop(3);
```

L'instruction with est utile pour accéder à plusieurs éléments dans la liste de chaîne de domaine de manière simultanée. Dans l'exemple suivant, l'objet Math intégré est placé au début de la chaîne de domaine. Déterminer Math comme objet par défaut convertit respectivement les identificateurs cos, sin, et PI en Math.cos, Math.sin, et Math.PI. Les identificateurs a, x, y, et rne sont pas des méthodes ou des propriétés de l'objet Math mais, puisqu'ils existent dans le domaine d'activation d'objet de la fonction polar(), ils renvoient aux variables locales correspondantes.

```
function polar(r:Number):Void {
 var a:Number, x:Number, y:Number;
 with (Math) {
 a = PI * pow(r, 2);
 x = r * cos(PI);
 y = r * sin(PI / 2);
 }
 trace("area = " + a);
 trace("x = " + x);
 trace("y = " + y);
} polar(3);
```

Le résultat suivant s'affiche dans le panneau de sortie.

```
area = 28.2743338823081
x = -3
y = 3
```

CHAPITRE 2

# Classes ActionScript

2

La documentation relative aux classes ActionScript inclut des informations sur la syntaxe, l'utilisation et des exemples de code concernant les méthodes, les propriétés, les gestionnaires d'événements et les écouteurs appartenant à une classe spécifique dans ActionScript (par opposition aux fonctions ou propriétés globales). Les classes sont répertoriées par ordre alphabétique et incluent de nouvelles classes dans Flash Player 8, contenues dans les paquets flash.*. Si vous ne savez pas à quelle classe appartient une méthode ou une propriété spécifique, reportez-vous à l'index.

## Accessibility

```
Object
  |
  +-Accessibility
```

```
public class Accessibility
extends Object
```

La classe Accessibility gère la communication avec les logiciels de lecture d'écran. Les logiciels de lecture d'écran sont un type de technologie d'assistance conçu pour les utilisateurs malvoyants fournissant une version sonore du contenu de l'écran. Les méthodes de la classe Accessibility sont statiques : ainsi, vous n'avez pas besoin de créer d'occurence de la classe pour utiliser ses méthodes.

Pour obtenir et définir les propriétés d'accessibilité d'un objet spécifique, tel qu'un bouton, clip ou champ de texte, utilisez la propriété `_accProps`. Pour spécifier si le lecteur doit s'exécuter dans un environnement qui prend en charge les fonctions d'accessibilité, utilisez `System.capabilities.hasAccessibility`.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

### Voir également

`hasAccessibility (propriété capabilities.hasAccessibility)`, `_accProps`, `propriété`

### Résumé des propriétés

*Propriétés héritées de la classe Object*

| |
|---|
| constructeur (propriété Object.constructor), __proto__ (Object.__proto__ property), prototype (Object.prototype, propriété), __resolve (Object.__resolve, propriété) |

### Résumé de la méthode

| Modificateurs | Signature | Description |
|---|---|---|
| static | isActive() : Boolean | Indique si une option d'accessibilité est actuellement active et si le lecteur communique avec elle. |
| static | updateProperties() : Void | Entraîne l'entrée en vigueur de toutes les modifications apportées aux objets _accProps (propriétés d'accessibilité). |

*Méthodes héritées de la classe Object*

| |
|---|
| addProperty (méthode Object.addProperty), hasOwnProperty (méthode Object.hasOwnProperty), isPropertyEnumerable (méthode Object.isPropertyEnumerable), isPrototypeOf (méthode Object.isPrototypeOf), registerClass (méthode Object.registerClass), toString (méthode Object.toString), unwatch (méthode Object.unwatch), valueOf (méthode Object.valueOf), watch (méthode Object.watch) |

## isActive (méthode Accessibility.isActive)

```
statique publique isActive() : Boolean
```

Indique si une option d'accessibilité est actuellement active et si le lecteur communique avec elle. Utilisez cette méthode si vous souhaitez que votre application se comporte différemment en présence d'un logiciel de lecture d'écran ou de toute autre option d'accessibilité.

*Remarque :* Si vous appelez cette méthode une ou deux secondes après que la fenêtre Flash, dans laquelle vous pouvez lire votre document, se soit affichée pour la première fois, vous recevrez peut-être une valeur de retour `false` même si le client Microsoft Active Accessibility (MSAA) est actif. Ceci est dû à un mécanisme de communication asynchrone entre les clients Flash et MSAA. Vous pouvez contourner cette restriction en respectant un délai de une à deux secondes, après avoir chargé votre document, avant d'appeler cette méthode.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

### Renvoie

`Boolean` - Une valeur booléenne : `true` si Flash Player communique avec une option d'accessibilité (généralement un logiciel de lecture d'écran) ; `false` sinon.

### Exemple

L'exemple suivant vérifie si une option d'accessibilité est actuellement active :

```
if (Accessibility.isActive()) {
  trace ("An accessibility aid is currently active");
} else {
  trace ("There is currently no active accessibility aid");
}
```

### Voir également

`updateProperties (méthode Accessibility.updateProperties)`, `_accProps`, `propriété`, `hasAccessibility (propriété capabilities.hasAccessibility)`

## updateProperties (méthode Accessibility.updateProperties)

```
statique publique updateProperties() : Void
```

Entraîne l'entrée en vigueur de toutes les modifications apportées aux objets `_accProps` (propriétés d'accessibilité). Pour plus d'informations sur la configuration des propriétés d'accessibilité, consultez `_accProps`.

Si vous modifiez les propriétés d'accessibilité de plusieurs objets, un seul appel de `Accessibility.updateProperties()` est nécessaire ; plusieurs appels peuvent entraîner des performances réduites et des résultats du logiciel de lecture d'écran inintelligibles.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6,0,65,0

### Exemple

Si vous modifiez une image et souhaitez mettre à jour sa description d'accessibilité, vous pouvez utiliser le code ActionScript suivant :

```
my_mc.gotoAndStop(2);

if (my_mc._accProps == undefined ) {
  my_mc._accProps = new Object();
}
my_mc._accProps.name = "Photo of Mount Rushmore";
Accessibility.updateProperties();
```

### Voir également

isActive (méthode Accessibility.isActive), _accProps, propriété, hasAccessibility (propriété capabilities.hasAccessibility)

# arguments

```
Object
  |
  +-arguments
```

public class *arguments*
extends Object

Un objet `arguments` est utilisé pour stocker les arguments d'une fonction et y accéder. Lorsqu'il se trouve dans le corps de la fonction, vous pouvez y accéder via la variable `arguments` locale.

Les arguments sont stockés en tant qu'éléments de tableau, le premier étant accessible en tant que `arguments[0]`, le deuxième en tant que `arguments[1]`, etc. La propriété `arguments.length` indique le nombre d'arguments transmis à la fonction. Sachez que le nombre d'arguments transmis peut différer de celui ayant été déclaré par la fonction.

**Disponibilité :** ActionScript 1.0 ; Flash Player 5

### Voir également

Function

### Résumé des propriétés

| Modificateurs | Propriété | Description |
|---|---|---|
| | `callee:Object` | Référence à la fonction en cours d'exécution. |
| | `caller:Object` | Référence à la fonction ayant appelé la fonction en cours d'exécution ou `null` si elle n'a pas été appelée à partir d'une autre fonction. |
| | `length:Number` | Le nombre d'arguments transmis à la fonction. |

*Propriétés héritées de la classe Object*

constructeur (propriété Object.constructor), __proto__ (Object.__proto__ property), prototype (Object.prototype, propriété), __resolve (Object.__resolve, propriété)

### Résumé de la méthode

*Méthodes héritées de la classe Object*

| |
|---|
| addProperty (méthode Object.addProperty), hasOwnProperty (méthode Object.hasOwnProperty), isPropertyEnumerable (méthode Object.isPropertyEnumerable), isPrototypeOf (méthode Object.isPrototypeOf), registerClass (méthode Object.registerClass), toString (méthode Object.toString), unwatch (méthode Object.unwatch), valueOf (méthode Object.valueOf), watch (méthode Object.watch) |

## callee (propriété arguments.callee)

```
public callee : Object
```

Référence à la fonction en cours d'exécution.

**Disponibilité :** ActionScript 1.0 ; Flash Player 5

### Voir également

caller (propriété arguments.caller)

## caller (propriété arguments.caller)

```
public caller : Object
```

Référence à la fonction ayant appelé la fonction en cours d'exécution ou `null` si elle n'a pas été appelée à partir d'une autre fonction.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

### Voir également

callee (propriété arguments.callee)

## length (propriété arguments.length)

```
public length : Number
```

Le nombre d'arguments transmis à la fonction. Ce nombre peut être supérieur ou inférieur à celui ayant été déclaré par la fonction.

**Disponibilité :** ActionScript 1.0 ; Flash Player 5

# Array

```
Object
  |
  +-Array
```

```
public dynamic class Array
extends Object
```

La classe Array vous permet d'accéder aux tableaux indexés et de les manipuler. Un tableau indexé est un objet dont les propriétés sont identifiées par un nombre représentant leur position au sein de celui-ci. Ce nombre est appelé *index*. Tous les tableaux indexés sont basés sur zéro, ce qui signifie que le premier élément du tableau est [0], le deuxième est [1], etc. Pour créer un objet Array, utilisez le constructeur new `Array()`. Pour accéder aux éléments d'un tableau, utilisez l'opérateur d'accès au tableau (`[]`).

Vous pouvez stocker divers types de données dans un élément de tableau, y compris les nombres, les chaînes, les objets et même d'autres tableaux. Vous pouvez créer un tableau *multidimensionnel* en concevant un tableau indexé et en affectant à chacun de ses éléments un tableau indexé différent. Ce type de tableau est considéré comme étant miltidimensionnel car il peut être utilisé pour représenter des données dans un tableau.

L'affectation au tableau s'effectue par référence plutôt que par valeur : lorsque vous affectez une variable de tableau à une autre variable de tableau, elles renvoient toutes deux au même tableau :

```
var oneArray:Array = new Array("a", "b", "c");
var twoArray:Array = oneArray; // Both array variables refer to the same
  array.
twoArray[0] = "z";
trace(oneArray); // Output: z,b,c.
```

La classe Array ne doit pas être utilisée pour créer des *tableaux associatifs* car il s'agit de structures de données différentes qui contiennent des éléments nommés au lieu d'éléments numérotés. Il est recommandé d'utiliser la classe Object pour créer des tableaux associatifs (également appelés *hachages*). Bien que ActionScript vous permette de créer des tableaux associatifs à l'aide de la classe Array, vous ne pouvez pas utiliser les méthodes ou les propriétés de cette dernière. Sous sa forme de base, un tableau associatif est une occurrence de la classe Object et chaque paire clé/valeur est représentée par une propriété et sa valeur. Vous pouvez également déclarer un tableau associatif à l'aide du type de données Object pour la raison suivante : cela vous permet d'utiliser ensuite un littéral d'objet pour alimenter votre tableau associatif (uniquement au moment de la déclaration). L'exemple suivant crée un tableau associatif à l'aide d'un littéral d'objet, accède aux éléments à l'aide de l'opérateur point et d'accès au tableau, puis ajoute une nouvelle paire clé/valeur en créant une nouvelle propriété :

```
var myAssocArray:Object = {fname:"John", lname:"Public"};
```

```
trace(myAssocArray.fname); // Output: John
trace(myAssocArray["lname"]); // Output: Public
myAssocArray.initial = "Q";
trace(myAssocArray.initial); // Output: Q
```

**Disponibilité :** ActionScript 1.0 ; Flash Player 5

## Exemple

Dans l'exemple suivant, my_array contient quatre mois de l'année :

```
var my_array:Array = new Array();
my_array[0] = "January";
my_array[1] = "February";
my_array[2] = "March";
my_array[3] = "April";
```

## Résumé des propriétés

| Modificateurs | Propriété | Description |
|---|---|---|
| static | CASEINSENSITIVE:Number | Dans les méthodes de tri, cette constante spécifie le tri non sensible à la casse. |
| static | DESCENDING:Number | Dans les méthodes de tri, cette constante spécifie l'ordre de tri décroissant. |
| | length:Number | Un entier non négatif spécifiant le nombre d'éléments contenus dans le tableau. |
| static | NUMERIC:Number | Dans les méthodes de tri, cette constante spécifie le tri numérique (au lieu de la chaîne de caractères). |
| static | RETURNINDEXEDARRAY:Number | Spécifie qu'un tri renvoie un tableau indexé résultant de l'appel de la méthode sort() ou sortOn(). |
| static | UNIQUESORT:Number | Dans les méthodes de tri, cette constante spécifie l'unique exigence de tri. |

*Propriétés héritées de la classe Object*

constructeur (propriété Object.constructor), __proto__ (Object.__proto__ property), prototype (Object.prototype, propriété), __resolve (Object.__resolve, propriété)

## Résumé des constructeurs

| Signature | Description |
|---|---|
| Array([value:Object]) | Permet de créer un tableau. |

Résumé de la méthode

| Modificateurs | Signature | Description |
|---|---|---|
| | `concat([value:Object]) : Array` | Concatène les éléments spécifiés dans les paramètres avec ceux contenus dans un tableau et crée un nouveau tableau. |
| | `join([delimiter:String]) : String` | Convertit les éléments d'un tableau en chaînes, insère le séparateur spécifié entre les éléments, les concatène, puis renvoie la chaîne obtenue. |
| | `pop() : Object` | Supprime le dernier élément d'un tableau et renvoie la valeur de cet élément. |
| | `push(value:Object) : Number` | Ajoute un ou plusieurs éléments à la fin d'un tableau et renvoie la nouvelle longueur du tableau. |
| | `reverse() : Void` | Inverse le tableau. |
| | `shift() : Object` | Supprime le premier élément d'un tableau et renvoie cet élément. |
| | `slice([startIndex:Number], [endIndex:Number]) : Array` | Renvoie un nouveau tableau constitué d'un éventail d'éléments issus du tableau original, sans modifier ce dernier. |
| | `sort([compareFunction:Object], [options:Number]) : Array` | Trie les éléments d'un tableau. |
| | `sortOn(fieldName:Object, [options:Object]) : Array` | Trie les éléments d'un tableau selon un ou plusieurs champs du tableau. |
| | `splice(startIndex:Number, [deleteCount:Number], [value:Object]) : Array` | Ajoute et supprime des éléments dans un tableau. |
| | `toString() : String` | Renvoie une valeur de chaîne représentant les éléments dans l'objet Array spécifié. |
| | `unshift(value:Object) : Number` | Ajoute un ou plusieurs éléments au début d'un tableau et renvoie la nouvelle longueur du tableau. |

`addProperty (méthode Object.addProperty)`, `hasOwnProperty (méthode Object.hasOwnProperty)`, `isPropertyEnumerable (méthode Object.isPropertyEnumerable)`, `isPrototypeOf (méthode Object.isPrototypeOf)`, `registerClass (méthode Object.registerClass)`, `toString (méthode Object.toString)`, `unwatch (méthode Object.unwatch)`, `valueOf (méthode Object.valueOf)`, `watch (méthode Object.watch)`

# constructeur Array

```
public Array([value:Object])
```

Permet de créer un tableau. Vous pouvez utiliser le constructeur pour créer différents types de tableaux : un tableau vide, un tableau d'une longueur spécifique mais dont les éléments ont des valeurs non définies, ou un tableau dont les éléments ont des valeurs spécifiques.

Utilisation 1 : si vous ne spécifiez aucun paramètre, un tableau d'une longueur de 0 est créé.

Utilisation 2 : si vous spécifiez uniquement une longueur, un tableau contenant un nombre d'éléments de `length` est créé. La valeur de chaque élément est définie sur `undefined`.

Utilisation 3 : Si vous utilisez les paramètres `element` pour spécifier des valeurs, un tableau est créé avec des valeurs spécifiques.

**Disponibilité :** ActionScript 1.0 ; Flash Player 5

## Paramètres

*`value`*`:Object` [facultatif] - Soit :

- Un entier spécifiant le nombre d'éléments contenus dans le tableau.
- Une liste de deux valeurs arbitraires ou plus. Les valeurs peuvent être de type Boolean, Number, String, Object ou Array. La valeur de l'index ou de la position du premier élément d'un tableau est toujours 0.

*Remarque :* Si un seul paramètre numérique est transmis au constructeur Array, il s'agit du paramètre `length` par défaut ; celui-ci est converti en entier à l'aide de la fonction `Integer()`.

## Exemple

Utilisation 1 : L'exemple suivant crée un nouvel objet `Array` d'une longueur initiale de 0 :

```
var my_array:Array = new Array();
trace(my_array.length); // Traces 0.
```

Utilisation 2 : L'exemple suivant crée un nouvel objet Array d'une longueur initiale de 4 :

```
var my_array:Array = new Array(4);
trace(my_array.length); // Returns 4.
```

```
trace(my_array[0]); // Returns undefined.
if (my_array[0] == undefined) { // No quotation marks around undefined.
  trace("undefined is a special value, not a string");
} // Traces: undefined is a special value, not a string.
```

Utilisation 3 : L'exemple suivant crée le nouvel objet Array `go_gos_array` d'une longueur initiale de 5 :

```
var go_gos_array:Array = new Array("Belinda", "Gina", "Kathy", "Charlotte",
  "Jane");
trace(go_gos_array.length); // Returns 5.
trace(go_gos_array.join(", ")); // Displays elements.
```

Les éléments initiaux du tableau `go_gos_array` sont identifiés, comme indiqué dans l'exemple suivant :

```
go_gos_array[0] = "Belinda";
go_gos_array[1] = "Gina";
go_gos_array[2] = "Kathy";
go_gos_array[3] = "Charlotte";
go_gos_array[4] = "Jane";
```

Le code suivant ajoute un sixième élément au tableau `go_gos_array` et modifie le deuxième élément :

```
go_gos_array[5] = "Donna";
go_gos_array[1] = "Nina"
trace(go_gos_array.join(" + "));
// Returns Belinda + Nina + Kathy + Charlotte + Jane + Donna.
```

### Voir également

`Opérateur d'accès au tableau []`, `length (propriété Array.length)`

## CASEINSENSITIVE (propriété Array.CASEINSENSITIVE)

```
statique publique CASEINSENSITIVE : Number
```

Dans les méthodes de tri, cette constante spécifie le tri non sensible à la casse. Vous pouvez utiliser cette constante pour le paramètre `options` de la méthode `sort()` ou `sortOn()`.

La valeur de cette constante est 1.

**Disponibilité :** ActionScript 1.0 ; Flash Player 7

### Voir également

`sort (méthode Array.sort)`, `sortOn (méthode Array.sortOn)`

# concat (méthode Array.concat)

```
public concat([value:Object]) : Array
```

Concatène les éléments spécifiés dans les paramètres avec ceux contenus dans un tableau et crée un nouveau tableau. Si les paramètres `value` spécifient un tableau, les éléments de celui-ci sont concaténés, au lieu du tableau lui-même. Le tableau `my_array` demeure inchangé.

**Disponibilité :** ActionScript 1.0 ; Flash Player 5

## Paramètres

*`value`*`:Object` [facultatif] - Nombres, éléments ou chaînes à concaténer dans un nouveau tableau. Si vous ne transmettez aucune valeur, une duplication de `my_array` est créée.

## Renvoie

`Array` - Un tableau qui contient les éléments de ce tableau suivi des éléments des paramètres.

## Exemple

Le code suivant concatène deux tableaux :

```
var alpha_array:Array = new Array("a","b","c");
var numeric_array:Array = new Array(1,2,3);
var alphaNumeric_array:Array =alpha_array.concat(numeric_array);
trace(alphaNumeric_array);
// Creates array [a,b,c,1,2,3].
```

Le code suivant concatène trois tableaux :

```
var num1_array:Array = [1,3,5];
var num2_array:Array = [2,4,6];
var num3_array:Array = [7,8,9];
var nums_array:Array=num1_array.concat(num2_array,num3_array)
trace(nums_array);
// Creates array [1,3,5,2,4,6,7,8,9].
```

Les tableaux incorporés ne sont pas aplatis de la même manière que les tableaux normaux. Les éléments d'un tableau incorporé ne sont pas séparés en éléments distincts dans le tableau `x_array`, comme indiqué dans l'exemple suivant :

```
var a_array:Array = new Array ("a","b","c");

// 2 and 3 are elements in a nested array.
var n_array:Array = new Array(1, [2, 3], 4);

var x_array:Array = a_array.concat(n_array);
trace(x_array[0]); // a
trace(x_array[1]); // b
trace(x_array[2]); // c
trace(x_array[3]); // 1
```

```
trace(x_array[4]); // 2, 3
trace(x_array[5]); // 4
```

# DESCENDING (propriété Array.DESCENDING)

```
statique publique DESCENDING : Number
```

Dans les méthodes de tri, cette constante spécifie l'ordre de tri décroissant. Vous pouvez utiliser cette constante pour le paramètre `options` de la méthode `sort()` ou `sortOn()`.

La valeur de cette constante est 2.

**Disponibilité :** ActionScript 1.0 ; Flash Player 7

## Voir également

`sort (méthode Array.sort)`, `sortOn (méthode Array.sortOn)`

# join (méthode Array.join)

```
public join([delimiter:String]) : String
```

Convertit les éléments d'un tableau en chaînes, insère le séparateur spécifié entre les éléments, les concatène, puis renvoie la chaîne obtenue. Un tableau imbriqué est toujours séparé par une virgule (,), et non pas par le séparateur transmis à la méthode `join()`.

**Disponibilité :** ActionScript 1.0 ; Flash Player 5

## Paramètres

*`delimiter`*`:String` [facultatif] - Un caractère ou une chaîne séparant les éléments du tableau dans la chaîne renvoyée. Si vous omettez ce paramètre, une virgule (,) est utilisée en tant que séparateur par défaut.

## Renvoie

`String` - Une chaîne.

## Exemple

L'exemple suivant crée un tableau incluant trois éléments : Earth, Moon et Sun. Il relie ensuite le tableau trois fois, d'abord à l'aide du séparateur par défaut (une virgule [,] et un espace), puis à l'aide d'un tiret (-) et enfin d'un signe plus (+).

```
var a_array:Array = new Array("Earth","Moon","Sun")
trace(a_array.join());
// Displays Earth,Moon,Sun.
trace(a_array.join(" - "));
// Displays Earth - Moon - Sun.
```

```
trace(a_array.join(" + "));
// Displays Earth + Moon + Sun.
```

L'exemple suivant crée un tableau incorporé qui contient deux tableaux. Le premier tableau inclut trois éléments : Europa, Io et Callisto. Le deuxième tableau inclut deux éléments : Titan et Rhea. Il relie le tableau à l'aide d'un signe plus (+) mais les éléments de chaque tableau incorporé restent séparés par des virgules (,).

```
var a_nested_array:Array = new Array(["Europa", "Io", "Callisto"],
  ["Titan", "Rhea"]);
trace(a_nested_array.join(" + "));
// Returns Europa,Io,Callisto + Titan,Rhea.
```

### Voir également

`split (méthode String.split)`

## length (propriété Array.length)

```
public length : Number
```

Un entier non négatif spécifiant le nombre d'éléments contenus dans le tableau. Cette propriété est automatiquement mise à jour lorsque vous ajoutez de nouveaux éléments dans le tableau. Lorsque vous affectez une valeur à un élément de tableau (par exemple, `my_array[index] = value`), si `index` est un nombre et si `index+1` est supérieur à la propriété `length`, la propriété `length` est mise à jour et définie sur la valeur `index+1`.

*Remarque :* Si vous affectez une valeur plus courte que la valeur existante à la propriété `length`, le tableau sera tronqué.

**Disponibilité :** ActionScript 1.0 ; Flash Player 5

### Exemple

Le code suivant explique la façon dont la propriété `length` est mise à jour. La valeur de la longueur initiale est 0, puis 1, 2 et 10. Si vous affectez une valeur plus courte que la valeur existante à la propriété `length`, le tableau sera tronqué :

```
var my_array:Array = new Array();
trace(my_array.length); // initial length is 0
my_array[0] = "a";
trace(my_array.length); // my_array.length is updated to 1
my_array[1] = "b";
trace(my_array.length); // my_array.length is updated to 2
my_array[9] = "c";
trace(my_array.length); // my_array.length is updated to 10
trace(my_array);
// displays:
```

```
//
  a,b,undefined,undefined,undefined,undefined,undefined,undefined,undefine
  d,c

// if the length property is now set to 5, the array will be truncated
my_array.length = 5;
trace(my_array.length); // my_array.length is updated to 5
trace(my_array); // outputs: a,b,undefined,undefined,undefined
```

# NUMERIC (propriété Array.NUMERIC)

```
statique publique NUMERIC : Number
```

Dans les méthodes de tri, cette constante spécifie le tri numérique (au lieu de la chaîne de caractères). Si vous l'incluez au paramètre `options`, cela oblige les méthodes `sort()` et `sortOn()` à trier les nombres en tant que valeurs numériques, et non en tant que chaînes de caractères numériques. En l'absence de la constante NUMERIC, le tri traite chaque élément de tableau en tant que chaîne de caractère et donne les résultats selon l'ordre Unicode.

Prenons l'exemple suivant pour le tableau de valeurs [2005, 7, 35] : si la constante NUMERIC n'est *pas* incluse dans le paramètre `options`, le tableau trié est [2005, 35, 7] ; en revanche, si la constante NUMERIC *est* incluse, le tableau trié est [7, 35, 2005].

Notez que cette constante s'applique uniquement aux nombres contenus dans le tableau ; elle ne s'applique pas aux chaînes qui contiennent des données numériques (telles que [23, 5]).

La valeur de cette constante est 16.

**Disponibilité :** ActionScript 1.0 ; Flash Player 7

## Voir également

`sort (méthode Array.sort)`, `sortOn (méthode Array.sortOn)`

# pop (méthode Array.pop)

```
public pop() : Object
```

Supprime le dernier élément d'un tableau et renvoie la valeur de cet élément.

**Disponibilité :** ActionScript 1.0 ; Flash Player 5

## Renvoie

`Object` - La valeur du dernier élément dans le tableau spécifié.

### Exemple

Le code suivant crée le tableau `myPets_array` contenant quatre éléments, puis supprime son dernier élément :

```
var myPets_array:Array = new Array("cat", "dog", "bird", "fish");
var popped:Object = myPets_array.pop();
trace(popped); // Displays fish.
trace(myPets_array); // Displays cat,dog,bird.
```

### Voir également

`push (méthode Array.push)`, `shift (méthode Array.shift)`, `unshift (méthode Array.unshift)`

## push (méthode Array.push)

```
public push(value:Object) : Number
```

Ajoute un ou plusieurs éléments à la fin d'un tableau et renvoie la nouvelle longueur du tableau.

**Disponibilité :** ActionScript 1.0 ; Flash Player 5

### Paramètres

*`value`*`:Object` - Une ou plusieurs valeurs à ajouter au tableau.

### Renvoie

`Number` - Un entier représentant la longueur du nouveau tableau.

### Exemple

L'exemple suivant crée le tableau `myPets_array` incluant deux éléments, `cat` et `dog`. La deuxième ligne ajoute deux éléments au tableau.

Etant donné que la méthode `push()` renvoie la nouvelle longueur du tableau, l'instruction `trace()` de la dernière ligne envoie la nouvelle longueur du tableau `myPets_array` (`4`) vers le panneau de sortie.

```
var myPets_array:Array = new Array("cat", "dog");
var pushed:Number = myPets_array.push("bird", "fish");
trace(pushed); // Displays 4.
```

### Voir également

`pop (méthode Array.pop)`, `shift (méthode Array.shift)`, `unshift (méthode Array.unshift)`

# RETURNINDEXEDARRAY (propriété Array.RETURNINDEXEDARRAY)

```
statique publique RETURNINDEXEDARRAY : Number
```

Spécifie qu'un tri renvoie un tableau indexé résultant de l'appel de la méthode `sort()` ou `sortOn()`. Vous pouvez utiliser cette constante pour le paramètre `options` de la méthode `sort()` ou `sortOn()`. Cette méthode fournit les fonctions d'aperçu et de copie en renvoyant un tableau qui représente les résultats du tri et ne modifie pas le tableau d'origine.

La valeur de cette constante est 8.

**Disponibilité :** ActionScript 1.0 ; Flash Player 7

## Voir également

`sort (méthode Array.sort)`, `sortOn (méthode Array.sortOn)`

# reverse (méthode Array.reverse)

```
public reverse() : Void
```

Inverse le tableau.

**Disponibilité :** ActionScript 1.0 ; Flash Player 5

## Exemple

L'exemple suivant utilise cette méthode pour inverser le tableau `numbers_array` :

```
var numbers_array:Array = new Array(1, 2, 3, 4, 5, 6);
trace(numbers_array); // Displays 1,2,3,4,5,6.
numbers_array.reverse();
trace(numbers_array); // Displays 6,5,4,3,2,1.
```

# shift (méthode Array.shift)

```
public shift() : Object
```

Supprime le premier élément d'un tableau et renvoie cet élément.

**Disponibilité :** ActionScript 1.0 ; Flash Player 5

## Renvoie

`Object` - Le premier élément d'un tableau.

### Exemple

Le code suivant crée le tableau `myPets_array`, supprime le premier élément du tableau, puis l'affecte à la variable `shifted` :

```
var myPets_array:Array = new Array("cat", "dog", "bird", "fish");
var shifted:Object = myPets_array.shift();
trace(shifted); // Displays "cat".
trace(myPets_array); // Displays dog,bird,fish.
```

### Voir également

`pop (méthode Array.pop)`, `push (méthode Array.push)`, `unshift (méthode Array.unshift)`

## slice (méthode Array.slice)

```
public slice([startIndex:Number], [endIndex:Number]) : Array
```

Renvoie un nouveau tableau constitué d'un éventail d'éléments issus du tableau original, sans modifier ce dernier. Le tableau renvoyé inclut l'élément `startIndex` et tous les éléments, excepté l'élément `endIndex`.

Si vous ne transmettez aucun paramètre, une duplication du tableau d'origine est créée.

**Disponibilité :** ActionScript 1.0 ; Flash Player 5

### Paramètres

*`startIndex`*`:Number` [facultatif] - Un nombre spécifiant l'index du point de départ pour la découpe. Si *`start`* est un nombre négatif, le point de départ se trouve à la fin du tableau, où la valeur -1 est le dernier élément.

*`endIndex`*`:Number` [facultatif] - Un nombre spécifiant l'index du point d'arrivée pour la découpe. Si vous omettez ce paramètre, la découpe inclut tous les éléments du point de départ à la fin du tableau. Si *`end`* est un nombre négatif, le point d'arrivée spécifié se trouve à la fin du tableau, où la valeur -1 est le dernier élément.

### Renvoie

`Array` - Un tableau constitué d'un éventail d'éléments issus du tableau original.

### Exemple

L'exemple suivant crée un tableau incluant cinq animaux domestiques et utilise la méthode `slice()` pour alimenter un nouveau tableau contenant uniquement les animaux à quatre pattes :

```
var myPets_array:Array = new Array("cat", "dog", "fish", "canary",
  "parrot");
var myFourLeggedPets_array:Array = new Array();
var myFourLeggedPets_array = myPets_array.slice(0, 2);
trace(myFourLeggedPets_array); // Returns cat,dog.
trace(myPets_array); // Returns cat,dog,fish,canary,parrot.
```

L'exemple suivant crée un tableau incluant cinq animaux domestiques, puis utilise la méthode `slice()` avec un paramètre `start` négatif pour copier les deux derniers éléments du tableau :

```
var myPets_array:Array = new Array("cat", "dog", "fish", "canary",
  "parrot");
var myFlyingPets_array:Array = myPets_array.slice(-2);
trace(myFlyingPets_array); // Traces canary,parrot.
```

L'exemple suivant crée un tableau incluant cinq animaux domestiques et utilise la méthode `slice()` avec un paramètre `end` négatif pour copier l'élément central du tableau :

```
var myPets_array:Array = new Array("cat", "dog", "fish", "canary",
  "parrot");
var myAquaticPets_array:Array = myPets_array.slice(2,-2);
trace(myAquaticPets_array); // Returns fish.
```

## sort (méthode Array.sort)

```
public sort([compareFunction:Object], [options:Number]) : Array
```

Trie les éléments d'un tableau. Flash trie selon les valeurs Unicode. (ASCII est un sous-ensemble de Unicode.)

Par défaut, `Array.sort()` fonctionne comme décrit dans la liste suivante :

- Le tri tient compte de la casse (*Z* précède *a*).
- Le tri est ascendant (*a* précède *b*).
- Le tableau est modifié afin de refléter l'ordre de tri ; plusieurs éléments, dont les champs de tri sont identiques, sont placés de manière consécutive dans le tableau trié dans un ordre quelconque.
- Les champs numériques sont triés comme s'il s'agissait de chaînes : ainsi, 100 précède 99 car « 1 » est une valeur de chaîne inférieure à « 9 ».

Si vous voulez trier un tableau à l'aide de paramètres qui ne correspondent pas aux paramètres par défaut, vous pouvez utiliser l'une des options de tri décrites dans l'entrée du paramètre `options` ou vous pouvez créer votre propre fonction personnalisée pour effectuer le tri. Si vous créez une fonction personnalisée, vous pouvez l'utiliser en appelant la méthode `sort()` et en utilisant le nom de votre fonction personnalisée en tant que premier paramètre (`compareFunction`).

**Disponibilité :** ActionScript 1.0 ; Flash Player 5

## Paramètres

*`compareFunction`*`:Object` [facultatif] - Une fonction de comparaison utilisée pour déterminer l'ordre de tri des éléments dans un tableau. Etant donné les éléments A et B, le résultat de `compareFunction` peut être l'une des trois valeurs suivantes :

- -1, si A apparaît avant B dans la séquence triée
- 0, si A = B
- 1, si A apparaît après B dans la séquence triée

*`options`*`:Number` [facultatif] - Un ou plusieurs nombres ou noms de constantes définies, séparés par l'opérateur OR `|` au niveau du bit, ce qui remplace le comportement de tri par défaut. Les valeurs suivantes sont valides pour le paramètre `options` :

- `Array.CASEINSENSITIVE` ou 1
- `Array.DESCENDING` ou 2
- `Array.UNIQUESORT` ou 4
- `Array.RETURNINDEXEDARRAY` ou 8
- `Array.NUMERIC` ou 16

Pour plus d'informations sur ce paramètre, consultez la méthode `Array.sortOn()`.

*Remarque :* La méthode `Array.sort()` est définie dans la norme ECMA-262 mais les options de tri de tableau introduites dans Flash Player 7 sont des extensions spécifiques à Flash de la spécification ECMA-262.

## Renvoie

`Array` - La valeur de renvoi dépend du fait que vous transmettiez ou non des paramètres, comme décrit dans la liste suivante :

- Si vous spécifiez une valeur de 4 ou `Array.UNIQUESORT` pour le paramètre `options` et si au moins deux éléments triés ont des champs de tri identiques, Flash renvoie une valeur de 0 et ne modifie pas le tableau.

- Si vous spécifiez une valeur de 8 ou `Array.RETURNINDEXEDARRAY` pour le paramètre `options`, Flash renvoie un tableau qui reflète les résultats du tri et ne modifie pas le tableau.
- Dans le cas contraire, Flash ne renvoie rien et modifie le tableau pour refléter l'ordre de tri.

## Exemple

Utilisation 1 : L'exemple suivant illustre l'utilisation de `Array.sort()`, avec et sans valeur transmise à `options` :

```
var fruits_array:Array = new Array("oranges", "apples", "strawberries",
  "pineapples", "cherries");
trace(fruits_array); // Displays
  oranges,apples,strawberries,pineapples,cherries.
fruits_array.sort();
trace(fruits_array); // Displays
  apples,cherries,oranges,pineapples,strawberries.
trace(fruits_array); // Writes
  apples,cherries,oranges,pineapples,strawberries.
fruits_array.sort(Array.DESCENDING);
trace(fruits_array); // Displays
  strawberries,pineapples,oranges,cherries,apples.
trace(fruits_array); // Writes
  strawberries,pineapples,oranges,cherries,apples.
```

Utilisation 2 : L'exemple suivant utilise `Array.sort()` avec une fonction de comparaison. Les entrées sont triées sous la forme nom:mot de passe. Triez en utilisant uniquement la partie nom de l'entrée en tant que clé :

```
var passwords_array:Array = new Array("mom:glam", "ana:ring", "jay:mag",
  "anne:home", "regina:silly");
function order(a, b):Number {
  var name1:String = a.split(":")[0];
  var name2:String = b.split(":")[0];
  if (name1<name2) {
  return -1;
  } else if (name1>name2) {
  return 1;
  } else {
  return 0;
  }
}
trace("Unsorted:");
//Displays Unsorted:
trace(passwords_array);
//Displays mom:glam,ana:ring,jay:mag,anne:home,regina:silly.
//Writes mom:glam,ana:ring,jay:mag,anne:home,regina:silly
passwords_array.sort(order);
trace("Sorted:");
//Displays Sorted:
```

```
trace(passwords_array);
//Displays ana:ring,anne:home,jay:mag,mom:glam,regina:silly.
//Writes ana:ring,anne:home,jay:mag,mom:glam,regina:silly.
```

### Voir également

`| Opérateur OR au niveau du bit`, `sortOn (méthode Array.sortOn)`

## sortOn (méthode Array.sortOn)

```
public sortOn(fieldName:Object, [options:Object]) : Array
```

Trie les éléments d'un tableau selon un ou plusieurs champs du tableau. Le tableau doit être doté des caractéristiques suivantes :

- Le tableau est indexé et non associatif.
- Chaque élément du tableau contient un objet doté d'une ou de plusieurs propriétés.
- Tous les objets ont au moins une propriété en commun dont les valeurs peuvent être utilisées pour trier le tableau. Ce type de propriété est connu sous le nom de *champ*.

Si vous transmettez plusieurs paramètres `fieldName`, le premier champ représente le champ de tri principal, le deuxième représente le champ de tri suivant, etc. Flash trie selon les valeurs Unicode. (ASCII est un sous-ensemble de Unicode.) Si l'un des éléments comparés ne contient pas le champ spécifié dans le paramètre `fieldName`, le champ est considéré comme étant `undefined` et les éléments sont placés de manière consécutive dans le tableau trié dans un ordre quelconque.

Par défaut, `Array.sortOn()` fonctionne de la façon suivante :

- Le tri tient compte de la casse (*Z* précède *a*).
- Le tri est ascendant (*a* précède *b*).
- Le tableau est modifié afin de refléter l'ordre de tri ; plusieurs éléments, dont les champs de tri sont identiques, sont placés de manière consécutive dans le tableau trié dans un ordre quelconque.
- Les champs numériques sont triés comme s'il s'agissait de chaînes : ainsi, 100 précède 99 car « 1 » est une valeur de chaîne inférieure à « 9 ».

Flash Player 7 a jouté le paramètre `options` que vous pouvez utiliser pour annuler le comportement de tri par défaut. Pour trier un tableau simple (par exemple, un tableau contenant un seul champ) ou pour spécifier un ordre de tri non pris en charge par le paramètre `options`, utilisez `Array.sort()`.

Pour définir plusieurs indicateurs, séparez-les à l'aide de l'opérateur OR (`|`) au niveau du bit :

```
my_array.sortOn(someFieldName, Array.DESCENDING | Array.NUMERIC);
```

La fonctionnalité ajoutée à Flash Player 8 permet de spécifier une option de tri différente pour chaque champ lors d'un tri selon plusieurs champs. Dans Flash Player 8, le paramètre `options` accepte un tableau d'options de tri de telle sorte que chaque option de tri correspond à un champ de tri dans le paramètre `fieldName`. L'exemple suivant trie le champ de tri principal, `a`, selon un tri décroissant, le deuxième champ de tri, `b`, selon un tri numérique et le troisième champ de tri, `c`, selon un tri non sensible à la casse :

```
Array.sortOn (["a", "b", "c"], [Array.DESCENDING, Array.NUMERIC,
  Array.CASEINSENSITIVE]);
```

*Remarque :* Les tableaux `fieldName` et `options` doivent contenir le même nombre d'éléments ; sinon, le tableau `options` est ignoré. En outre, les options `Array.UNIQUESORT` et `Array.RETURNINDEXEDARRAY` ne peuvent être utilisées qu'en tant que premier élément du tableau, sinon elles sont ignorées.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

## Paramètres

*`fieldName`*`:Object` - Une chaîne identifiant un champ à utiliser en tant que valeur de tri ou un tableau dans lequel le premier élément représente le champ de tri principal, le deuxième le champ de tri secondaire, etc.

*`options`*`:Object` [facultatif] - Un ou plusieurs nombres ou noms de constantes définies, séparés par l'opérateur `bitwise OR (|)`, ce qui remplace le comportement de tri. Les valeurs suivantes sont valides pour le paramètre `options` :

- `Array.CASEINSENSITIVE` ou 1
- `Array.DESCENDING` ou 2
- `Array.UNIQUESORT` ou 4
- `Array.RETURNINDEXEDARRAY` ou 8
- `Array.NUMERIC` ou 16

Les conseils de code sont activés si vous utilisez le format chaîne de l'indicateur (par exemple, `DESCENDING`) au lieu du format numérique (2).

## Renvoie

`Array` - La valeur de renvoi dépend du fait que vous transmettiez ou non des paramètres :

- Si vous spécifiez une valeur de 4 ou `Array.UNIQUESORT` pour le paramètre `options` et si au moins deux éléments triés ont des champs de tri identiques, la valeur 0 est renvoyée et le tableau n'est pas modifié.

- Si vous spécifiez une valeur de 8 ou `Array.RETURNINDEXEDARRAY` pour le paramètre `options`, un tableau qui reflète les résultats du tri est renvoyé et le tableau n'est pas modifié.
- Dans le cas contraire, Flash ne renvoie rien et modifie le tableau pour refléter l'ordre de tri.

## Exemple

L'exemple suivant crée un nouveau tableau et le trie selon les champs `name` et `city`. Le premier tri utilise `name` en tant que première valeur de tri et `city` en tant que deuxième valeur de tri. Le deuxième tri utilise `city` en tant que première valeur de tri et `name` en tant que deuxième valeur de tri.

```
var rec_array:Array = new Array();
rec_array.push({name: "john", city: "omaha", zip: 68144});
rec_array.push({name: "john", city: "kansas city", zip: 72345});
rec_array.push({name: "bob", city: "omaha", zip: 94010});
for(i=0; i<rec_array.length; i++){
  trace(rec_array[i].name + ", " + rec_array[i].city);
}
// Results:
// john, omaha
// john, kansas city
// bob, omaha

rec_array.sortOn(["name", "city"]);
for(i=0; i<rec_array.length; i++){
  trace(rec_array[i].name + ", " + rec_array[i].city);
}
// Results:
// bob, omaha
// john, kansas city
// john, omaha

rec_array.sortOn(["city", "name" ]);
for(i=0; i<rec_array.length; i++){
  trace(rec_array[i].name + ", " + rec_array[i].city);
}
// Results:
// john, kansas city
// bob, omaha
// john, omaha
```

Le tableau d'objets suivant est utilisé par les exemples suivants, qui montrent comment utiliser le paramètre `options` :

```
var my_array:Array = new Array();
my_array.push({password: "Bob", age:29});
my_array.push({password: "abcd", age:3});
my_array.push({password: "barb", age:35});
```

```
my_array.push({password: "catchy", age:4});
```

La réalisation d'un tri par défaut à partir du champ du mot de passe donne les résultats suivants :

```
my_array.sortOn("password");
// Bob
// abcd
// barb
// catchy
```

La réalisation d'un tri non sensible à la casse à partir du champ du mot de passe donne les résultats suivants :

```
my_array.sortOn("password", Array.CASEINSENSITIVE);
// abcd
// barb
// Bob
// catchy
```

La réalisation d'un tri décroissant non sensible à la casse à partir du champ du mot de passe donne les résultats suivants :

```
my_array.sortOn("password", Array.CASEINSENSITIVE | Array.DESCENDING);
// catchy
// Bob
// barb
// abcd
```

La réalisation d'un tri par défaut à partir du champ âge donne les résultats suivants :

```
my_array.sortOn("age");
// 29
// 3
// 35
// 4
```

La réalisation d'un tri numérique à partir du champ âge donne les résultats suivants :

```
my_array.sortOn("age", Array.NUMERIC);
// my_array[0].age = 3
// my_array[1].age = 4
// my_array[2].age = 29
// my_array[3].age = 35
```

La réalisation d'un tri numérique décroissant à partir du champ âge donne les résultats suivants :

```
my_array.sortOn("age", Array.DESCENDING | Array.NUMERIC);
// my_array[0].age = 35
// my_array[1].age = 29
// my_array[2].age = 4
// my_array[3].age = 3
```

Lorsque vous utilisez l'option de tri `Array.RETURNEDINDEXARRAY`, vous devez affecter la valeur renvoyée à un tableau différent. Le tableau d'origine n'est pas modifié.

```
var indexArray:Array = my_array.sortOn("age", Array.RETURNINDEXEDARRAY);
```

### Voir également

| Opérateur OR au niveau du bit, sort (méthode Array.sort)

## splice (méthode Array.splice)

```
public splice(startIndex:Number, [deleteCount:Number], [value:Object]) :
  Array
```

Ajoute et supprime des éléments dans un tableau. Cette méthode modifie le tableau sans faire de copie.

**Disponibilité :** ActionScript 1.0 ; Flash Player 5

### Paramètres

*startIndex*:Number - Un entier spécifiant l'index de la position d'insertion ou de suppression de l'élément dans le tableau. Vous pouvez spécifier un entier négatif pour définir une position par rapport à la fin du tableau (par exemple, la valeur -1 représente le dernier élément du tableau).

*deleteCount*:Number [facultatif] - Un entier spécifiant le nombre d'éléments à supprimer. Ce nombre inclut l'élément spécifié dans le paramètre `startIndex`. Si aucune valeur n'est spécifiée pour le paramètre `deleteCount`, la méthode supprime toutes les valeurs comprises entre l'élément `startIndex` et le dernier élément du tableau. Si la valeur est 0, aucun élément n'est supprimé.

*value*:Object [facultatif] - Spécifie les valeurs à insérer dans le tableau au point d'insertion défini dans le paramètre `startIndex`.

### Renvoie

`Array` - Un tableau contenant les éléments supprimés du tableau original.

### Exemple

L'exemple suivant crée un tableau et le relie à l'aide de l'élément index 1 pour le paramètre `startIndex`. Tous les éléments du tableau à partir du deuxième élément sont ainsi supprimés : seul l'élément à l'index 0 est conservé dans le tableau d'origine :

```
var myPets_array:Array = new Array("cat", "dog", "bird", "fish");
trace( myPets_array.splice(1) ); // Displays dog,bird,fish.
trace( myPets_array ); // cat
```

L'exemple suivant crée un tableau et le relie à l'aide de l'élément index 1 pour le paramètre `startIndex` et du nombre 2 pour le paramètre `deleteCount`. Deux éléments du tableau à partir du deuxième élément sont ainsi supprimés : seuls les premier et dernier éléments sont conservés dans le tableau d'origine :

```
var myFlowers_array:Array = new Array("roses", "tulips", "lilies",
  "orchids");
trace( myFlowers_array.splice(1,2 ) ); // Displays tulips,lilies.
trace( myFlowers_array ); // roses,orchids
```

L'exemple suivant crée un tableau et le relie à l'aide de l'élément index 1 pour le paramètre `startIndex`, du nombre 0 pour le paramètre `deleteCount` et de la chaîne `chair` pour le paramètre `value`. Aucun élément n'est supprimé du tableau d'origine et la chaîne `chair` est ajoutée à l'index 1 :

```
var myFurniture_array:Array = new Array("couch", "bed", "desk", "lamp");
trace( myFurniture_array.splice(1,0, "chair" ) ); // Displays empty array.
trace( myFurniture_array ); // displays couch,chair,bed,desk,lamp
```

# toString (méthode Array.toString)

```
public toString() : String
```

Renvoie une valeur de chaîne représentant les éléments dans l'objet Array spécifié. Chaque élément du tableau, commençant par l'index 0 et se terminant par l'index le plus élevé, est converti en chaîne concaténée et séparé par des virgules. Pour spécifier un séparateur personnalisé, utilisez la méthode `Array.join()`.

**Disponibilité :** ActionScript 1.0 ; Flash Player 5

## Renvoie

`String` - Une chaîne.

## Exemple

L'exemple suivant crée le tableau `my_array` et le convertit en chaîne.

```
var my_array:Array = new Array();
my_array[0] = 1;
my_array[1] = 2;
my_array[2] = 3;
my_array[3] = 4;
my_array[4] = 5;
trace(my_array.toString()); // Displays 1,2,3,4,5.
```

Cet exemple renvoie le résultat 1,2,3,4,5 de l'instruction `trace`.

### Voir également

split (méthode String.split), join (méthode Array.join)

## UNIQUESORT (propriété Array.UNIQUESORT)

```
statique publique UNIQUESORT : Number
```

Dans les méthodes de tri, cette constante spécifie l'unique exigence de tri. Vous pouvez utiliser cette constante pour le paramètre options de la méthode `sort()` ou `sortOn()`. L'option de tri unique abandonne le tri si deux éléments ou champs triés ont des valeurs identiques.

La valeur de cette constante est 4.

**Disponibilité :** ActionScript 1.0 ; Flash Player 7

### Voir également

sort (méthode Array.sort), sortOn (méthode Array.sortOn)

## unshift (méthode Array.unshift)

```
public unshift(value:Object) : Number
```

Ajoute un ou plusieurs éléments au début d'un tableau et renvoie la nouvelle longueur du tableau.

**Disponibilité :** ActionScript 1.0 ; Flash Player 5

### Paramètres

*value*:Object - Un ou plusieurs nombres, éléments ou variables à insérer au début du tableau.

### Renvoie

Number - Un entier représentant la nouvelle longueur du tableau.

### Exemple

L'exemple suivant illustre l'utilisation de la méthode `Array.unshift()` :

```
var pets_array:Array = new Array("dog", "cat", "fish");
trace( pets_array ); // Displays dog,cat,fish.
pets_array.unshift("ferrets", "gophers", "engineers");
trace( pets_array ); // Displays ferrets,gophers,engineers,dog,cat,fish.
```

### Voir également

pop (méthode Array.pop), push (méthode Array.push), shift (méthode Array.shift)

# AsBroadcaster

```
Object
  |
  +-AsBroadcaster
```

public class *AsBroadcaster*
extends Object

Offre des fonctionnalités de notification d'événements et de gestion des écouteurs pouvant être ajoutées à des objets définis par l'utilisateur. Cette classe est destinée aux utilisateurs expérimentés qui souhaitent créer des mécanismes de gestion d'événements personnalisés. Vous pouvez utiliser cette classe pour définir un objet en tant que diffuseur d'événements et pour créer un ou plusieurs objets écouteurs qui reçoivent une notification à chaque fois que l'objet de diffusion appelle la méthode `broadcastMessage()`.

Aucune fonction constructeur n'existe pour la classe AsBroadcaster. Pour utiliser cette classe, procédez comme suit :

- Sélectionnez ou créez un objet faisant office de diffuseur d'événements.
- Définissez l'objet en tant que diffuseur d'événements en appelant la méthode `AsBroadcaster.initialize(obj:Object)` statique, où le paramètre `obj` est le nom de l'objet que vous avez sélectionné en tant que diffuseur.
- Sélectionnez ou créez un ou plusieurs objets écouteurs. Les objets écouteurs reçoivent une notification à chaque fois que l'objet de diffusion envoie un message
- Définissez une méthode d'écouteur pour chaque objet écouteur. La méthode d'écouteur exécute le code ActionScript en réponse à la notification d'événement. Le nom de la méthode doit correspondre à celui de l'événement diffusé par l'objet de diffusion.
- Enregistrez chaque objet écouteur avec le diffuseur d'événements en appelant `myBroadcaster.addListener(myListener)`, où `myBroadcaster` est le nom de l'objet diffuseur d'événements et `myListener` le nom de l'objet écouteur. Chaque diffuseur d'événements stocke une liste d'objets écouteurs devant être informés lorsqu'un message est diffusé. Utilisez la méthode `addListener()` pour ajouter des écouteurs dans la liste et `removeListener()` pour les supprimer de la liste.

- Enfin, pour envoyer un message, appelez la méthode `myBroadcaster.broadcastMessage(eventName:String)`, où `myBroadcaster` est le nom du diffuseur d'événements et `eventName` le nom de l'événement correspondant à celui de la méthode d'écouteur.

*Remarque :* Une erreur courante consiste à mettre en majuscule la deuxième lettre de *AsBroadcaster*. Lorsque vous appelez la méthode `AsBroadcaster.initialize()`, assurez-vous que la deuxième lettre est en minuscule. Toute faute d'orthographe apparaissant dans *AsBroadcaster* échoue sans indication.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

## Résumé des propriétés

| Modificateurs | Propriété | Description |
|---|---|---|
| | `_listeners:Array` [lecture seule] | Une liste de références à tous les objets écouteurs enregistrés. |

*Propriétés héritées de la classe Object*

`constructeur (propriété Object.constructor), __proto__ (Object.__proto__ property), prototype (Object.prototype, propriété), __resolve (Object.__resolve, propriété)`

### Résumé de la méthode

| Modificateurs | Signature | Description |
|---|---|---|
| | `addListener(listenerObj:Object) : Boolean` | Enregistre un objet de façon à ce que ce dernier reçoive les messages de notification. |
| | `broadcastMessage(eventName:String) : Void` | Envoie un message d'événement à chaque objet de la liste d'écouteurs. |
| `static` | `initialize(obj:Object) : Void` | Ajoute une fonctionnalité de notification d'événements et de gestion des écouteurs à un objet donné. |
| | `removeListener(listenerObj:Object) : Boolean` | Supprime un objet de la liste d'objets recevant des messages de notification d'événement. |

*Méthodes héritées de la classe Object*

`addProperty (méthode Object.addProperty)`, `hasOwnProperty (méthode Object.hasOwnProperty)`, `isPropertyEnumerable (méthode Object.isPropertyEnumerable)`, `isPrototypeOf (méthode Object.isPrototypeOf)`, `registerClass (méthode Object.registerClass)`, `toString (méthode Object.toString)`, `unwatch (méthode Object.unwatch)`, `valueOf (méthode Object.valueOf)`, `watch (méthode Object.watch)`

# addListener (méthode AsBroadcaster.addListener)

```
public addListener(listenerObj:Object) : Boolean
```

Enregistre un objet de façon à ce que ce dernier reçoive les messages de notification. Cette méthode est appelée sur l'objet de diffusion et l'objet écouteur est envoyé en tant qu'argument.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

### Paramètres

*`listenerObj`*`:Object` - Le nom de l'objet écouteur recevant la notification d'événement.

### Renvoie

`Boolean` - Bien que, d'un point de vue technique, cette méthode renvoie une valeur booléenne, elle renvoie `Void` dans la pratique car elle renvoie toujours la valeur `true`.

### Exemple

L'exemple suivant est tiré de l'exemple détaillé en intégralité dans l'entrée de la méthode `AsBroadcaster.initialize()`.

```
someObject.addListener(myListener1); // Register myListener1 as listener.
someObject.addListener(myListener2); // Register myListener2 as listener.
```

### Voir également

`initialize (méthode AsBroadcaster.initialize)`, `removeListener (méthode AsBroadcaster.removeListener)`

## broadcastMessage (méthode AsBroadcaster.broadcastMessage)

```
public broadcastMessage(eventName:String) : Void
```

Envoie un message d'événement à chaque objet de la liste d'écouteurs. Dès que l'objet d'écoute a reçu le message, Flash Player tente d'appeler une fonction du même nom sur l'objet d'écoute. Supposons que votre objet envoie un message d'événement tel que celui-ci :

```
obj.broadcastMessage("onAlert");
```

Une fois ce message reçu, Flash Player appelle une méthode intitulée `onAlert()` sur l'objet écouteur de réception.

*Remarque :* Vous pouvez transmettre des arguments à vos fonctions d'écouteur en incluant des arguments supplémentaires à la méthode `broadcastMessage()`. Tous les arguments apparaissant après le paramètre `eventName` sont reçus en tant qu'arguments par la méthode d'écouteur.

Cette méthode ne peut être appelée qu'à partir d'un objet ayant été initialisé à l'aide de la méthode `AsBroadcaster.initialize()`.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

### Paramètres

*`eventName`*`:String` - Le nom de l'événement à diffuser. Le nom des méthodes d'écouteur doit correspondre à ce paramètre afin de pouvoir recevoir l'événement de diffusion. Vous pouvez transmettre des arguments aux méthodes d'écouteur en incluant des arguments supplémentaires après `eventName`.

### Exemple

L'exemple suivant est tiré de l'exemple détaillé en intégralité dans l'entrée de la méthode `AsBroadcaster.initialize()` :

```
someObject.broadcastMessage("someEvent"); // Broadcast the "someEvent"
  message.
```

L'exemple suivant est tiré du deuxième exemple détaillé en intégralité dans l'entrée de la méthode `AsBroadcaster.initialize()`. Il indique comment envoyer des arguments aux méthodes d'écouteur.

```
someObject.broadcastMessage("someEvent", 3, "arbitrary string");
```

### Voir également

`initialize (méthode AsBroadcaster.initialize)`, `removeListener (méthode AsBroadcaster.removeListener)`

## initialize (méthode AsBroadcaster.initialize)

```
statique publique initialize(obj:Object) : Void
```

Ajoute une fonctionnalité de notification d'événements et de gestion des écouteurs à un objet donné. Il s'agit d'une méthode statique ; elle doit être appelée à l'aide de la classe AsBroadcaster (où `someObject` est le nom de l'objet à initialiser en tant que diffuseur d'événements) :

```
AsBroadcaster.initialize(someObject);
```

*Remarque :* Une erreur courante consiste à mettre en majuscule la deuxième lettre de *AsBroadcaster*. Lorsque vous appelez la méthode `AsBroadcaster.initialize()`, assurez-vous que la deuxième lettre est en minuscule. Toute faute d'orthographe apparaissant dans *AsBroadcaster* échoue sans indication.

Cette méthode ajoute la propriété `_listeners`, ainsi que les trois méthodes suivantes, à l'objet spécifié par le paramètre `obj` :

- `obj.addListener()`
- `obj.removeListener()`
- `obj.broadcastMessage()`

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

### Paramètres

*`obj`*`:Object` - Un objet faisant office d'objet de diffusion.

## Exemple

L'exemple suivant crée un objet générique, `someObject`, et le transforme en diffuseur d'événements. Les valeurs obtenues doivent correspondre aux chaînes affichées dans les deux instructions `trace()` :

```
var someObject:Object = new Object(); // Creates broadcast object.

var myListener1:Object = new Object(); // Creates listener object.
var myListener2:Object = new Object(); // Creates listener object.

myListener1.someEvent = function() { // Creates listener method.
  trace("myListener1 received someEvent");
}
myListener2.someEvent = function() { // Createz listener method.
  trace("myListener2 received someEvent");
}

AsBroadcaster.initialize(someObject); // Makes someObject an event
  broadcaster.
someObject.addListener(myListener1); // Registers myListener1 as listener.
someObject.addListener(myListener2); // Registers myListener2 as listener.
someObject.broadcastMessage("someEvent"); // Broadcasts the "someEvent"
  message.
```

L'exemple suivant indique comment transmettre des arguments supplémentaires à une méthode d'écouteur à l'aide de la méthode `broadcastMessage()`. Les valeurs obtenues doivent correspondre aux trois chaînes affichées dans les trois instructions `trace()`, qui incluent également les arguments transmis via la méthode `broadcastMessage()`.

```
var someObject:Object = new Object();

var myListener:Object = new Object();
myListener.someEvent = function(param1:Number, param2:String) {
  trace("myListener received someEvent");
  trace("param1: " + param1);
  trace("param2: " + param2);
}

AsBroadcaster.initialize(someObject);
someObject.addListener(myListener);
someObject.broadcastMessage("someEvent", 3, "arbitrary string");
```

# _listeners (propriété AsBroadcaster._listeners)

```
public _listeners : Array [lecture seule]
```

Une liste de références à tous les objets écouteurs enregistrés. Cette propriété est réservée à un usage interne uniquement et n'est pas destinée à une manipulation directe. Les objets sont ajoutés et supprimés dans ce tableau en appelant les méthodes `addListener()` et `removelistener()`.

Cette propriété ne peut être appelée qu'à partir d'un objet ayant été initialisé à l'aide de la méthode `AsBroadcaster.initialize()`.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

## Exemple

L'exemple suivant indique comment utiliser la propriété `length` pour déterminer le nombre d'objets écouteurs actuellement enregistrés auprès d'un diffuseur d'événements. Le code suivant fonctionne s'il est ajouté au premier exemple détaillé en intégralité dans la section Exemples de l'entrée `AsBroadcaster.initialize()` :

```
trace(someObject._listeners.length); // Output: 2
```

Pour les utilisateurs expérimentés, l'exemple suivant indique comment utiliser la propriété `_listeners` pour répertorier tous les écouteurs enregistrés avec un diffuseur d'événements, ainsi que toutes les propriétés de chaque objet écouteur. L'exemple suivant crée deux méthodes d'écouteur différentes pour le premier objet écouteur.

```
var someObject:Object = new Object(); // create broadcast object

var myListener1:Object = new Object(); // create listener object
var myListener2:Object = new Object(); // create listener object

myListener1.someEvent = function() { // create listener method
  trace("myListener1 received someEvent");
}
myListener1.anotherEvent = function() { // create another listener method
  trace("myListener1 received anotherEvent");
}
myListener2.someEvent = function() { // create listener method
  trace("myListener2 received someEvent");
}

AsBroadcaster.initialize(someObject); // make someObject an event
  broadcaster
someObject.addListener(myListener1); // register myListener1 as listener
someObject.addListener(myListener2); // register myListener2 as listener

var numListeners:Number = someObject._listeners.length; // get number of
  registered listeners
```

```
// cycle through all listener objects, listing all properties of each
  listener object
for (var i:Number = 0; i < numListeners; i++) {
  trace("Listener " + i + " listens for these events:");
  for (item in someObject._listeners[i]) {
    trace (" " + item + ": " + someObject._listeners[i][item]);
  }
}
```

### Voir également

`initialize (méthode AsBroadcaster.initialize)`

## removeListener (méthode AsBroadcaster.removeListener)

`public removeListener(listenerObj:Object) : Boolean`

Supprime un objet de la liste d'objets recevant des messages de notification d'événement.

Cette méthode ne peut être appelée qu'à partir d'un objet ayant été initialisé à l'aide de la méthode `AsBroadcaster.initialize()`.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

### Paramètres

*`listenerObj`*`:Object` - Le nom d'un objet écouteur enregistré pour recevoir des notifications d'événement émanant de l'objet de diffusion.

### Renvoie

`Boolean` - Renvoie `true` si l'objet écouteur est supprimé ; `false` sinon.

### Exemple

L'exemple suivant indique comment supprimer un écouteur de la liste des écouteurs enregistrés. Le code suivant fonctionne s'il est ajouté au premier exemple détaillé en intégralité dans la section Exemples de l'entrée `AsBroadcaster.initialize()`. Les instructions `trace()` sont incluses uniquement pour s'assurer que le nombre d'écouteurs enregistrés est réduit d'une unité après avoir appelé la méthode `removeListener()`.

```
trace(someObject._listeners.length); // Output: 2
someObject.removeListener(myListener1);
trace(someObject._listeners.length); // Output: 1
```

### Voir également

addListener (méthode AsBroadcaster.addListener), initialize (méthode AsBroadcaster.initialize)

# BevelFilter (flash.filters.BevelFilter)

```
Object
  |
  +-flash.filters.BitmapFilter
    |
    +-flash.filters.BevelFilter
```

public class *BevelFilter*
extends BitmapFilter

La classe BevelFilter vous permet d'ajouter un effet de biseau à divers objets dans Flash. L'effet de biseau donne aux objets tels que des boutons un aspect tridimensionnel. Vous pouvez personnaliser l'aspect du biseau grâce à différentes couleurs de soulignement et d'ombre, au montant de flou sur le biseau, à l'angle du biseau, au positionnement du biseau et à un effet de poinçonnage.

L'utilisation de filtres dépend de l'objet auquel vous appliquez le filtre.

- Pour appliquer des filtres aux clips, champs de texte et boutons lors de l'exécution, utilisez la propriété `filters`. Lorsque vous définissez la propriété `filters` d'un objet, celui-ci n'est pas modifié. En outre, vous pouvez l'annuler en supprimant la propriété `filters`.
- Pour appliquer des filtres aux occurrences BitmapData, utilisez la méthode `BitmapData.applyFilter()`. L'appel de `applyFilter` sur un objet BitmapData prend l'objet BitmapData source et l'objet de filtre, et génère une image filtrée.

Vous pouvez également appliquer des effets de filtre aux images et aux données vidéo pendant la programmation. Pour plus d'informations, consultez la documentation relative à la programmation.

Si vous appliquez un filtre à un clip ou à un bouton, la propriété `cacheAsBitmap` du clip ou du bouton est définie sur `true`. Si vous supprimez tous les filtres, la valeur d'origine de `cacheAsBitmap` est restaurée.

Ce filtre prend en charge le redimensionnement de la scène. Toutefois, le redimensionnement général, la rotation et l'inclinaison ne sont pas pris en charge. Si l'objet est redimensionné (si `_xscale` et `_yscale` ne sont pas définis sur 100 %), le filtre ne l'est pas. Le redimensionnement est effectué uniquement en cas de zoom avant sur la scène.

Aucun filtre n'est appliqué si l'image obtenue dépasse 2 880 pixels en hauteur ou en largeur. Si, par exemple, vous effectuez un zoom avant sur un clip volumineux auquel est appliqué un filtre, celui-ci est désactivé si l'image obtenue dépasse la limite de 2 880 pixels.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

## Voir également

`filters (propriété MovieClip.filters)`, `cacheAsBitmap (propriété MovieClip.cacheAsBitmap)`, `filters (propriété Button.filters)`, `cacheAsBitmap (propriété Button.cacheAsBitmap)`, `filters (propriété TextField.filters)`, `applyFilter (méthode BitmapData.applyFilter)`, `MovieClip`

## Résumé des propriétés

| Modificateurs | Propriété | Description |
|---|---|---|
| | `angle:Number` | L'angle du biseau. |
| | `blurX:Number` | Le montant de flou horizontal dans les pixels. |
| | `blurY:Number` | Le montant de flou vertical dans les pixels. |
| | `distance:Number` | La distance de décalage du biseau. |
| | `highlightAlpha:Number` | La valeur de transparence alpha de la couleur de soulignement. |
| | `highlightColor:Number` | La couleur de soulignement du biseau. |
| | `knockout:Boolean` | Applique un effet de poinçonnage (`true`), qui rend le remplissage de l'objet effectivement transparent et révèle la couleur d'arrière-plan du document. |
| | `quality:Number` | Le nombre d'applications du filtre. |
| | `shadowAlpha:Number` | La valeur de transparence alpha de la couleur d'ombre. |
| | `shadowColor:Number` | La couleur d'ombre du biseau. |
| | `strength:Number` | L'intensité de l'impression ou du recouvrement. |
| | `type:String` | Le type de biseau. |

*Propriétés héritées de la classe Object*

`constructeur (propriété Object.constructor)`, `__proto__ (Object.__proto__ property)`, `prototype (Object.prototype, propriété)`, `__resolve (Object.__resolve, propriété)`

## Résumé des constructeurs

| Signature | Description |
|---|---|
| `BevelFilter([distance:Number], [angle:Number], [highlightColor:Number], [highlightAlpha:Number], [shadowColor:Number], [shadowAlpha:Number], [blurX:Number], [blurY:Number], [strength:Number], [quality:Number], [type:String], [knockout:Boolean])` | Initialise une nouvelle occurrence BevelFilter avec les paramètres spécifiés. |

## Résumé de la méthode

| Modificateurs | Signature | Description |
|---|---|---|
| | `clone() : BevelFilter` | Renvoie une copie de cet objet filtre. |

*Méthodes héritées de la classe BitmapFilter*

`clone (méthode BitmapFilter.clone )`

*Méthodes héritées de la classe Object*

`addProperty (méthode Object.addProperty), hasOwnProperty (méthode Object.hasOwnProperty), isPropertyEnumerable (méthode Object.isPropertyEnumerable), isPrototypeOf (méthode Object.isPrototypeOf), registerClass (méthode Object.registerClass), toString (méthode Object.toString), unwatch (méthode Object.unwatch), valueOf (méthode Object.valueOf), watch (méthode Object.watch)`

# angle (propriété BevelFilter.angle)

```
public angle : Number
```

L'angle du biseau. Les valeurs valides sont comprises entre 0 et 360 degrés. La valeur par défaut est 45.

La valeur d'angle représente l'angle de la source lumineuse théorique projetée sur l'objet et détermine le positionnement de l'effet par rapport à l'objet. Si la distance est définie sur 0, l'effet n'est pas décalé par rapport à l'objet ; par conséquent, la propriété angle n'a pas d'effet.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

## Exemple

L'exemple suivant modifie la propriété `angle` d'une occurrence MovieClip existante (`rect`) lorsqu'un utilisateur clique sur celle-ci :

```
import flash.filters.BevelFilter;

var rect:MovieClip = createBevelRectangle("BevelDistance");
rect.onRelease = function() {
  var filter:BevelFilter = this.filters[0];
  filter.angle = 225;
  this.filters = new Array(filter);
}

function createBevelRectangle(name:String):MovieClip {
  var w:Number = 100;
  var h:Number = 100;
  var bgColor:Number = 0x00CC00;

  var rect:MovieClip = this.createEmptyMovieClip(name,
  this.getNextHighestDepth());
  rect.beginFill(bgColor);
  rect.lineTo(w, 0);
  rect.lineTo(w, h);
  rect.lineTo(0, h);
  rect.lineTo(0, 0);
  rect._x = 20;
  rect._y = 20;

  var filter:BevelFilter = new BevelFilter(5, 45, 0xFFFF00, .8, 0x0000FF,
  .8, 20, 20, 1, 3, "inner", false);
  rect.filters = new Array(filter);
  return rect;
}
```

# Constructeur BevelFilter

```
public BevelFilter([distance:Number], [angle:Number],
  [highlightColor:Number], [highlightAlpha:Number], [shadowColor:Number],
  [shadowAlpha:Number], [blurX:Number], [blurY:Number], [strength:Number],
  [quality:Number], [type:String], [knockout:Boolean])
```

Initialise une nouvelle occurrence BevelFilter avec les paramètres spécifiés.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

## Paramètres

*distance*:Number [facultatif] - La distance de décalage du biseau, en pixels (virgule flottante). La valeur par défaut est 4.

*angle*:Number [facultatif] - L'angle du biseau, de 0 à 360 degrés. La valeur par défaut est 45.

*highlightColor*:Number [facultatif] - La couleur de soulignement du biseau, *0xRRVVBB*. La valeur par défaut est 0xFFFFFF.

*highlightAlpha*:Number [facultatif] - La valeur de transparence alpha de la couleur de soulignement. Les valeurs valides sont comprises entre 0 et 1. Par exemple, 0,25 définit une valeur de transparence de 25 %. La valeur par défaut est 1.

*shadowColor*:Number [facultatif] - La couleur d'ombre du biseau, *0xRRVVBB*. La valeur par défaut est 0x000000.

*shadowAlpha*:Number [facultatif] - La valeur de transparence alpha de la couleur d'ombre. Les valeurs valides sont comprises entre 0 et 1. Par exemple, 0,25 définit une valeur de transparence de 25 %. La valeur par défaut est 1.

*blurX*:Number [facultatif] - Le montant de flou horizontal dans les pixels. Les valeurs valides sont comprises entre 0 et 255 (virgule flottante). La valeur par défaut est 4. Les valeurs qui sont une puissance de 2 (telles que 2, 4, 8, 16 et 32) sont optimisées de manière à ce que leur rendu soit obtenu plus rapidement que celui des autres valeurs.

*blurY*:Number [facultatif] - Le montant de flou vertical dans les pixels. Les valeurs valides sont comprises entre 0 et 255 (virgule flottante). La valeur par défaut est 4. Les valeurs qui sont une puissance de 2 (telles que 2, 4, 8, 16 et 32) sont optimisées de manière à ce que leur rendu soit obtenu plus rapidement que celui des autres valeurs.

*strength*:Number [facultatif] - L'intensité de l'impression ou du recouvrement. Plus la valeur est élevée, plus l'intensité des couleurs apparaît à l'impression et plus le contraste est important entre le biseau et l'arrière-plan. Les valeurs valides sont comprises entre 0 et 255. La valeur par défaut est 1.

*quality*:Number [facultatif] - Le nombre d'applications du filtre. La valeur par défaut est 1, ce qui correspond à un niveau de qualité faible. La valeur 2 correspond à une qualité moyenne et la valeur 3 à une qualité élevée.

*type*:String [facultatif] - Le type de biseau. Les valeurs valides sont "inner", "outer" et "full". La valeur par défaut est "inner".

*knockout*:Boolean [facultatif] - Applique un effet de poinçonnage (true), qui rend le remplissage de l'objet effectivement transparent et révèle la couleur d'arrière-plan du document. La valeur par défaut est false (pas de poinçonnage).

## Exemple

L'exemple suivant instancie une nouvelle occurrence BevelFilter et l'applique à l'occurrence MovieClip (rect) :

```
import flash.filters.BevelFilter;

var distance:Number = 5;
var angleInDegrees:Number = 45;
var highlightColor:Number = 0xFFFF00;
var highlightAlpha:Number = .8;
var shadowColor:Number = 0x0000FF;
var shadowAlpha:Number = .8;
var blurX:Number = 5;
var blurY:Number = 5;
var strength:Number = 5;
var quality:Number = 3;
var type:String = "inner";
var knockout:Boolean = false;

var filter:BevelFilter = new BevelFilter(distance,
                         angleInDegrees,
                         highlightColor,
                         highlightAlpha,
                         shadowColor,
                         shadowAlpha,
                         blurX,
                         blurY,
                         strength,
                         quality,
                         type,
                         knockout);

var rect:MovieClip = createRectangle(100, 100, 0x00CC00,
  "bevelFilterExample");
rect.filters = new Array(filter);

function createRectangle(w:Number, h:Number, bgColor:Number,
  name:String):MovieClip {
```

```
  var rect:MovieClip = this.createEmptyMovieClip(name,
  this.getNextHighestDepth());
  rect.beginFill(bgColor);
  rect.lineTo(w, 0);
  rect.lineTo(w, h);
  rect.lineTo(0, h);
  rect.lineTo(0, 0);
  rect._x = 20;
  rect._y = 20;
  return rect;
}
```

# blurX (propriété BevelFilter.blurX)

```
public blurX : Number
```

Le montant de flou horizontal dans les pixels. Les valeurs valides sont comprises entre 0 et 255 (virgule flottante). La valeur par défaut est 4. Les valeurs qui sont une puissance de 2 (telles que 2, 4, 8, 16 et 32) sont optimisées de manière à ce que leur rendu soit obtenu plus rapidement que celui des autres valeurs.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

## Exemple

L'exemple suivant modifie la propriété `blurX` de l'occurrence MovieClip existante (`rect`) lorsqu'un utilisateur clique sur celle-ci :

```
import flash.filters.BevelFilter;

var rect:MovieClip = createBevelRectangle("BevelBlurX");
rect.onRelease = function() {
  var filter:BevelFilter = this.filters[0];
  filter.blurX = 10;
  this.filters = new Array(filter);
}

function createBevelRectangle(name:String):MovieClip {
  var w:Number = 100;
  var h:Number = 100;
  var bgColor:Number = 0x00CC00;

  var rect:MovieClip = this.createEmptyMovieClip(name,
  this.getNextHighestDepth());
  rect.beginFill(bgColor);
  rect.lineTo(w, 0);
  rect.lineTo(w, h);
  rect.lineTo(0, h);
  rect.lineTo(0, 0);
  rect._x = 20;
```

```
  rect._y = 20;

  var filter:BevelFilter = new BevelFilter(5, 45, 0xFFFF00, .8, 0x0000FF,
  .8, 20, 20, 1, 3, "inner", false);
  rect.filters = new Array(filter);
  return rect;
}
```

# blurY (propriété BevelFilter.blurY)

```
public blurY : Number
```

Le montant de flou vertical dans les pixels. Les valeurs valides sont comprises entre 0 et 255 (virgule flottante). La valeur par défaut est 4. Les valeurs qui sont une puissance de 2 (telles que 2, 4, 8, 16 et 32) sont optimisées de manière à ce que leur rendu soit obtenu plus rapidement que celui des autres valeurs.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

## Exemple

L'exemple suivant modifie la propriété `blurY` de l'occurrence MovieClip existante (`rect`) lorsqu'un utilisateur clique sur celle-ci :

```
import flash.filters.BevelFilter;

var rect:MovieClip = createBevelRectangle("BevelBlurY");
rect.onRelease = function() {
  var filter:BevelFilter = this.filters[0];
  filter.blurY = 10;
  this.filters = new Array(filter);
}

function createBevelRectangle(name:String):MovieClip {
  var w:Number = 100;
  var h:Number = 100;
  var bgColor:Number = 0x00CC00;

  var rect:MovieClip = this.createEmptyMovieClip(name,
  this.getNextHighestDepth());
  rect.beginFill(bgColor);
  rect.lineTo(w, 0);
  rect.lineTo(w, h);
  rect.lineTo(0, h);
  rect.lineTo(0, 0);
  rect._x = 20;
  rect._y = 20;

  var filter:BevelFilter = new BevelFilter(5, 45, 0xFFFF00, .8, 0x0000FF,
  .8, 20, 20, 1, 3, "inner", false);
```

```
  rect.filters = new Array(filter);
  return rect;
}
```

# clone (méthode BevelFilter.clone)

```
public clone() : BevelFilter
```

Renvoie une copie de cet objet filtre.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

## Renvoie

`flash.filters.BevelFilter` - Une nouvelle occurrence BevelFilter dont toutes les propriétés sont identiques à celles de l'occurrence BevelFilter d'origine.

## Exemple

L'exemple suivant crée trois objets BevelFilter et les compare. Vous pouvez créer l'objet `filter_1` à l'aide du constructeur BevelFilter. Créez l'objet `filter_2` en lui attribuant une valeur égale à `filter_1`. Créez `clonedFilter` en clonant `filter_1`. Vous pouvez constater que, contrairement à `filter_2`, considéré comme étant égal à `filter_1`, `clonedFilter` ne l'est pas, même s'il contient les mêmes valeurs que `filter_1`.

```
import flash.filters.BevelFilter;

var filter_1:BevelFilter = new BevelFilter(5, 45, 0xFFFF00, .8, 0x0000FF,
  .8, 20, 20, 1, 3, "inner", false);
var filter_2:BevelFilter = filter_1;
var clonedFilter:BevelFilter = filter_1.clone();

trace(filter_1 == filter_2); // true
trace(filter_1 == clonedFilter); // false

for(var i in filter_1) {
  trace(">> " + i + ": " + filter_1[i]);
  // >> clone: [type Function]
  // >> type: inner
  // >> blurY: 20
  // >> blurX: 20
  // >> knockout: false
  // >> strength: 1
  // >> quality: 3
    // >> shadowAlpha: 0.8
  // >> shadowColor: 255
  // >> highlightAlpha: 0.8
  // >> highlightColor: 16776960
  // >> angle: 45
```

```
  // >> distance: 5
}

for(var i in clonedFilter) {
  trace(">> " + i + ": " + clonedFilter[i]);
  // >> clone: [type Function]
  // >> type: inner
  // >> blurY: 20
  // >> blurX: 20
  // >> knockout: false
  // >> strength: 1
  // >> quality: 3
  // >> shadowAlpha: 0.8
  // >> shadowColor: 255
  // >> highlightAlpha: 0.8
  // >> highlightColor: 16776960
  // >> angle: 45
  // >> distance: 5
}
```

Pour illustrer de manière plus détaillée les relations qui existent entre `filter_1`, `filter_2` et `clonedFilter`, l'exemple suivant modifie la propriété `knockout` de `filter_1`. La modification de la propriété `knockout` démontre que la méthode `clone()` crée une occurrence en fonction des valeurs de `filter_1` au lieu de se référer à ces valeurs.

```
import flash.filters.BevelFilter;

var filter_1:BevelFilter = new BevelFilter(5, 45, 0xFFFF00, .8, 0x0000FF,
  .8, 20, 20, 1, 3, "inner", false);
var filter_2:BevelFilter = filter_1;
var clonedFilter:BevelFilter = filter_1.clone();

trace(filter_1.knockout); // false
trace(filter_2.knockout); // false
trace(clonedFilter.knockout); // false

filter_1.knockout = true;

trace(filter_1.knockout); // true
trace(filter_2.knockout); // true
trace(clonedFilter.knockout); // false
```

## distance (propriété BevelFilter.distance)

```
public distance : Number
```

La distance de décalage du biseau. Les valeurs valides sont en pixels (virgule flottante). La valeur par défaut est 4.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

### Exemple

L'exemple suivant modifie la propriété `distance` de l'occurrence MovieClip existante (`rect`) lorsqu'un utilisateur clique sur celle-ci :

```
import flash.filters.BevelFilter;

var rect:MovieClip = createBevelRectangle("BevelDistance");
rect.onRelease = function() {
  var filter:BevelFilter = this.filters[0];
  filter.distance = 3;
  this.filters = new Array(filter);
}

function createBevelRectangle(name:String):MovieClip {
  var w:Number = 100;
  var h:Number = 100;
  var bgColor:Number = 0x00CC00;

  var rect:MovieClip = this.createEmptyMovieClip(name,
  this.getNextHighestDepth());
  rect.beginFill(bgColor);
  rect.lineTo(w, 0);
  rect.lineTo(w, h);
  rect.lineTo(0, h);
  rect.lineTo(0, 0);
  rect._x = 20;
  rect._y = 20;

  var filter:BevelFilter = new BevelFilter(5, 45, 0xFFFF00, .8, 0x0000FF,
  .8, 20, 20, 1, 3, "inner", false);
  rect.filters = new Array(filter);
  return rect;
}
```

## highlightAlpha (propriété BevelFilter.highlightAlpha)

```
public highlightAlpha : Number
```

La valeur de transparence alpha de la couleur de soulignement. La valeur spécifiée est une valeur normalisée comprise entre 0 et 1. Par exemple, 0,25 définit une valeur de transparence de 25 %. La valeur par défaut est 1.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

### Exemple

L'exemple suivant modifie la propriété `highlightAlpha` de l'occurrence MovieClip existante (`rect`) lorsqu'un utilisateur clique sur celle-ci :

```
import flash.filters.BevelFilter;
```

```
var rect:MovieClip = createBevelRectangle("BevelHighlightAlpha");
rect.onRelease = function() {
  var filter:BevelFilter = this.filters[0];
  filter.highlightAlpha = .2;
  this.filters = new Array(filter);
}

function createBevelRectangle(name:String):MovieClip {
  var w:Number = 100;
  var h:Number = 100;
  var bgColor:Number = 0x00CC00;

  var rect:MovieClip = this.createEmptyMovieClip(name,
  this.getNextHighestDepth());
  rect.beginFill(bgColor);
  rect.lineTo(w, 0);
  rect.lineTo(w, h);
  rect.lineTo(0, h);
  rect.lineTo(0, 0);
  rect._x = 20;
  rect._y = 20;

  var filter:BevelFilter = new BevelFilter(5, 45, 0xFFFF00, .8, 0x0000FF,
  .8, 20, 20, 1, 3, "inner", false);
  rect.filters = new Array(filter);
  return rect;
}
```

# highlightColor (propriété BevelFilter.highlightColor)

```
public highlightColor : Number
```

La couleur de soulignement du biseau. Les valeurs valides sont au format hexadécimal, *0xRRVVBB*. La valeur par défaut est 0xFFFFFF.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

## Exemple

L'exemple suivant modifie la propriété `highlightColor` de l'occurrence MovieClip existante (`rect`) lorsqu'un utilisateur clique sur celle-ci :

```
import flash.filters.BevelFilter;

var rect:MovieClip = createBevelRectangle("BevelHighlightColor");
rect.onRelease = function() {
  var filter:BevelFilter = this.filters[0];
  filter.highlightColor = 0x0000FF;
  this.filters = new Array(filter);
```

```
}

function createBevelRectangle(name:String):MovieClip {
  var w:Number = 100;
  var h:Number = 100;
  var bgColor:Number = 0x00CC00;

  var rect:MovieClip = this.createEmptyMovieClip(name,
  this.getNextHighestDepth());
  rect.beginFill(bgColor);
  rect.lineTo(w, 0);
  rect.lineTo(w, h);
  rect.lineTo(0, h);
  rect.lineTo(0, 0);
  rect._x = 20;
  rect._y = 20;

  var filter:BevelFilter = new BevelFilter(5, 45, 0xFFFF00, .8, 0x0000FF,
  .8, 20, 20, 1, 3, "inner", false);
  rect.filters = new Array(filter);
  return rect;
}
```

# knockout (propriété BevelFilter.knockout)

```
public knockout : Boolean
```

Applique un effet de poinçonnage (`true`), qui rend le remplissage de l'objet effectivement transparent et révèle la couleur d'arrière-plan du document. La valeur par défaut est `false` (pas de poinçonnage).

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

## Exemple

L'exemple suivant modifie la propriété `knockout` de l'occurrence MovieClip existante (`rect`) lorsqu'un utilisateur clique sur celle-ci :

```
import flash.filters.BevelFilter;

var rect:MovieClip = createBevelRectangle("BevelKnockout");
rect.onRelease = function() {
  var filter:BevelFilter = this.filters[0];
  filter.knockout = true;
  this.filters = new Array(filter);
}

function createBevelRectangle(name:String):MovieClip {
  var w:Number = 100;
  var h:Number = 100;
```

```
  var bgColor:Number = 0x00CC00;

  var rect:MovieClip = this.createEmptyMovieClip(name,
  this.getNextHighestDepth());
  rect.beginFill(bgColor);
  rect.lineTo(w, 0);
  rect.lineTo(w, h);
  rect.lineTo(0, h);
  rect.lineTo(0, 0);
  rect._x = 20;
  rect._y = 20;

  var filter:BevelFilter = new BevelFilter(5, 45, 0xFFFF00, .8, 0x0000FF,
  .8, 20, 20, 1, 3, "inner", false);
  rect.filters = new Array(filter);
  return rect;
}
```

## quality (propriété BevelFilter.quality)

```
public quality : Number
```

Le nombre d'applications du filtre. La valeur par défaut est 1, ce qui correspond à un niveau de qualité faible. La valeur 2 correspond à une qualité moyenne et la valeur 3 à une qualité élevée. Les rendus des filtres de valeurs inférieures sont obtenus plus rapidement.

Pour la plupart des applications, une valeur `quality` de 1, 2 ou 3 est suffisante. Bien que vous puissiez utiliser des valeurs numériques supplémentaires pouvant aller jusqu'à 15 pour obtenir d'autres effets, les rendus des valeurs plus élevées sont obtenus moins rapidement. Au lieu d'augmenter la valeur de `quality`, vous pouvez souvent obtenir un effet similaire dont le rendu est obtenu plus rapidement. Pour ce faire, il vous suffit d'augmenter les valeurs de `blurX` et `blurY`.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

### Exemple

L'exemple suivant modifie la propriété `quality` de l'occurrence MovieClip existante (`rect`) lorsqu'un utilisateur clique sur celle-ci :

```
import flash.filters.BevelFilter;

var rect:MovieClip = createBevelRectangle("BevelQuality");
rect.onRelease = function() {
  var filter:BevelFilter = this.filters[0];
  filter.quality = 1;
  this.filters = new Array(filter);
}
```

```
function createBevelRectangle(name:String):MovieClip {
  var w:Number = 100;
  var h:Number = 100;
  var bgColor:Number = 0x00CC00;

  var rect:MovieClip = this.createEmptyMovieClip(name,
  this.getNextHighestDepth());
  rect.beginFill(bgColor);
  rect.lineTo(w, 0);
  rect.lineTo(w, h);
  rect.lineTo(0, h);
  rect.lineTo(0, 0);
  rect._x = 20;
  rect._y = 20;

  var filter:BevelFilter = new BevelFilter(5, 45, 0xFFFF00, .8, 0x0000FF,
  .8, 20, 20, 1, 3, "inner", false);
  rect.filters = new Array(filter);
  return rect;
}
```

# shadowAlpha (propriété BevelFilter.shadowAlpha)

```
public shadowAlpha : Number
```

La valeur de transparence alpha de la couleur d'ombre. Cette valeur spécifiée est une valeur normalisée comprise entre 0 et 1. Par exemple, 0,25 définit une valeur de transparence de 25 %. La valeur par défaut est 1.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

## Exemple

L'exemple suivant modifie la propriété `shadowAlpha` de l'occurrence MovieClip existante (`rect`) lorsqu'un utilisateur clique sur celle-ci :

```
import flash.filters.BevelFilter;

var rect:MovieClip = createBevelRectangle("BevelShadowAlpha");
rect.onRelease = function() {
  var filter:BevelFilter = this.filters[0];
  filter.shadowAlpha = .2;
  this.filters = new Array(filter);
}

function createBevelRectangle(name:String):MovieClip {
  var w:Number = 100;
  var h:Number = 100;
  var bgColor:Number = 0x00CC00;
```

```
  var rect:MovieClip = this.createEmptyMovieClip(name,
  this.getNextHighestDepth());
  rect.beginFill(bgColor);
  rect.lineTo(w, 0);
  rect.lineTo(w, h);
  rect.lineTo(0, h);
  rect.lineTo(0, 0);
  rect._x = 20;
  rect._y = 20;

  var filter:BevelFilter = new BevelFilter(5, 45, 0xFFFF00, .8, 0x0000FF,
  .8, 20, 20, 1, 3, "inner", false);
  rect.filters = new Array(filter);
  return rect;
}
```

# shadowColor (propriété BevelFilter.shadowColor)

```
public shadowColor : Number
```

La couleur d'ombre du biseau. Les valeurs valides sont au format hexadécimal, *0xRRVVBB*. La valeur par défaut est 0x000000.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

## Exemple

L'exemple suivant modifie la propriété `shadowColor` de l'occurrence MovieClip existante (`rect`) lorsqu'un utilisateur clique sur celle-ci :

```
import flash.filters.BevelFilter;

var rect:MovieClip = createBevelRectangle("BevelShadowColor");
rect.onRelease = function() {
  var filter:BevelFilter = this.filters[0];
  filter.shadowColor = 0xFFFF00;
  this.filters = new Array(filter);
}

function createBevelRectangle(name:String):MovieClip {
  var w:Number = 100;
  var h:Number = 100;
  var bgColor:Number = 0x00CC00;

  var rect:MovieClip = this.createEmptyMovieClip(name,
  this.getNextHighestDepth());
  rect.beginFill(bgColor);
  rect.lineTo(w, 0);
  rect.lineTo(w, h);
  rect.lineTo(0, h);
```

```
  rect.lineTo(0, 0);
  rect._x = 20;
  rect._y = 20;

  var filter:BevelFilter = new BevelFilter(5, 45, 0xFFFF00, .8, 0x0000FF,
  .8, 20, 20, 1, 3, "inner", false);
  rect.filters = new Array(filter);
  return rect;
}
```

# strength (propriété BevelFilter.strength)

```
public strength : Number
```

L'intensité de l'impression ou du recouvrement. Les valeurs valides sont comprises entre 0 et 255. Plus la valeur est élevée, plus l'intensité des couleurs apparaît à l'impression et plus le contraste est important entre le biseau et l'arrière-plan. La valeur par défaut est 1.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

## Exemple

L'exemple suivant modifie la propriété `strength` de l'occurrence MovieClip existante (`rect`) lorsqu'un utilisateur clique sur celle-ci :

```
import flash.filters.BevelFilter;

var rect:MovieClip = createBevelRectangle("BevelStrength");
rect.onRelease = function() {
  var filter:BevelFilter = this.filters[0];
  filter.strength = 10;
  this.filters = new Array(filter);
}

function createBevelRectangle(name:String):MovieClip {
  var w:Number = 100;
  var h:Number = 100;
  var bgColor:Number = 0x00CC00;

  var rect:MovieClip = this.createEmptyMovieClip(name,
  this.getNextHighestDepth());
  rect.beginFill(bgColor);
  rect.lineTo(w, 0);
  rect.lineTo(w, h);
  rect.lineTo(0, h);
  rect.lineTo(0, 0);
  rect._x = 20;
  rect._y = 20;
```

```
  var filter:BevelFilter = new BevelFilter(5, 45, 0xFFFF00, .8, 0x0000FF,
  .8, 20, 20, 1, 3, "inner", false);
  rect.filters = new Array(filter);
  return rect;
}
```

# type (propriété BevelFilter.type)

```
public type : String
```

Le type de biseau. Les valeurs valides sont `"inner"`, `"outer"` et `"full"`.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

## Exemple

L'exemple suivant modifie la propriété `type` de l'occurrence MovieClip existante (`rect`) lorsqu'un utilisateur clique sur celle-ci :

```
import flash.filters.BevelFilter;

var rect:MovieClip = createBevelRectangle("BevelType");
rect.onRelease = function() {
  var filter:BevelFilter = this.filters[0];
  filter.type = "outer";
  this.filters = new Array(filter);
}

function createBevelRectangle(name:String):MovieClip {
  var w:Number = 100;
  var h:Number = 100;
  var bgColor:Number = 0x00CC00;

  var rect:MovieClip = this.createEmptyMovieClip(name,
  this.getNextHighestDepth());
  rect.beginFill(bgColor);
  rect.lineTo(w, 0);
  rect.lineTo(w, h);
  rect.lineTo(0, h);
  rect.lineTo(0, 0);
  rect._x = 20;
  rect._y = 20;

  var filter:BevelFilter = new BevelFilter(5, 45, 0xFFFF00, .8, 0x0000FF,
  .8, 20, 20, 1, 3, "inner", false);
  rect.filters = new Array(filter);
  return rect;
}
```

# BitmapData (flash.display.BitmapData)

```
Object
  |
  +-flash.display.BitmapData
```

public class ***BitmapData***
extends Object

La classe BitmapData vous permet de créer des images bitmap transparentes ou opaques dimensionnées de manière arbitraire et de les manipuler à votre guise lors de l'exécution.

Cette classe vous permet de séparer les opérations de rendu de bitmap dans les routines de mise à jour de l'affichage interne de Flash Player. En manipulant un objet BitmapData directement, vous pouvez créer des images très complexes sans utiliser de temps système supplémentaire par image résultant du retraçage du contenu des données vectorielles.

Les méthodes de la classe BitmapData prennent en charge de nombreux effets qui ne sont pas disponibles dans l'interface du filtre générique.

Un objet BitmapData contient un tableau de données de pixels. Ces données peuvent représenter un bitmap entièrement opaque ou entièrement transparent contenant des données de canal alpha. Chaque type d'objet BitmapData est stocké en tant que tampon converti en entiers 32 bits. Chaque entier 32 bits détermine les propriétés d'un pixel unique du bitmap.

Chaque entier 32 bits est une combinaison de quatre valeurs de canal de 8 bits (de zéro à 255) décrivant les valeurs de transparence alpha et de rouge, vert et bleu (ARVB) du pixel.

Les quatre canaux (rouge, vert, bleu et alpha) sont représentés sous forme de nombres lorsque vous les utilisez avec la méthode `BitmapData.copyChannel()` ou avec les propriétés `DisplacementMapFilter.componentX` et `DisplacementMapFilter.componentY`, comme suit :

- `1` (rouge)
- `2` (vert)
- `4` (bleu)
- `8` (alpha)

Vous pouvez associer des objets BitmapData à un objet MovieClip à l'aide de la méthode `MovieClip.attachBitmap()`.

Vous pouvez utiliser un objet BitmapData pour remplir une zone d'un clip à l'aide de la méthode `MovieClip.beginBitmapFill()`.

Les largeur et hauteur maximales d'un objet BitmapData sont de 2880 pixels.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

## Voir également

attachBitmap (méthode MovieClip.attachBitmap), beginBitmapFill (méthode MovieClip.beginBitmapFill)

## Résumé des propriétés

| Modificateurs | Propriété | Description |
|---|---|---|
| | height:Number [lecture seule] | La hauteur de l'image bitmap en pixels. |
| | rectangle:Rectangle [lecture seule] | Le rectangle qui délimite la taille et l'emplacement de l'image bitmap. |
| | transparent:Boolean [lecture seule] | Définit si l'image bitmap prend en charge la transparence par pixel. |
| | width:Number [lecture seule] | La largeur de l'image bitmap en pixels. |

*Propriétés héritées de la classe Object*

constructeur (propriété Object.constructor), __proto__ (Object.__proto__ property), prototype (Object.prototype, propriété), __resolve (Object.__resolve, propriété)

## Résumé des constructeurs

| Signature | Description |
|---|---|
| BitmapData(width:Number, height:Number, [transparent:Boolean], [fillColor:Number]) | Crée un objet BitmapData à la largeur et la hauteur spécifiées. |

Résumé de la méthode

| Modificateurs | Signature | Description |
|---|---|---|
| | `applyFilter(sourceBitmap:BitmapData, sourceRect:Rectangle, destPoint:Point, filter:BitmapFilter) : Number` | Prend une image source et un objet filtre et génère l'image filtrée. |
| | `clone() : BitmapData` | Renvoie un nouvel objet BitmapData, clone de l'occurrence d'origine avec une copie exacte du bitmap contenu. |
| | `colorTransform(rect:Rectangle, colorTransform:ColorTransform) : Void` | Définit les valeurs de couleur dans une zone spécifiée d'une image bitmap avec un objet ColorTransform. |
| | `copyChannel(sourceBitmap:BitmapData, sourceRect:Rectangle, destPoint:Point, sourceChannel:Number, destChannel:Number) : Void` | Transfère les données du canal d'un autre objet BitmapData ou de l'objet actuel vers un canal de l'objet BitmapData actuel. |
| | `copyPixels(sourceBitmap:BitmapData, sourceRect:Rectangle, destPoint:Point, [alphaBitmap:BitmapData], [alphaPoint:Point], [mergeAlpha:Boolean]) : Void` | Met en place une routine rapide permettant de manipuler les pixels de différentes images sans effets d'étirement, de rotation ou de couleur. |
| | `dispose() : Void` | Libère la mémoire utilisée pour stocker l'objet BitmapData. |

| Modificateurs | Signature | Description |
|---|---|---|
| | `draw(source:Object, [matrix:Matrix], [colorTransform:ColorTransform], [blendMode:Object], [clipRect:Rectangle], [smooth:Boolean]) : Void` | Dessine une image source ou un clip sur une image de destination avec la fonctionnalité de rendu vectoriel de Flash Player. |
| | `fillRect(rect:Rectangle, color:Number) : Void` | Remplit une zone rectangulaire de pixels avec une couleur ARVB spécifiée. |
| | `floodFill(x:Number, y:Number, color:Number) : Void` | Effectue une opération de peinture sur une image à partir de certaines coordonnées (*x*, *y*) et à l'aide d'une certaine couleur. |
| | `generateFilterRect(sourceRect:Rectangle, filter:BitmapFilter) : Rectangle` | Détermine le rectangle de destination affecté par l'appel de la méthode `applyFilter()`, en fonction d'un objet BitmapData, d'un rectangle source et d'un objet filtre spécifiés. |
| | `getColorBoundsRect(mask:Number, color:Number, [findColor:Boolean]) : Rectangle` | Détermine une zone rectangulaire qui regroupe tous les pixels d'une couleur spécifiée au sein de l'image bitmap. |
| | `getPixel(x:Number, y:Number) : Number` | Renvoie un entier représentant une valeur de pixels RVB à partir d'un objet BitmapData à un point spécifique (*x*, *y*). |
| | `getPixel32(x:Number, y:Number) : Number` | Renvoie une valeur de couleur ARVB qui contient des données de canal alpha, ainsi que les données RVB. |
| | `hitTest(firstPoint:Point, firstAlphaThreshold:Number, secondObject:Object, [secondBitmapPoint:Point], [secondAlphaThreshold:Number]) : Boolean` | Procède à la détection des clics au niveau des pixels entre une image bitmap et un point, un rectangle ou toute autre image bitmap. |

| Modificateurs | Signature | Description |
|---|---|---|
| static | loadBitmap(id:String) : BitmapData | Renvoie un nouvel objet BitmapData qui contient une version bitmap du symbole identifié par un ID de liaison spécifié dans la bibliothèque. |
| | merge(sourceBitmap:BitmapData, sourceRect:Rectangle, destPoint:Point, redMult:Number, greenMult:Number, blueMult:Number, alphaMult:Number) : Void | Procède au mélange canal par canal d'une image source vers une image de destination. |
| | noise(randomSeed:Number, [low:Number], [high:Number], [channelOptions:Number], [grayScale:Boolean]) : Void | Remplit une image avec des pixels représentant un bruit aléatoire. |
| | paletteMap(sourceBitmap:BitmapData, sourceRect:Rectangle, destPoint:Point, [redArray:Array], [greenArray:Array], [blueArray:Array], [alphaArray:Array]) : Void | Remappe les valeurs des canaux de couleur dans une image recevant jusqu'à quatre tableaux de données de palette de couleurs, un pour chaque canal. |
| | perlinNoise(baseX:Number, baseY:Number, numOctaves:Number, randomSeed:Number, stitch:Boolean, fractalNoise:Boolean, [channelOptions:Number], [grayScale:Boolean], [offsets:Object]) : Void | Génère une image de bruit Perlin. |

| Modificateurs | Signature | Description |
|---|---|---|
| | `pixelDissolve(sourceBitmap:BitmapData, sourceRect:Rectangle, destPoint:Point, [randomSeed:Number], [numberOfPixels:Number], [fillColor:Number]) : Number` | Procède à la dissolution de pixels, soit d'une image source vers une image de destination, soit en utilisant la même image. |
| | `scroll(x:Number, y:Number) : Void` | Fait défiler une image en fonction d'un certain montant en pixels (*x*, *y*). |
| | `setPixel(x:Number, y:Number, color:Number) : Void` | Définit la couleur d'un pixel unique d'un objet BitmapData. |
| | `setPixel32(x:Number, y:Number, color:Number) : Void` | Définit la couleur et les valeurs de transparence alpha d'un pixel unique d'un objet BitmapData. |
| | `threshold(sourceBitmap:BitmapData, sourceRect:Rectangle, destPoint:Point, operation:String, threshold:Number, [color:Number], [mask:Number], [copySource:Boolean]) : Number` | Teste les valeurs de pixels d'une image selon un seuil spécifié et définit les pixels qui réussissent le test sur de nouvelles valeurs de couleur. |

*Méthodes héritées de la classe Object*

`addProperty (méthode Object.addProperty), hasOwnProperty (méthode Object.hasOwnProperty), isPropertyEnumerable (méthode Object.isPropertyEnumerable), isPrototypeOf (méthode Object.isPrototypeOf), registerClass (méthode Object.registerClass), toString (méthode Object.toString), unwatch (méthode Object.unwatch), valueOf (méthode Object.valueOf), watch (méthode Object.watch)`

# applyFilter (méthode BitmapData.applyFilter)

```
public applyFilter(sourceBitmap:BitmapData, sourceRect:Rectangle,
  destPoint:Point, filter:BitmapFilter) : Number
```

Prend une image source et un objet filtre et génère l'image filtrée.

Cette méthode repose sur le comportement des objets filtres intégrés, contenant du code leur permettant de déterminer le rectangle de destination affecté par un rectangle source d'entrée.

Une fois le filtre appliqué, la taille de l'image obtenue peut être supérieure à celle de l'image d'entrée. Par exemple, si vous utilisez une classe BlurFilter pour rendre flou un rectangle source de (50,50,100,100) et un point de destination de (10,10), la zone modifiée sur l'image de destination est supérieure à (10,10,60,60) en raison du flou. Cela se produit en interne au cours de l'appel `applyFilter()`.

Si le paramètre `sourceRect` du paramètre `sourceBitmapData` est une zone intérieure, telle que (50,50,100,100) sur une image 200 x 200, le filtre utilise les pixels source hors du paramètre `sourceRect` pour générer le rectangle de destination.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

## Paramètres

*`sourceBitmap`*`:flash.display.BitmapData` - L'image bitmap d'entrée à utiliser. L'image source peut être un objet BitmapData différent ou peut faire référence à l'occurrence BitmapData actuelle.

*`sourceRect`*`:flash.geom.Rectangle` - Un rectangle qui définit la zone de l'image source à utiliser en tant qu'entrée.

*`destPoint`*`:flash.geom.Point` - Le point sur l'image de destination (l'occurrence BitmapData actuelle) correspondant au coin supérieur gauche du rectangle source.

*`filter`*`:flash.filters.BitmapFilter` - L'objet filtre utilisé pour effectuer l'opération de filtrage. Chaque type de filtre dispose d'exigences spécifiques, comme suit :

- *BlurFilter* - Ce filtre peut utiliser les images source et de destination opaques ou transparentes. Si les formats des images ne correspondent pas, la copie de l'image source effectuée lors du filtrage correspond au format de l'image de destination.
- *BevelFilter, DropShadowFilter, GlowFilter* - L'image de destination de ces filtres doit être transparente. L'appel de DropShadowFilter ou de GlowFilter permet de créer une image contenant les données de canal alpha de l'ombre portée ou du rayonnement. Il ne permet pas de créer l'ombre portée sur l'image de destination. Si vous utilisez l'un de ces filtres sur une image de destination opaque, une valeur de code d'erreur de -6 est renvoyée.
- *ConvolutionFilter* - Ce filtre peut utiliser les images source et de destination opaques ou transparentes.

- *ColorMatrixFilter* - Ce filtre peut utiliser les images source et de destination opaques ou transparentes.
- *DisplacementMapFilter* - Ce filtre peut utiliser les images source et de destination opaques ou transparentes à condition que leurs formats soient identiques.

## Renvoie

`Number` - Un nombre indiquant si le filtre a été appliqué avec succès. Si 0 est renvoyé, cela signifie que le filtre a été appliqué avec succès. Si un nombre négatif est renvoyé, cela signifie qu'une erreur s'est produite au cours de l'application du filtre.

## Exemple

L'exemple suivant indique comment appliquer un filtre de biseau à une occurrence BitmapData :

```
import flash.display.BitmapData;
import flash.filters.BevelFilter;
import flash.geom.Point;

var myBitmapData:BitmapData = new BitmapData(100, 80, true, 0xCCCCCCCC);

var mc:MovieClip = this.createEmptyMovieClip("mc",
  this.getNextHighestDepth());
mc.attachBitmap(myBitmapData, this.getNextHighestDepth());

var filter:BevelFilter = new BevelFilter(5, 45, 0xFFFF00, .8, 0x0000FF, .8,
  20, 20, 1, 3, "inner", false);

mc.onPress = function() {
  myBitmapData.applyFilter(myBitmapData, myBitmapData.rectangle, new
  Point(0, 0), filter);
}
```

## Voir également

BevelFilter (flash.filters.BevelFilter), BlurFilter (flash.filters.BlurFilter), ColorMatrixFilter (flash.filters.ColorMatrixFilter), ConvolutionFilter (flash.filters.ConvolutionFilter), DisplacementMapFilter (flash.filters.DisplacementMapFilter), DropShadowFilter (flash.filters.DropShadowFilter), GlowFilter (flash.filters.GlowFilter), filters (propriété MovieClip.filters)

# Constructeur BitmapData

```
public BitmapData(width:Number, height:Number, [transparent:Boolean],
  [fillColor:Number])
```

Crée un objet BitmapData à la largeur et la hauteur spécifiées. Si vous spécifiez une valeur pour le paramètre `fillColor`, chaque pixel du bitmap est défini sur cette couleur.

Par défaut, le bitmap créé est opaque, sauf si vous transmettez la valeur `true` au paramètre `transparent`. Une fois le bitmap opaque créé, vous ne pouvez pas le transformer en bitmap transparent. Chaque pixel d'un bitmap opaque utilise uniquement 24 bits d'informations de canal de couleur. Si vous définissez le bitmap sur `transparent`, chaque pixel utilise 32 bits d'informations de canal de couleur, y compris un canal de transparence alpha.

Les largeur et hauteur maximales d'un objet BitmapData sont de 2880 pixels. Si vous spécifiez une valeur de largeur ou de hauteur supérieure à 2880, la nouvelle occurrence n'est pas créée.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

## Paramètres

*`width`*`:Number` - La largeur de l'image bitmap en pixels.

*`height`*`:Number` - La hauteur de l'image bitmap en pixels.

*`transparent`*`:Boolean` [facultatif] - Spécifie si l'image bitmap prend en charge la transparence par pixel. La valeur par défaut est `true` (transparent). Pour créer un bitmap entièrement transparent, définissez la valeur du paramètre `transparent` sur `true` et celle du paramètre `fillColor` sur `0x00000000` (ou sur 0).

*`fillColor`*`:Number` [facultatif] - Une valeur de couleur ARVB 32 bits utilisée pour remplir la zone de l'image bitmap. La valeur par défaut est 0xFFFFFFFF (blanc uni).

## Exemple

L'exemple suivant crée un nouvel objet BitmapData. Les valeurs utilisées dans cet exemple sont les valeurs par défaut des paramètres `transparent` et `fillColor` ; vous pouvez appeler le constructeur sans ces paramètres et obtenir le même résultat.

```
import flash.display.BitmapData;

var width:Number = 100;
var height:Number = 80;
var transparent:Boolean = true;
var fillColor:Number = 0xFFFFFFFF;

var bitmap_1:BitmapData = new BitmapData(width, height, transparent,
  fillColor);

trace(bitmap_1.width); // 100
```

```
trace(bitmap_1.height); // 80
trace(bitmap_1.transparent); // true

var bitmap_2:BitmapData = new BitmapData(width, height);

trace(bitmap_2.width); // 100
trace(bitmap_2.height); // 80
trace(bitmap_2.transparent); // true
```

## clone (méthode BitmapData.clone)

```
public clone() : BitmapData
```

Renvoie un nouvel objet BitmapData, clone de l'occurrence d'origine avec une copie exacte du bitmap contenu.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

### Renvoie

`flash.display.BitmapData` - Un nouvel objet BitmapData identique à l'original.

### Exemple

L'exemple suivant crée trois objets BitmapData et les compare. Vous pouvez créer l'occurrence `bitmap_1` à l'aide du constructeur `BitmapData`. Créez l'occurrence `bitmap_2` en lui attribuant une valeur égale à `bitmap_1`. Créez l'occurrence `clonedBitmap` en clonant `bitmap_1`. Vous pouvez constater que, contrairement à `bitmap_2`, considéré comme étant égal à `bitmap_1`, `clonedBitmap` ne l'est pas, même s'il contient les mêmes valeurs que `bitmap_1`.

```
import flash.display.BitmapData;

var bitmap_1:BitmapData = new BitmapData(100, 80, false, 0x000000);
var bitmap_2:BitmapData = bitmap_1;
var clonedBitmap:BitmapData = bitmap_1.clone();

trace(bitmap_1 == bitmap_2); // true
trace(bitmap_1 == clonedBitmap); // false

for(var i in bitmap_1) {
  trace(">> " + i + ": " + bitmap_1[i]);
  // >> generateFilterRect: [type Function]
  // >> dispose: [type Function]
  // >> clone: [type Function]
  // >> copyChannel: [type Function]
  // >> noise: [type Function]
  // >> merge: [type Function]
  // >> paletteMap: [type Function]
```

```
  // >> hitTest: [type Function]
  // >> colorTransform: [type Function]
  // >> perlinNoise: [type Function]
  // >> getColorBoundsRect: [type Function]
  // >> floodFill: [type Function]
  // >> setPixel32: [type Function]
  // >> getPixel32: [type Function]
  // >> pixelDissolve: [type Function]
  // >> draw: [type Function]
  // >> threshold: [type Function]
  // >> scroll: [type Function]
  // >> applyFilter: [type Function]
  // >> copyPixels: [type Function]
  // >> fillRect: [type Function]
  // >> setPixel: [type Function]
  // >> getPixel: [type Function]
  // >> transparent: false
  // >> rectangle: (x=0, y=0, w=100, h=80)
  // >> height: 80
  // >> width: 100
}

for(var i in clonedBitmap) {
  trace(">> " + i + ": " + clonedBitmap[i]);
  // >> generateFilterRect: [type Function]
  // >> dispose: [type Function]
  // >> clone: [type Function]
  // >> copyChannel: [type Function]
  // >> noise: [type Function]
  // >> merge: [type Function]
  // >> paletteMap: [type Function]
  // >> hitTest: [type Function]
  // >> colorTransform: [type Function]
  // >> perlinNoise: [type Function]
  // >> getColorBoundsRect: [type Function]
  // >> floodFill: [type Function]
  // >> setPixel32: [type Function]
  // >> getPixel32: [type Function]
  // >> pixelDissolve: [type Function]
  // >> draw: [type Function]
  // >> threshold: [type Function]
  // >> scroll: [type Function]
  // >> applyFilter: [type Function]
  // >> copyPixels: [type Function]
  // >> fillRect: [type Function]
  // >> setPixel: [type Function]
  // >> getPixel: [type Function]
  // >> transparent: false
  // >> rectangle: (x=0, y=0, w=100, h=80)
  // >> height: 80
```

```
  // >> width: 100
}
```

Pour illustrer de manière plus détaillée les relations qui existent entre `bitmap_1`, `bitmap_2` et `clonedBitmap`, l'exemple suivant modifie la valeur de pixels au point (1, 1) de `bitmap_1`. La modification de la valeur de pixels au point (1, 1) montre que la méthode `clone()` crée une occurrence en fonction des valeurs de l'occurrence `bitmap_1` au lieu de se référer à ces valeurs.

```
import flash.display.BitmapData;

var bitmap_1:BitmapData = new BitmapData(100, 80, false, 0x000000);
var bitmap_2:BitmapData = bitmap_1;
var clonedBitmap:BitmapData = bitmap_1.clone();

trace(bitmap_1.getPixel32(1, 1)); // -16777216
trace(bitmap_2.getPixel32(1, 1)); // -16777216
trace(clonedBitmap.getPixel32(1, 1)); // -16777216

bitmap_1.setPixel32(1, 1, 0xFFFFFF);

trace(bitmap_1.getPixel32(1, 1)); // -1
trace(bitmap_2.getPixel32(1, 1)); // -1
trace(clonedBitmap.getPixel32(1, 1)); // -16777216
```

# colorTransform (méthode BitmapData.colorTransform)

```
public colorTransform(rect:Rectangle, colorTransform:ColorTransform) : Void
```

Définit les valeurs de couleur dans une zone spécifiée d'une image bitmap avec un objet ColorTransform. Si le rectangle correspond aux limites de l'image bitmap, cette méthode transforme les valeurs de couleur de l'image toute entière.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

## Paramètres

*rect*`:flash.geom.Rectangle` - Un objet Rectangle qui définit la zone de l'image dans laquelle l'objet ColorTransform est appliqué.

*colorTransform*`:flash.geom.ColorTransform` - Un objet ColorTransform décrivant les valeurs de transformation de couleur à appliquer.

## Exemple

L'exemple suivant indique comment appliquer une opération de transformation de couleurs à une occurrence BitmapData.

```
import flash.display.BitmapData;
```

```
import flash.geom.ColorTransform;

var myBitmapData:BitmapData = new BitmapData(100, 80, false, 0x00CCCCCC);

var mc:MovieClip = this.createEmptyMovieClip("mc",
  this.getNextHighestDepth());
mc.attachBitmap(myBitmapData, this.getNextHighestDepth());

mc.onPress = function() {
  myBitmapData.colorTransform(myBitmapData.rectangle, new
  ColorTransform(1, 0, 0, 1, 255, 0, 0, 0));
}
```

### Voir également

`ColorTransform (flash.geom.ColorTransform)`, `Rectangle (flash.geom.Rectangle)`

## copyChannel (méthode BitmapData.copyChannel)

```
public copyChannel(sourceBitmap:BitmapData, sourceRect:Rectangle,
  destPoint:Point, sourceChannel:Number, destChannel:Number) : Void
```

Transfère les données du canal d'un autre objet BitmapData ou de l'objet actuel vers un canal de l'objet BitmapData actuel. Toutes les données contenues dans les autres canaux de l'objet BitmapData de destination sont préservées.

La valeur du canal source et de destination peut être l'une des valeurs suivantes :

- `1` (rouge)
- `2` (vert)
- `4` (bleu)
- `8` (alpha)

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

### Paramètres

*`sourceBitmap`*`:flash.display.BitmapData` - L'image bitmap d'entrée à utiliser. L'image source peut être un objet BitmapData différent ou peut faire référence à l'objet BitmapData actuel.

*`sourceRect`*`:flash.geom.Rectangle` - L'objet Rectangle source. Si vous souhaitez uniquement copier les données de canal à partir d'une zone de taille inférieure sur le bitmap, spécifiez un rectangle source dont la taille est inférieure à la taille globale de l'objet BitmapData.

*destPoint*:flash.geom.Point - L'objet Point de destination qui représente le coin supérieur gauche de la zone rectangulaire dans laquelle les nouvelles données de canal sont placées. Si vous souhaitez copier les données de canal d'une zone vers une autre sur l'image de destination, spécifiez un point autre que (0,0).

*sourceChannel*:Number - Le canal source. Utilisez l'une des valeurs de l'ensemble (1,2,4,8), représentant respectivement les canaux rouge, vert, bleu et alpha.

*destChannel*:Number - Le canal de destination. Utilisez l'une des valeurs de l'ensemble (1,2,4,8), représentant respectivement les canaux rouge, vert, bleu et alpha.

### Exemple

L'exemple suivant indique comment copier un canal ARVB source à partir d'un objet BitmapData situé à un emplacement différent :

```
import flash.display.BitmapData;
import flash.geom.Rectangle;
import flash.geom.Point;

var myBitmapData:BitmapData = new BitmapData(100, 80, false, 0x00CCCCCC);

var mc:MovieClip = this.createEmptyMovieClip("mc",
  this.getNextHighestDepth());
mc.attachBitmap(myBitmapData, this.getNextHighestDepth());

mc.onPress = function() {
  myBitmapData.copyChannel(myBitmapData, new Rectangle(0, 0, 50, 80), new
  Point(51, 0), 3, 1);
}
```

### Voir également

Rectangle (flash.geom.Rectangle)

## copyPixels (méthode BitmapData.copyPixels)

```
public copyPixels(sourceBitmap:BitmapData, sourceRect:Rectangle,
  destPoint:Point, [alphaBitmap:BitmapData], [alphaPoint:Point],
  [mergeAlpha:Boolean]) : Void
```

Met en place une routine rapide permettant de manipuler les pixels de différentes images sans effets d'étirement, de rotation ou de couleur. Cette méthode copie une zone rectangulaire d'une image source dans une zone rectangulaire de taille identique au point de destination de l'objet BitmapData de destination.

Si elle inclut les paramètres `alphaBitmap` et `alphaPoint`, vous pouvez utiliser une image secondaire en tant que source alpha pour l'image source. Si l'image source contient des données alpha, les deux ensembles de données alpha sont utilisés pour composer des pixels de l'image source vers l'image de destination. Le paramètre `alphaPoint` est le point, sur l'image alpha, correspondant au coin supérieur gauche du rectangle source. Tous les pixels situés hors de l'intersection de l'image source et de l'image alpha ne sont pas copiés sur l'image de destination.

La propriété `mergeAlpha` contrôle si le canal alpha est utilisé ou non lorsqu'une image transparente est copiée sur une autre image transparente. Pour copier des pixels (sans utiliser de valeur alpha), il vous suffit de définir la propriété `mergeAlpha` sur `false`. Tous les pixels sont ensuite copiés de la source vers la destination. Par défaut, la propriété `mergeAlpha` est définie sur `true`.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

## Paramètres

*`sourceBitmap`*`:flash.display.BitmapData` - L'image bitmap d'entrée à partir de laquelle les pixels sont copiés. L'image source peut être une occurrence BitmapData différente ou peut faire référence à l'occurrence BitmapData actuelle.

*`sourceRect`*`:flash.geom.Rectangle` - Un rectangle qui définit la zone de l'image source à utiliser en tant qu'entrée.

*`destPoint`*`:flash.geom.Point` - Le point de destination représentant le coin supérieur gauche de la zone rectangulaire dans laquelle les nouveaux pixels sont placés.

*`alphaBitmap`*`:flash.display.BitmapData` [facultatif] - Une source de l'objet BitmapData alpha secondaire.

*`alphaPoint`*`:flash.geom.Point` [facultatif] - Le point, sur l'objet BitmapData alpha, correspondant au coin supérieur gauche du paramètre `sourceRect`.

*`mergeAlpha`*`:Boolean` [facultatif] - Une valeur booléenne. Pour utiliser le canal alpha, définissez la valeur sur `true`. Pour copier des pixels sans canal alpha, définissez la valeur sur `false`.

## Exemple

L'exemple suivant indique comment copier les pixels d'une occurrence BitmapData vers une autre.

```
import flash.display.BitmapData;
import flash.geom.Rectangle;
import flash.geom.Point;

var bitmapData_1:BitmapData = new BitmapData(100, 80, false, 0x00CCCCCC);
```

```
var bitmapData_2:BitmapData = new BitmapData(100, 80, false, 0x00FF0000);

var mc_1:MovieClip = this.createEmptyMovieClip("mc",
  this.getNextHighestDepth());
mc_1.attachBitmap(bitmapData_1, this.getNextHighestDepth());

var mc_2:MovieClip = this.createEmptyMovieClip("mc",
  this.getNextHighestDepth());
mc_2.attachBitmap(bitmapData_2, this.getNextHighestDepth());
mc_2._x = 101;

mc_1.onPress = function() {
  bitmapData_2.copyPixels(bitmapData_1, new Rectangle(0, 0, 50, 80), new
  Point(51, 0));
}

mc_2.onPress = function() {
  bitmapData_1.copyPixels(bitmapData_2, new Rectangle(0, 0, 50, 80), new
  Point(51, 0));
}
```

# dispose (méthode BitmapData.dispose)

```
public dispose() : Void
```

Libère la mémoire utilisée pour stocker l'objet BitmapData.

Lorsque cette méthode est appelée sur une image, la largeur et la hauteur de cette dernière sont définies sur 0. Une fois la mémoire d'un objet BitmapData libérée, les appels d'accès aux méthodes et propriétés effectués sur l'occurrence échouent et une valeur de -1 sera renvoyée.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

## Exemple

L'exemple suivant indique comment libérer de la mémoire sur une occurrence BitmapData, entraînant ainsi la suppression de l'occurrence.

```
import flash.display.BitmapData;

var myBitmapData:BitmapData = new BitmapData(100, 80, false, 0x00CCCCCC);

var mc:MovieClip = this.createEmptyMovieClip("mc",
  this.getNextHighestDepth());
mc.attachBitmap(myBitmapData, this.getNextHighestDepth());

mc.onPress = function() {
  myBitmapData.dispose();

  trace(myBitmapData.width); // -1
```

```
  trace(myBitmapData.height); // -1
  trace(myBitmapData.transparent); // -1
}
```

# draw (méthode BitmapData.draw)

```
public draw(source:Object, [matrix:Matrix],
  [colorTransform:ColorTransform], [blendMode:Object],
  [clipRect:Rectangle], [smooth:Boolean]) : Void
```

Dessine une image source ou un clip sur une image de destination avec la fonctionnalité de rendu vectoriel de Flash Player. Vous pouvez utiliser les objets Matrix, ColorTransform, BlendMode et un objet Rectangle de destination pour contrôler la qualité du rendu. Vous pouvez également spécifier si le bitmap doit être lissé lorsqu'il est dimensionné. Cela fonctionne uniquement si l'objet source est un objet BitmapData.

Cette méthode correspond directement au mode de traçage des objets à l'aide de la fonctionnalité de rendu vectoriel standard pour les objets dans l'interface de l'outil de programmation.

Un objet MovieClip source n'utilise pas ses transformations sur scène pour cet appel. Il est traité de la manière dont il apparaît dans la bibliothèque ou dans le fichier, sans transformation de matrice, de couleurs et sans mode de fondu. Si vous souhaitez dessiner le clip en utilisant ses propres propriétés de transformation, vous pouvez utiliser son objet Transform pour transmettre les diverses propriétés de transformation.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

## Paramètres

*source*:Object - L'objet BitmapData à dessiner.

*matrix*:flash.geom.Matrix [facultatif] - Un objet Matrix utilisé pour redimensionner, faire pivoter ou traduire les coordonnées du bitmap. Si aucun objet n'est fourni, l'image bitmap ne sera pas transformée. Définissez ce paramètre sur une matrice d'identité, créée à l'aide du constructeur `new Matrix()` par défaut, si vous devez le transmettre mais ne souhaitez pas transformer l'image.

*colorTransform*:flash.geom.ColorTransform [facultatif] - Un objet ColorTransform utilisé pour définir les valeurs de couleur du bitmap. Si aucun objet n'est fourni, les couleurs de l'image bitmap ne seront pas transformées. Définissez ce paramètre sur un objet ColorTransform, créé à l'aide du constructeur `new ColorTransform()` par défaut, si vous devez le transmettre mais ne souhaitez pas transformer l'image.

*blendMode*:Object [facultatif] - Un objet BlendMode.

*clipRect*:flash.geom.Rectangle [facultatif] - Un objet Rectangle. Si cette valeur n'est pas fournie, aucun découpage n'est effectué.

*smooth*:Boolean [facultatif] - Une valeur booléenne indiquant si un objet BitmapData doit être lissé lorsqu'il est dimensionné. La valeur par défaut est `false`.

## Exemple

L'exemple suivant indique comment dessiner à partir d'une occurrence MovieClip source sur un objet BitmapData.

```
import flash.display.BitmapData;
import flash.geom.Rectangle;
import flash.geom.Matrix;
import flash.geom.ColorTransform;

var myBitmapData:BitmapData = new BitmapData(100, 80, false, 0x00CCCCCC);

var mc_1:MovieClip = this.createEmptyMovieClip("mc",
  this.getNextHighestDepth());
mc_1.attachBitmap(myBitmapData, this.getNextHighestDepth());

var mc_2:MovieClip = createRectangle(50, 40, 0xFF0000);
mc_2._x = 101;

var myMatrix:Matrix = new Matrix();
myMatrix.rotate(Math.PI/2);

var translateMatrix:Matrix = new Matrix();
translateMatrix.translate(70, 15);

myMatrix.concat(translateMatrix);

var myColorTransform:ColorTransform = new ColorTransform(0, 0, 1, 1, 0, 0,
  255, 0);
var blendMode:String = "normal";

var myRectangle:Rectangle = new Rectangle(0, 0, 100, 80);
var smooth:Boolean = true;

mc_1.onPress = function() {
  myBitmapData.draw(mc_2, myMatrix, myColorTransform, blendMode,
  myRectangle, smooth);
}

function createRectangle(width:Number, height:Number,
  color:Number):MovieClip {
  var depth:Number = this.getNextHighestDepth();
  var mc:MovieClip = this.createEmptyMovieClip("mc_" + depth, depth);
  mc.beginFill(color);
```

```
  mc.lineTo(0, height);
  mc.lineTo(width, height);
  mc.lineTo(width, 0);
  mc.lineTo(0, 0);
  return mc;
}
```

# fillRect (méthode BitmapData.fillRect)

```
public fillRect(rect:Rectangle, color:Number) : Void
```

Remplit une zone rectangulaire de pixels avec une couleur ARVB spécifiée.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

## Paramètres

*rect*`:flash.geom.Rectangle` - La zone rectangulaire à remplir.

*color*`:Number` - La valeur de couleur ARVB qui remplit la zone. Les couleurs ARVB sont souvent spécifiées au format hexadécimal, par exemple 0xFF336699.

## Exemple

L'exemple suivant indique comment remplir une zone définie par un `Rectangle` dans un `BitmapData` à l'aide d'une couleur.

```
import flash.display.BitmapData;
import flash.geom.Rectangle;

var myBitmapData:BitmapData = new BitmapData(100, 80, false, 0x00CCCCCC);

var mc:MovieClip = this.createEmptyMovieClip("mc",
  this.getNextHighestDepth());
mc.attachBitmap(myBitmapData, this.getNextHighestDepth());

mc.onPress = function() {
  myBitmapData.fillRect(new Rectangle(0, 0, 50, 40), 0x00FF0000);
}
```

## Voir également

`Rectangle (flash.geom.Rectangle)`

# floodFill (méthode BitmapData.floodFill)

```
public floodFill(x:Number, y:Number, color:Number) : Void
```

Effectue une opération de peinture sur une image à partir de certaines coordonnées (*x*, *y*) et à l'aide d'une certaine couleur. La méthode `floodFill()` est similaire à l'outil Pot de peinture dans divers programmes de dessin. La couleur ARVB contient des informations alpha ainsi que des informations sur les couleurs.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

## Paramètres

*x*`:Number` - La coordonnée *x* de l'image.

*y*`:Number` - La coordonnée *y* de l'image.

*color*`:Number` - La couleur ARVB à utiliser pour le remplissage. Les couleurs ARVB sont souvent spécifiées au format hexadécimal, tel que 0xFF336699.

## Exemple

L'exemple suivant indique comment appliquer une couleur de peinture à une image à partir du point sur lequel l'utilisateur clique sur le bouton de la souris au sein d'un objet BitmapData.

```
import flash.display.BitmapData;
import flash.geom.Rectangle;

var myBitmapData:BitmapData = new BitmapData(100, 80, false, 0x00CCCCCC);

var mc:MovieClip = this.createEmptyMovieClip("mc",
  this.getNextHighestDepth());
mc.attachBitmap(myBitmapData, this.getNextHighestDepth());

myBitmapData.fillRect(new Rectangle(0, 0, 50, 40), 0x00FF0000);

mc.onPress = function() {
  myBitmapData.floodFill(_xmouse, _ymouse, 0x000000FF);
}
```

# generateFilterRect (méthode BitmapData.generateFilterRect)

```
public generateFilterRect(sourceRect:Rectangle, filter:BitmapFilter) :
  Rectangle
```

Détermine le rectangle de destination affecté par l'appel de la méthode `applyFilter()`, en fonction d'un objet BitmapData, d'un rectangle source et d'un objet filtre spécifiés.

Par exemple, un filtre de flou affecte normalement une zone dont la taille est supérieure à celle de l'image d'origine. Une image de 100 x 200 pixels filtrée par une occurrence BlurFilter par défaut, où `blurX = blurY = 4` génère un rectangle de destination de `(-2,-2,104,204)`. La méthode `generateFilterRect()` vous permet de déterminer la taille de ce rectangle de destination à l'avance de manière à ce que vous puissiez dimensionner l'image de destination de manière appropriée avant d'effectuer une opération de filtrage.

Certains filtres découpent leur rectangle de destination selon la taille de l'image source. Par exemple, un filtre `DropShadow` interne ne génère pas de résulat dont la taille est supérieure à celle de son image source. Dans cette interface API, l'objet BitmapData fait office de bornes source et n'est pas utilisé en tant que paramètre `rect`.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

## Paramètres

*`sourceRect`*`:flash.geom.Rectangle` - Un rectangle définissant la zone de l'image source à utiliser en tant qu'entrée.

*`filter`*`:flash.filters.BitmapFilter` - Un objet filtre utilisé pour calculer les dimensions du rectangle de destination.

## Renvoie

`flash.geom.Rectangle` - Un rectangle de destination dont les dimensions ont été calculées à l'aide d'une image, du paramètre `sourceRect` et d'un filtre.

## Exemple

L'exemple suivant indique comment déterminer le rectangle de destination affecté par la méthode `applyfilter()` :

```
import flash.display.BitmapData;
import flash.filters.BevelFilter;
import flash.geom.Rectangle;

var myBitmapData:BitmapData = new BitmapData(100, 80, true, 0xCCCCCCCC);
```

```
var filter:BevelFilter = new BevelFilter(5, 45, 0xFFFF00, .8, 0x0000FF, .8,
  20, 20, 1, 3, "outter", false);

var filterRect:Rectangle =
  myBitmapData.generateFilterRect(myBitmapData.rectangle, filter);

trace(filterRect); // (x=-31, y=-31, w=162, h=142)
```

# getColorBoundsRect (méthode BitmapData.getColorBoundsRect)

```
public getColorBoundsRect(mask:Number, color:Number, [findColor:Boolean]) :
  Rectangle
```

Détermine une zone rectangulaire qui regroupe tous les pixels d'une couleur spécifiée au sein de l'image bitmap.

Par exemple, si vous disposez d'une image source et souhaitez déterminer le rectangle de l'image qui contient un canal alpha différent de zéro, utilisez `{mask: 0xFF000000, color: 0x00000000}` en tant que paramètres. Les bornes de pixels ayant le paramètre `(value & mask) != color` sont recherchées dans l'image toute entière. Pour déterminer les espaces blancs autour d'une image, utilisez `{mask: 0xFFFFFFFF, color: 0xFFFFFFFF}` pour rechercher les bornes des pixels autres que blanc.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

## Paramètres

*`mask`*`:Number` - Une valeur de couleur hexadécimale.

*`color`*`:Number` - Une valeur de couleur hexadécimale.

*`findColor`*`:Boolean` [facultatif] - Si la valeur est définie sur `true`, renvoie les bornes d'une valeur de couleur dans une image. Si la valeur est définie sur `false`, renvoie les bornes sur lesquelles cette couleur n'existe pas dans une image. La valeur par défaut est `true`.

## Renvoie

`flash.geom.Rectangle` - La zone de l'image correspondant à la couleur spécifiée.

## Exemple

L'exemple suivant indique comment déterminer une zone rectangulaire qui regroupe tous les pixels d'une couleur spécifiée au sein de l'image bitmap :

```
import flash.display.BitmapData;
import flash.geom.Rectangle;

var myBitmapData:BitmapData = new BitmapData(100, 80, false, 0x00CCCCCC);
```

```
var mc:MovieClip = this.createEmptyMovieClip("mc",
  this.getNextHighestDepth());
mc.attachBitmap(myBitmapData, this.getNextHighestDepth());
myBitmapData.fillRect(new Rectangle(0, 0, 50, 40), 0x00FF0000);

mc.onPress = function() {
  var colorBoundsRect:Rectangle =
  myBitmapData.getColorBoundsRect(0x00FFFFFF, 0x00FF0000, true);
  trace(colorBoundsRect); // (x=0, y=0, w=50, h=40)
}
```

# getPixel (méthode BitmapData.getPixel)

```
public getPixel(x:Number, y:Number) : Number
```

Renvoie un entier représentant une valeur de pixels RVB à partir d'un objet BitmapData à un point spécifique (*x*, *y*). La méthode `getPixel()` renvoie une valeur de pixels non multipliée. Aucune information alpha n'est renvoyée.

Tous les pixels d'un objet BitmapData sont stockés en tant que valeurs de couleur prémultipliées. Les valeurs des canaux de couleur rouge, vert et bleu d'un pixel image prémultiplié sont déjà multipliées par les données alpha. Par exemple, si la valeur alpha est 0, les canaux RVB sont également définis sur 0, indépendamment de leurs valeurs non multipliées.

Cette perte de données peut entraîner certains problèmes lorsque vous effectuez ces opérations. Toutes les méthodes Flash Player utilisent et renvoient des valeurs non multipliées. La représentation des pixels interne est non multipliée avant d'être renvoyée en tant que valeur. Au cours d'une opération de définition, la valeur de pixels est prémultipliée avant de définir le pixel d'image brut.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

## Paramètres

*x*`:Number` - La coordonnée *x* du pixel.

*y*`:Number` - La coordonnée *y* du pixel.

## Renvoie

`Number` - Un nombre représentant une valeur de pixels RVB. Si les coordonnées (*x*, *y*) se trouvent à l'extérieur des limites de l'image, la valeur 0 est renvoyée.

### Exemple

L'exemple suivant utilise la méthode `getPixel()` pour récupérer la valeur RVB d'un pixel à un emplacement *x* et *y* spécifique.

```
import flash.display.BitmapData;

var myBitmapData:BitmapData = new BitmapData(100, 80, false, 0x00CCCCCC);

var mc:MovieClip = this.createEmptyMovieClip("mc",
  this.getNextHighestDepth());
mc.attachBitmap(myBitmapData, this.getNextHighestDepth());
trace("0x" + myBitmapData.getPixel(0, 0).toString(16)); // 0xcccccc
```

### Voir également

`getPixel32 (méthode BitmapData.getPixel32)`

## getPixel32 (méthode BitmapData.getPixel32)

```
public getPixel32(x:Number, y:Number) : Number
```

Renvoie une valeur de couleur ARVB qui contient des données de canal alpha, ainsi que les données RVB. Cette méthode est similaire à la méthode `getPixel()` qui renvoie une couleur RVB sans les données de canal alpha.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

### Paramètres

*x*`:Number` - La coordonnée *x* du pixel.

*y*`:Number` - La coordonnée *y* du pixel.

### Renvoie

`Number` - Un nombre représentant une valeur de pixels ARVB. Si les coordonnées (*x*, *y*) se trouvent à l'extérieur des limites de l'image, la valeur 0 est renvoyée. Si le bitmap créé est opaque et non transparent, cette méthode renvoie alors un code d'erreur de -1.

### Exemple

L'exemple suivant utilise la méthode `getPixel32()` pour récupérer la valeur ARVB d'un pixel à un emplacement *x* et *y* spécifique :

```
import flash.display.BitmapData;

var myBitmapData:BitmapData = new BitmapData(100, 80, true, 0xFFAACCEE);
```

```
var mc:MovieClip = this.createEmptyMovieClip("mc",
  this.getNextHighestDepth());
mc.attachBitmap(myBitmapData, this.getNextHighestDepth());

var alpha:String = (myBitmapData.getPixel32(0, 0) >> 24 &
  0xFF).toString(16);
trace(">> alpha: " + alpha); // ff

var red:String = (myBitmapData.getPixel32(0, 0) >> 16 & 0xFF).toString(16);
trace(">> red: " + red); // aa

var green:String = (myBitmapData.getPixel32(0, 0) >> 8 &
  0xFF).toString(16);
trace(">> green: " + green); // cc

var blue:String = (myBitmapData.getPixel32(0, 0) & 0xFF).toString(16);
trace(">> blue: " + blue); // ee

trace("0x" + alpha + red + green + blue); // 0xffaaccee
```

### Voir également

getPixel (méthode BitmapData.getPixel)

## height (propriété BitmapData.height)

```
public height : Number [lecture seule]
```

La hauteur de l'image bitmap en pixels.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

### Exemple

L'exemple suivant montre que la propriété `height` de l'occurrence `BitmapData` est en lecture seule car il essaie de la définir mais échoue :

```
import flash.display.BitmapData;

var myBitmapData:BitmapData = new BitmapData(100, 80, false, 0x00CCCCCC);

var mc:MovieClip = this.createEmptyMovieClip("mc",
  this.getNextHighestDepth());
mc.attachBitmap(myBitmapData, this.getNextHighestDepth());
trace(myBitmapData.height); // 80

myBitmapData.height = 999;
trace(myBitmapData.height); // 80
```

# hitTest (méthode BitmapData.hitTest)

```
public hitTest(firstPoint:Point, firstAlphaThreshold:Number,
  secondObject:Object, [secondBitmapPoint:Point],
  [secondAlphaThreshold:Number]) : Boolean
```

Procède à la détection des clics au niveau des pixels entre une image bitmap et un point, un rectangle ou toute autre image bitmap. Aucun étirement, aucune rotation ou autre transformation n'est pris en compte lorsque vous effectuez un test de recherche.

Si une image est opaque, elle est considérée comme étant un rectangle entièrement opaque pour cette méthode. Les deux images doivent être transparentes pour effectuer un test de recherche au niveau des pixels tenant compte de la transparence. Lorsque vous testez deux images transparentes, les paramètres de seuil alpha déterminent les valeurs des canaux alpha, comprises entre 0 et 255, considérées comme étant opaques.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

## Paramètres

*`firstPoint`*`:flash.geom.Point` - Un point qui définit l'emplacement d'un pixel dans l'occurrence BitmapData actuelle.

*`firstAlphaThreshold`*`:Number` - La valeur du canal alpha la plus élevée considéré comme étant opaque pour ce test de recherche.

*`secondObject`*`:Object` - Un objet Rectangle, Point ou BitmapData.

*`secondBitmapPoint`*`:flash.geom.Point` [facultatif] - Un point qui définit l'emplacement d'un pixel dans le deuxième objet BitmapData. Utilisez uniquement ce paramètre lorsque la valeur de `secondObject` est un objet BitmapData.

*`secondAlphaThreshold`*`:Number` [facultatif] - La valeur du canal alpha la plus élevée considéré comme étant opaque dans le deuxième objet BitmapData. Utilisez uniquement ce paramètre lorsque la valeur de `secondObject` est un objet BitmapData et que les deux objets BitmapData sont transparents.

## Renvoie

`Boolean` - Une valeur booléenne. En cas de correspondance, renvoie une valeur de `true` ; `false` dans le cas contraire.

## Exemple

L'exemple suivant indique comment déterminer si un objet BitmapData entre en collision avec `MovieClip`.

```
import flash.display.BitmapData;
import flash.geom.Point;
```

```
var myBitmapData:BitmapData = new BitmapData(100, 80, false, 0x00CCCCCC);

var mc_1:MovieClip = this.createEmptyMovieClip("mc",
  this.getNextHighestDepth());
mc_1.attachBitmap(myBitmapData, this.getNextHighestDepth());

var mc_2:MovieClip = createRectangle(20, 20, 0xFF0000);

var destPoint:Point = new Point(myBitmapData.rectangle.x,
  myBitmapData.rectangle.y);
var currPoint:Point = new Point();

mc_1.onEnterFrame = function() {
  currPoint.x = mc_2._x;
  currPoint.y = mc_2._y;
  if(myBitmapData.hitTest(destPoint, 255, currPoint)) {
    trace(">> Collision at y:" + currPoint.x + " and y:" + currPoint.y);
  }
}

mc_2.startDrag(true);

function createRectangle(width:Number, height:Number,
  color:Number):MovieClip {
  var depth:Number = this.getNextHighestDepth();
  var mc:MovieClip = this.createEmptyMovieClip("mc_" + depth, depth);
  mc.beginFill(color);
  mc.lineTo(0, height);
  mc.lineTo(width, height);
  mc.lineTo(width, 0);
  mc.lineTo(0, 0);
  return mc;
}
```

# loadBitmap (méthode BitmapData.loadBitmap)

```
statique publique loadBitmap(id:String) : BitmapData
```

Renvoie un nouvel objet BitmapData qui contient une version bitmap du symbole identifié par un ID de liaison spécifié dans la bibliothèque.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

## Paramètres

*id*:String - Un ID de liaison d'un symbole dans la bibliothèque.

### Renvoie

`flash.display.BitmapData` - Le symbole représenté sous forme d'image bitmap.

### Exemple

L'exemple suivant charge un bitmap avec l'ID de liaison `libraryBitmap` à partir de votre bibliothèque. Vous devez l'associer à un objet `MovieClip` pour lui attribuer une représentation visuelle.

```
import flash.display.BitmapData;

var linkageId:String = "libraryBitmap";
var myBitmapData:BitmapData = BitmapData.loadBitmap(linkageId);
trace(myBitmapData instanceof BitmapData); // true

var mc:MovieClip = this.createEmptyMovieClip("mc",
  this.getNextHighestDepth());
mc.attachBitmap(myBitmapData, this.getNextHighestDepth());
```

## merge (méthode BitmapData.merge)

```
public merge(sourceBitmap:BitmapData, sourceRect:Rectangle,
  destPoint:Point, redMult:Number, greenMult:Number, blueMult:Number,
  alphaMult:Number) : Void
```

Procède au mélange canal par canal d'une image source vers une image de destination. La formule suivante est utilisée pour chaque canal :

```
new red dest = (red source * redMult) + (red dest * (256 - redMult) / 256;
```

Les valeurs `redMult`, `greenMult`, `blueMult` et `alphaMult` sont les multiplicateurs utilisés pour chaque canal de couleur. Leur plage valide est comprise entre 0 et 256.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

### Paramètres

*`sourceBitmap`*`:flash.display.BitmapData` - L'image bitmap d'entrée à utiliser. L'image source peut être un objet BitmapData différent ou peut faire référence à l'objet BitmapData actuel.

*`sourceRect`*`:flash.geom.Rectangle` - Un rectangle qui définit la zone de l'image source à utiliser en tant qu'entrée.

*`destPoint`*`:flash.geom.Point` - Le point sur l'image de destination (l'occurrence BitmapData actuelle) correspondant au coin supérieur gauche du rectangle source.

*`redMult`*`:Number` - Un nombre par lequel la valeur de canal red doit être multipliée.

*`greenMult`*`:Number` - Un nombre par lequel la valeur de canal green doit être multipliée.

*blueMult*:Number - Un nombre par lequel la valeur de canal blue doit être multipliée.

*alphaMult*:Number - Un nombre par lequel la valeur de transparence alpha doit être multipliée.

### Exemple

L'exemple suivant indique comment fusionner deux parties d'une occurrence BitmapData.

```
import flash.display.BitmapData;
import flash.geom.Rectangle;
import flash.geom.Point;

var bitmapData_1:BitmapData = new BitmapData(100, 80, false, 0x00CCCCCC);
var bitmapData_2:BitmapData = new BitmapData(100, 80, false, 0x00FF0000);

var mc_1:MovieClip = this.createEmptyMovieClip("mc",
  this.getNextHighestDepth());
mc_1.attachBitmap(bitmapData_1, this.getNextHighestDepth());

var mc_2:MovieClip = this.createEmptyMovieClip("mc",
  this.getNextHighestDepth());
mc_2.attachBitmap(bitmapData_2, this.getNextHighestDepth());
mc_2._x = 101;

mc_1.onPress = function() {
  bitmapData_1.merge(bitmapData_2, new Rectangle(0, 0, 50, 40), new
  Point(25, 20), 128, 0, 0, 0);
}
```

## noise (méthode BitmapData.noise)

```
public noise(randomSeed:Number, [low:Number], [high:Number],
  [channelOptions:Number], [grayScale:Boolean]) : Void
```

Remplit une image avec des pixels représentant un bruit aléatoire.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

### Paramètres

*randomSeed*:Number - La valeur de départ aléatoire à utiliser.

*low*:Number [facultatif] - La valeur la plus faible à générer pour chaque canal (de 0 à 255). La valeur par défaut est 0.

*high*:Number [facultatif] - La valeur la plus élevée à générer pour chaque canal (de 0 à 255). La valeur par défaut est 255.

*channelOptions*:Number [facultatif] - Un nombre pouvant être une combinaison des quatre valeurs de canaux de couleur : 1 (rouge), 2 (vert), 4 (bleu) et 8(alpha). Vous pouvez utiliser l'opérateur logique OR | pour combiner les valeurs de canaux. La valeur par défaut est (1 | 2 | 4).

*grayScale*:Boolean [facultatif] - Une valeur booléenne. Si la valeur est true, une image en nuances de gris est créée en définissant tous les canaux de couleur sur la même valeur. La sélection du canal alpha n'est pas affectée en définissant ce paramètre sur true. La valeur par défaut est false.

## Exemple

L'exemple suivant indique comment appliquer un bruit pixel à un objet BitmapData pour un bitmap couleur et noir et blanc.

```
import flash.display.BitmapData;
import flash.geom.Rectangle;
import flash.geom.Point;

var bitmapData_1:BitmapData = new BitmapData(100, 80, false, 0x00CCCCCC);
var bitmapData_2:BitmapData = new BitmapData(100, 80, false, 0x00FF0000);

var mc_1:MovieClip = this.createEmptyMovieClip("mc",
  this.getNextHighestDepth());
mc_1.attachBitmap(bitmapData_1, this.getNextHighestDepth());

var mc_2:MovieClip = this.createEmptyMovieClip("mc",
  this.getNextHighestDepth());
mc_2.attachBitmap(bitmapData_2, this.getNextHighestDepth());
mc_2._x = 101;

mc_1.onPress = function() {
    bitmapData_1.merge(bitmapData_2, new Rectangle(0, 0, 50, 40), new
  Point(25, 20), 128, 0, 0, 0);
}

mc_1.onPress = function() {
    bitmapData_1.noise(128, 0, 255, 1, true);
}

mc_2.onPress = function() {
    bitmapData_2.noise(128);
}
```

# paletteMap (méthode BitmapData.paletteMap)

```
public paletteMap(sourceBitmap:BitmapData, sourceRect:Rectangle,
  destPoint:Point, [redArray:Array], [greenArray:Array], [blueArray:Array],
  [alphaArray:Array]) : Void
```

Remappe les valeurs des canaux de couleur dans une image recevant jusqu'à quatre tableaux de données de palette de couleurs, un pour chaque canal.

Flash Player utilise la formule suivante pour générer l'image résultante.

Une fois le calcul des valeurs rouge, vert, bleu et alpha effectué, celles-ci sont additionnées en effectuant une opération arithmétique standard s'articulant autour d'un entier 32 bits. Les valeurs de couleur rouge, vert, bleu et alpha de chaque pixel sont extraites dans une valeur comprise entre 0 et 255 distincte. Ces valeurs sont utilisées pour rechercher les nouvelles valeurs de couleur dans le tableau approprié : `redArray`, `greenArray`, `blueArray` et `alphaArray`. Chacun de ces quatre tableaux doit contenir 256 valeurs. Une fois les quatre nouvelles valeurs de canaux récupérées, elles sont combinées dans une valeur ARVB standard appliquée au pixel.

Les effets multi-canaux sont pris en charge par cette méthode. Chaque tableau d'entrée peut contenir des valeurs entières 32 bits ; aucun décalage ne se produit lorsque les valeurs sont additionnées. Cette routine ne prend pas en charge le verrouillage canal par canal.

Si aucun tableau n'est spécifié pour un canal, le canal de couleur est simplement copié de l'image source vers l'image de destination.

Vous pouvez utiliser cette méthode pour de nombreux effets, tel que le mappage de palette général (qui consiste à sélectionner un canal pour le convertir en image couleur de valeur false). Vous pouvez également utiliser cette méthode pour de nombreux algorithmes de manipulation de couleurs avancés, tels que gamma, courbes, niveaux et quantification.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

## Paramètres

*`sourceBitmap`*`:flash.display.BitmapData` - L'image bitmap d'entrée à utiliser. L'image source peut être un objet BitmapData différent ou peut faire référence à l'objet BitmapData actuel.

*`sourceRect`*`:flash.geom.Rectangle` - Un rectangle qui définit la zone de l'image source à utiliser en tant qu'entrée.

*`destPoint`*`:flash.geom.Point` - Le point sur l'image de destination (l'objet BitmapData actuel) correspondant au coin supérieur gauche du rectangle source.

*`redArray`*`:Array` [facultatif] - Si `redArray` n'est pas `null`, `red = redArray[source red value] else red = source rect value`.

*greenArray*:Array [facultatif] - Si greenArray n'est pas null, green = greenArray[source green value] else green = source green value.

*blueArray*:Array [facultatif] - Si blueArray n'est pas null, blue = blueArray[source blue value] else blue = source blue value.

*alphaArray*:Array [facultatif] - Si alphaArray n'est pas null, alpha = alphaArray[source alpha value] else alpha = source alpha value.

## Exemple

L'exemple suivant indique comment utiliser une palette pour convertir le rouge uni en vert, et le vert uni en rouge, dans un objet BitmapData unique.

```
import flash.display.BitmapData;
import flash.geom.Rectangle;
import flash.geom.Point;

var myBitmapData:BitmapData = new BitmapData(100, 80, false, 0x00FF0000);

var mc:MovieClip = this.createEmptyMovieClip("mc",
  this.getNextHighestDepth()); mc.attachBitmap(myBitmapData,
  this.getNextHighestDepth());

myBitmapData.fillRect(new Rectangle(51, 0, 50, 80), 0x0000FF00);

mc.onPress = function() {
  var redArray:Array = new Array(256);
  var greenArray:Array = new Array(256);

  for(var i = 0; i < 255; i++) {
    redArray[i] = 0x00000000;
    greenArray[i] = 0x00000000;
  }

  redArray[0xFF] = 0x0000FF00;
  greenArray[0xFF] = 0x00FF0000;

  myBitmapData.paletteMap(myBitmapData, new Rectangle(0, 0, 100, 40), new
  Point(0, 0), redArray, greenArray, null, null);
}
```

# perlinNoise (méthode BitmapData.perlinNoise)

```
public perlinNoise(baseX:Number, baseY:Number, numOctaves:Number,
  randomSeed:Number, stitch:Boolean, fractalNoise:Boolean,
  [channelOptions:Number], [grayScale:Boolean], [offsets:Object]) : Void
```

Génère une image de bruit Perlin.

L'algorithme permettant de générer un bruit Perlin interpole et combine des fonctions de bruit aléatoire individuelles (appelées octaves) en fonction unique qui génère un bruit aléatoire qui semble plus naturel. Tout comme les octaves musicales, la fréquence de chaque fonction d'octave est doublée par rapport à celle qui la précède. Le bruit Perlin est décrit comme étant une « somme de bruit fractale » car il combine plusieurs ensembles de données de bruit avec différents niveaux de détails.

Vous pouvez utiliser les fonctions de bruit Perlin pour simuler des phénomènes naturels et des paysages tels que le grain du bois, les nuages ou les chaînes de montagnes. Dans la plupart des cas, la sortie d'une fonction de bruit Perlin ne s'affiche pas directement mais est utilisée pour améliorer d'autres images et leur attribuer des variations pseudo-aléatoires.

Les fonctions de bruit aléatoire numériques simples produisent souvent des images aux points durs et contrastés. On ne retrouve pas souvent ce type de contraste dur dans la nature. L'algorithme de bruit Perlin mélange plusieurs fonctions de bruit ayant des niveaux de détails différents. Cet algorithme engendre des variations plus petites parmi les valeurs des pixels environnants.

*Remarque :* On doit l'algorithme de bruit Perlin à Ken Perlin qui a été le premier à le mettre au point après avoir créé des images graphiques sur ordinateur pour le film *Tron* sorti en 1982. Perlin a reçu un Oscar pour avoir mis au point la fonction de bruit Perlin en 1997.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

## Paramètres

*baseX*:Number - Fréquence à utiliser dans la direction *x*. Par exemple, pour générer un bruit adapté à une image de 64 x 128 pixels, définissez la valeur `baseX` sur 64.

*baseY*:Number - Fréquence à utiliser dans la direction *y*. Par exemple, pour générer un bruit adapté à une image de 64 x 128 pixels, définissez la valeur `baseY` sur 128.

*numOctaves*:Number - Nombre d'octaves ou fonctions de bruit individuelles à combiner pour créer ce bruit. Plus les nombres d'octaves sont élevés, plus les images créées sont détaillées. Les nombres d'octaves plus élevés nécessitent également un temps de traitement plus important.

*randomSeed*:Number - Le nombre de la valeur de départ aléatoire à utiliser. Si vous conservez tous les autres paramètres, vous pouvez générer différents résultats pseudo-aléatoires en variant la valeur de départ aléatoire. La fonction de bruit Perlin est une fonction de mappage et non une fonction permettant de générer des nombres aléatoires de valeur true. Elle permet donc de créer les mêmes résultats à chaque fois à partir de la même valeur de départ aléatoire.

*stitch*:Boolean - Une valeur booléenne. Si la valeur est `true`, la méthode tente de lisser les bords de transition de l'image pour créer des textures transparentes en vue du remplissage du bitmap en forme de mosaïque.

*fractalNoise*:Boolean - Une valeur booléenne. Si la valeur est `true`, la méthode génère un bruit fractal ; sinon, elle génère une turbulence. Les dégradés d'une image créée à partir d'une turbulence présentent des discontinuités visibles qui lui permettent de mieux appréhender les effets visuels plus saillants, comme les flammes ou les vagues de l'océan.

*channelOptions*:Number [facultatif] - Un nombre indiquant un ou plusieurs canaux de couleur. Pour créer cette valeur, vous pouvez utiliser ou combiner l'une des quatre constantes de canaux de couleur : `1` (rouge), `2` (vert), `4` (bleu) et `8`(alpha). Vous pouvez combiner les valeurs de canaux à l'aide de l'opérateur logique OR ; par exemple, vous pouvez combiner les canaux rouge et vert en utilisant le code suivant : `1 | 2`.

*grayScale*:Boolean [facultatif] - Une valeur booléenne. Si la valeur est `true`, une image en nuances de gris est créée en définissant les canaux de couleur rouge, vert et bleu sur des valeurs identiques. La valeur du canal alpha n'est pas affectée si cette valeur est définie sur `true`. La valeur par défaut est `false`.

*offsets*:Object [facultatif] - Un tableau de points correspondant aux décalages *x* et *y* pour chaque octave. En manipulant les valeurs de décalage, vous pouvez effectuer un défilement lisse d'une image perlinNoise. Chaque point du tableau de décalage affecte une fonction de bruit d'octave spécifique.

## Exemple

L'exemple suivant indique comment appliquer un bruit Perlin à un objet BitmapData.

```
import flash.display.BitmapData;

var bitmapData_1:BitmapData = new BitmapData(100, 80, false, 0x00CCCCCC);
var bitmapData_2:BitmapData = new BitmapData(100, 80, false, 0x00FF0000);

var mc_1:MovieClip = this.createEmptyMovieClip("mc",
  this.getNextHighestDepth());
mc_1.attachBitmap(bitmapData_1, this.getNextHighestDepth());

var mc_2:MovieClip = this.createEmptyMovieClip("mc",
  this.getNextHighestDepth());
mc_2.attachBitmap(bitmapData_2, this.getNextHighestDepth());
```

```
mc_2._x = 101;

mc_1.onPress = function() {
  var randomNum:Number = Math.floor(Math.random() * 10);
    bitmapData_1.perlinNoise(100, 80, 6, randomNum, false, true, 1, true,
  null);
}

mc_2.onPress = function() {
  var randomNum:Number = Math.floor(Math.random() * 10);
    bitmapData_2.perlinNoise(100, 80, 4, randomNum, false, false, 15,
  false, null);
}
```

# pixelDissolve (méthode BitmapData.pixelDissolve)

```
public pixelDissolve(sourceBitmap:BitmapData, sourceRect:Rectangle,
  destPoint:Point, [randomSeed:Number], [numberOfPixels:Number],
  [fillColor:Number]) : Number
```

Procède à la dissolution de pixels, soit d'une image source vers une image de destination, soit en utilisant la même image. Flash Player utilise une valeur `randomSeed` pour générer une dissolution de pixels aléatoire. La valeur renvoyée par la fonction doit être transmise lors des appels suivants pour poursuivre la dissolution de pixels jusqu'à ce qu'elle soit terminée.

Si l'image source diffère de l'image de destination, les pixels sont copiés de la source vers la destination à l'aide de toutes les propriétés. Cela permet de procéder à la dissolution d'une image vide dans une image entièrement remplie.

Si les images source et de destination sont équivalentes, les pixels sont remplis avec le paramètre `color`. Cela permet de procéder à la dissolution d'une image entièrement remplie. Dans ce mode, le paramètre `point` de destination est ignoré.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

## Paramètres

*`sourceBitmap`*`:flash.display.BitmapData` - L'image bitmap d'entrée à utiliser. L'image source peut être un objet BitmapData différent ou peut faire référence à l'occurrence BitmapData actuelle.

*`sourceRect`*`:flash.geom.Rectangle` - Un rectangle qui définit la zone de l'image source à utiliser en tant qu'entrée.

*`destPoint`*`:flash.geom.Point` - Le point sur l'image de destination (l'occurrence BitmapData actuelle) correspondant au coin supérieur gauche du rectangle source.

*`randomSeed`*`:Number` [facultatif] - La valeur de départ aléatoire à utiliser pour démarrer la dissolution de pixels. La valeur par défaut est 0.

*numberOfPixels*:Number [facultatif] - La valeur par défaut est égale à 1/30 de la zone source (largeur x hauteur).

*fillColor*:Number [facultatif] - Une valeur de couleur ARVB utilisée pour remplir les pixels dont la valeur source est égale à sa valeur de destination. La valeur par défaut est 0.

## Renvoie

Number - La nouvelle valeur de départ aléatoire à utiliser pour les prochains appels.

## Exemple

L'exemple suivant utilise pixelDissolve() pour convertir un objet BitmapData gris en rouge en procédant à la dissolution de 40 pixels à la fois jusqu'à ce que les 8000 pixels aient changé de couleurs :

```
import flash.display.BitmapData;
import flash.geom.Point;

var myBitmapData:BitmapData = new BitmapData(100, 80, false, 0x00CCCCCC);

var mc:MovieClip = this.createEmptyMovieClip("mc",
  this.getNextHighestDepth());
mc.attachBitmap(myBitmapData, this.getNextHighestDepth());

mc.onPress = function() {
  var randomNum:Number = Math.floor(Math.random() * 10);
  dissolve(randomNum);
}

var intervalId:Number;
var totalDissolved:Number = 0;
var totalPixels:Number = 8000;

function dissolve(randomNum:Number) {
  var newNum:Number = myBitmapData.pixelDissolve(myBitmapData,
  myBitmapData.rectangle, new Point(0, 0), randomNum, 40, 0x00FF0000);
  clearInterval(intervalId);
  if(totalDissolved < totalPixels) {
    intervalId = setInterval(dissolve, 10, newNum);
  }
  totalDissolved += 40;
}
```

# rectangle (propriété BitmapData.rectangle)

```
public rectangle : Rectangle [lecture seule]
```

Le rectangle qui délimite la taille et l'emplacement de l'image bitmap. Le haut et le côté gauche du rectangle sont définis sur 0 ; la largeur et la hauteur sont égales à la largeur et à la hauteur, en pixels, de l'objet BitmapData.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

## Exemple

L'exemple suivant montre que la propriété `rectangle` de l'occurrence `Bitmap` est en lecture seule car il essaie de la définir mais échoue :

```
import flash.display.BitmapData;
import flash.geom.Rectangle;

var myBitmapData:BitmapData = new BitmapData(100, 80, false, 0x00CCCCCC);

var mc:MovieClip = this.createEmptyMovieClip("mc",
  this.getNextHighestDepth());
mc.attachBitmap(myBitmapData, this.getNextHighestDepth());
trace(myBitmapData.rectangle); // (x=0, y=0, w=100, h=80)

myBitmapData.rectangle = new Rectangle(1, 2, 4, 8);
trace(myBitmapData.rectangle); // (x=0, y=0, w=100, h=80)
```

# scroll (méthode BitmapData.scroll)

```
public scroll(x:Number, y:Number) : Void
```

Fait défiler une image en fonction d'un certain montant en pixels (*x*, *y*). Les zones du bord situées hors de la zone de défilement demeurent inchangées.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

## Paramètres

*x*`:Number` - La valeur définie en vue du défilement horizontal.

*y*`:Number` - La valeur définie en vue du défilement vertical.

## Exemple

L'exemple suivant illustre comment faire défiler un objet BitmapData.

```
import flash.display.BitmapData;
import flash.geom.Rectangle;

var myBitmapData:BitmapData = new BitmapData(100, 80, false, 0x00CCCCCC);
```

```
var mc:MovieClip = this.createEmptyMovieClip("mc",
  this.getNextHighestDepth());
mc.attachBitmap(myBitmapData, this.getNextHighestDepth());

myBitmapData.fillRect(new Rectangle(0, 0, 25, 80), 0x00FF0000);

mc.onPress = function() {
  myBitmapData.scroll(25, 0);
}
```

## setPixel (méthode BitmapData.setPixel)

```
public setPixel(x:Number, y:Number, color:Number) : Void
```

Définit la couleur d'un pixel unique d'un objet BitmapData. La valeur de canal alpha actuelle du pixel de l'image est préservée au cours de cette opération. La valeur du paramètre de couleur RVB est traitée en tant que valeur de couleur non multipliée.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

### Paramètres

*x*`:Number` - La coordonnée *x* du pixel dont la valeur change.

*y*`:Number` - La coordonnée *y* du pixel dont la valeur change.

*color*`:Number` - La couleur RVB sur laquelle le pixel va être défini.

### Exemple

L'exemple suivant utilise la méthode `setPixel()` pour affecter la valeur RVB à un pixel à un emplacement *x* et *y* spécifique. Vous pouvez dessiner sur le bitmap créé sur la valeur 0x000000 en le faisant glisser.

```
import flash.display.BitmapData;

var myBitmapData:BitmapData = new BitmapData(100, 80, false, 0x00CCCCCC);

var mc:MovieClip = this.createEmptyMovieClip("mc",
  this.getNextHighestDepth());
mc.attachBitmap(myBitmapData, this.getNextHighestDepth());

mc.onPress = function() {
  this.onEnterFrame = sketch;
}

mc.onRelease = function() {
  delete this.onEnterFrame;
}
```

```
function sketch() {
  myBitmapData.setPixel(_xmouse, _ymouse, 0x000000);
}
```

### Voir également

getPixel (méthode BitmapData.getPixel), setPixel32 (méthode BitmapData.setPixel32)

## setPixel32 (méthode BitmapData.setPixel32)

```
public setPixel32(x:Number, y:Number, color:Number) : Void
```

Définit la couleur et les valeurs de transparence alpha d'un pixel unique d'un objet BitmapData. Cette méthode est similaire à la méthode `setPixel()` ; la principale différence réside dans le fait que la méthode `setPixel32()` adopte une valeur de couleur ARVB contenant les informations de canal alpha.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

### Paramètres

*x*:Number - La coordonnée *x* du pixel dont la valeur change.

*y*:Number - La coordonnée *y* du pixel dont la valeur change.

*color*:Number - La couleur ARVB sur laquelle le pixel va être défini. Si vous avez créé un bitmap opaque (non transparent), la partie de transparence alpha de cette valeur de couleur est ignorée.

### Exemple

L'exemple suivant utilise la méthode `setPixel32()` pour affecter une valeur ARVB à un pixel à un emplacement x et y spécifique. Vous pouvez dessiner sur le bitmap créé sur la valeur 0x000000 en l'absence de valeur alpha. Pour ce faire, il vous suffit d'appuyer sur le bouton de la souris et de le faire glisser.

```
import flash.display.BitmapData;

var myBitmapData:BitmapData = new BitmapData(100, 80, true, 0xFFCCCCCC);

var mc:MovieClip = this.createEmptyMovieClip("mc",
  this.getNextHighestDepth());
mc.attachBitmap(myBitmapData, this.getNextHighestDepth());

mc.onPress = function() {
  this.onEnterFrame = sketch;
}
```

```
mc.onRelease = function() {
  delete this.onEnterFrame;
}

function sketch() {
  myBitmapData.setPixel32(_xmouse, _ymouse, 0x00000000);
}
```

### Voir également

getPixel32 (méthode BitmapData.getPixel32), setPixel (méthode BitmapData.setPixel)

## threshold (méthode BitmapData.threshold)

```
public threshold(sourceBitmap:BitmapData, sourceRect:Rectangle,
  destPoint:Point, operation:String, threshold:Number, [color:Number],
  [mask:Number], [copySource:Boolean]) : Number
```

Teste les valeurs de pixels d'une image selon un seuil spécifié et définit les pixels qui réussissent le test sur de nouvelles valeurs de couleur. L'utilisation de la méthode `threshold()` vous permet d'isoler et de remplacer les gammes de couleurs d'une image et d'effectuer d'autres opérations logiques sur les pixels de l'image.

La logique du test de seuil est définie comme suit :

```
if ((pixelValue & mask) operation (threshold & mask)) then

    set pixel to color

 else

    if (copySource) then

       set pixel to corresponding pixel value from sourceBitmap
```

Le paramètre `operation` spécifie l'opérateur de comparaison à utiliser pour le test de seuil. Par exemple, si vous utilisez « == », vous pouvez isoler une valeur de couleur spécifique dans une image. Ou si vous utilisez `{operation: "<", mask: 0xFF000000, threshold: 0x7f000000, color: 0x00000000}`, vous pouvez définir tous les pixels de destination comme étant entièrement transparents lorsque la valeur alpha du pixel de l'image source est inférieure à 0x7F. Vous pouvez utiliser cette technique pour les transitions animées et les autres effets.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

## Paramètres

*sourceBitmap*:flash.display.BitmapData - L'image bitmap d'entrée à utiliser. L'image source peut être un objet BitmapData différent ou peut faire référence à l'occurrence BitmapData actuelle.

*sourceRect*:flash.geom.Rectangle - Un rectangle qui définit la zone de l'image source à utiliser en tant qu'entrée.

*destPoint*:flash.geom.Point - Le point sur l'image de destination (l'occurrence BitmapData actuelle) correspondant au coin supérieur gauche du rectangle source.

*operation*:String - L'un des opérateurs de comparaison suivants, transmis en tant que chaîne : "<", "<=", ">", ">=", "==", "!="

*threshold*:Number - La valeur par rapport à laquelle chaque pixel est testé afin de déterminer s'il se trouve dans la plage du seuil ou s'il le dépasse.

*color*:Number [facultatif] - La valeur de couleur sur laquelle un pixel est défini si le test de seuil réussit. La valeur par défaut est 0x00000000.

*mask*:Number [facultatif] - Le masque à utiliser pour isoler un composant de couleur. La valeur par défaut est 0xFFFFFFFF.

*copySource*:Boolean [facultatif] - Une valeur booléenne. Si la valeur est true, les valeurs de pixels de l'image source sont copiées vers la destination lorsque le test de seuil échoue. Si la valeur est false, l'image source n'est pas copiée lorsque le test de seuil échoue. La valeur par défaut est false.

## Renvoie

Number - Le nombre de pixels modifiés.

## Exemple

L'exemple suivant indique comment modifier la valeur de couleur des pixels dont la valeur de couleur est supérieure ou égale à un seuil donné.

```
import flash.display.BitmapData;
import flash.geom.Rectangle;
import flash.geom.Point;

var myBitmapData:BitmapData = new BitmapData(100, 80, false, 0x00CCCCCC);

var mc:MovieClip = this.createEmptyMovieClip("mc",
  this.getNextHighestDepth());
mc.attachBitmap(myBitmapData, this.getNextHighestDepth());

myBitmapData.fillRect(new Rectangle(0, 0, 50, 80), 0x00FF0000);
```

```
mc.onPress = function() {
  myBitmapData.threshold(myBitmapData, new Rectangle(0, 0, 100, 40), new
  Point(0, 0), ">=", 0x00CCCCCC, 0x000000FF, 0x00FF0000, false);
}
```

# transparent (propriété BitmapData.transparent)

```
public transparent : Boolean [lecture seule]
```

Définit si l'image bitmap prend en charge la transparence par pixel. Vous pouvez définir cette valeur uniquement lorsque vous créez un objet BitmapData en transmettant la valeur `true` au paramètre `transparent`. Après avoir créé un objet BitmapData, vous pouvez vérifier s'il prend en charge la transparence par pixel en déterminant si la valeur de la propriété `transparent` est `true`.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

## Exemple

L'exemple suivant montre que la propriété `transparent` de l'occurrence `Bitmap` est en lecture seule car il essaie de la définir mais échoue :

```
import flash.display.BitmapData;

var myBitmapData:BitmapData = new BitmapData(100, 80, false, 0x00CCCCCC);

var mc:MovieClip = this.createEmptyMovieClip("mc",
  this.getNextHighestDepth());
mc.attachBitmap(myBitmapData, this.getNextHighestDepth());
trace(myBitmapData.transparent); // false

myBitmapData.transparent = true;
trace(myBitmapData.transparent); // false
```

# width (propriété BitmapData.width)

```
public width : Number [lecture seule]
```

La largeur de l'image bitmap en pixels.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

## Exemple

L'exemple suivant montre que la propriété `width` de l'occurrence `Bitmap` est en lecture seule car il essaie de la définir mais échoue :

```
import flash.display.BitmapData;

var myBitmapData:BitmapData = new BitmapData(100, 80, false, 0x00CCCCCC);
```

```
var mc:MovieClip = this.createEmptyMovieClip("mc",
  this.getNextHighestDepth());
mc.attachBitmap(myBitmapData, this.getNextHighestDepth());
trace(myBitmapData.width); // 100

myBitmapData.width = 999;
trace(myBitmapData.width); // 100
```

# BitmapFilter (flash.filters.BitmapFilter)

```
Object
  |
  +-flash.filters.BitmapFilter
```

public class ***BitmapFilter***
extends Object

La classe de base BitmapFilter pour tous les effets de filtrage d'image.

Les classes BevelFilter, BlurFilter, ColorMatrixFilter, ConvolutionFilter, DisplacementMapFilter, DropShadowFilter, GlowFilter, GradientBevelFilter et GradientGlowFilter héritent de la classe BitmapFilter. Vous pouvez appliquer ces effets de filtre aux bitmaps et aux occurrences MovieClip.

Vous pouvez créer des sous-classes uniquement pour les précédentes sous-classes de la classe BitmapFilter.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

## Résumé des propriétés

*Propriétés héritées de la classe Object*

| |
|---|
| constructeur (propriété Object.constructor), __proto__ (Object.__proto__ property), prototype (Object.prototype, propriété), __resolve (Object.__resolve, propriété) |

### Résumé de la méthode

| Modificateurs | Signature | Description |
|---|---|---|
| | `clone() : BitmapFilter` | Renvoie un objet BitmapFilter qui est une copie exacte de l'objet BitmapFilter d'origine. |

*Méthodes héritées de la classe Object*

`addProperty (méthode Object.addProperty), hasOwnProperty (méthode Object.hasOwnProperty), isPropertyEnumerable (méthode Object.isPropertyEnumerable), isPrototypeOf (méthode Object.isPrototypeOf), registerClass (méthode Object.registerClass), toString (méthode Object.toString), unwatch (méthode Object.unwatch), valueOf (méthode Object.valueOf), watch (méthode Object.watch)`

## clone (méthode BitmapFilter.clone )

`public clone() : BitmapFilter`

Renvoie un objet BitmapFilter qui est une copie exacte de l'objet BitmapFilter d'origine.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

### Valeur renvoyée

`flash.filters.BitmapFilter` - Objet BitmapFilter.

# BlurFilter (flash.filters.BlurFilter)

```
Object
  |
  +-flash.filters.BitmapFilter
    |
    +-flash.filters.BlurFilter
```

`public class` ***BlurFilter***
`extends BitmapFilter`

La classe BlurFilter vous permet d'appliquer un effet visuel de flou à divers objets dans Flash. Un effet de flou adoucit les détails d'une image. Vous pouvez produire une panoplie de flous vous permettant d'obtenir un aspect doux n'ayant pas le focus, un flou gaussien ou encore un aspect voilé dont l'effet est identique à celui d'une image que l'on regarde à travers un verre semi-opaque. Quand la propriété `quality` de ce filtre est réglée sur 1, vous obtenez un aspect doux n'ayant pas le focus. Quand la propriété `quality` est réglée sur 3, il se rapproche d'un filtre de flou gaussien.

L'utilisation de filtres dépend de l'objet auquel vous appliquez le filtre.

- Pour appliquer des filtres aux clips, champs de texte et boutons lors de l'exécution, utilisez la propriété `filters`. Lorsque vous définissez la propriété `filters` d'un objet, celui-ci n'est pas modifié. En outre, vous pouvez l'annuler en supprimant la propriété `filters`.
- Pour appliquer des filtres aux occurrences BitmapData, utilisez la méthode `BitmapData.applyFilter()`. L'appel d'`applyFilter` sur un objet BitmapData utilise l'objet BitmapData d'origine ainsi que l'objet filtre pour générer une image filtrée.

Vous pouvez également appliquer des effets de filtre aux images et aux données vidéo pendant la programmation. Pour plus d'informations, consultez la documentation relative à la programmation.

Si vous appliquez un filtre à un clip ou à un bouton, la propriété `cacheAsBitmap` du clip ou du bouton est définie sur `true`. Si vous supprimez tous les filtres, la valeur d'origine de `cacheAsBitmap` est restaurée.

Ce filtre prend en charge le redimensionnement de la scène. Cependant, il ne prend pas en charge la mise à l'échelle, la rotation ni l'inclinaison. Si l'objet lui-même est redimensionné (`_xscale` et `_yscale` ne sont pas à 100%), l'effet de filtre n'est pas redimensionné. Le redimensionnement est effectué uniquement en cas de zoom avant sur la scène.

Un filtre ne peut s'appliquer si l'image résultante dépasse 2 880 pixels en largeur ou en hauteur. Par exemple, si vous faites un zoom avant sur un grand clip auquel un filtre est appliqué, le filtre est désactivé si l'image résultante dépasse la limite de 2880 pixels.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

## Voir également

`filters (propriété MovieClip.filters)`, `cacheAsBitmap (propriété MovieClip.cacheAsBitmap)`, `filters (propriété Button.filters)`, `cacheAsBitmap (propriété Button.cacheAsBitmap)`, `filters (propriété TextField.filters)`, `applyFilter (méthode BitmapData.applyFilter)`

## Résumé des propriétés

| Modificateurs | Propriété | Description |
|---|---|---|
| | `blurX:Number` | Le montant de flou horizontal. |
| | `blurY:Number` | Le montant de flou vertical. |
| | `quality:Number` | Le nombre d'applications du flou. |

*Propriétés héritées de la classe Object*

`constructeur (propriété Object.constructor), __proto__ (Object.__proto__ property), prototype (Object.prototype, propriété), __resolve (Object.__resolve, propriété)`

## Récapitulatif des constructeurs

| Signature | Description |
|---|---|
| `BlurFilter([blurX:Number], [blurY:Number], [quality:Number])` | Initialise le filtre avec les paramètres spécifiés. |

## Résumé de la méthode

| Modificateurs | Signature | Description |
|---|---|---|
| | `clone() : BlurFilter` | Renvoie une copie de cet objet filtre. |

*Méthodes héritées de la classe BitmapFilter*

`clone (méthode BitmapFilter.clone )`

*Méthodes héritées de la classe Object*

`addProperty (méthode Object.addProperty), hasOwnProperty (méthode Object.hasOwnProperty), isPropertyEnumerable (méthode Object.isPropertyEnumerable), isPrototypeOf (méthode Object.isPrototypeOf), registerClass (méthode Object.registerClass), toString (méthode Object.toString), unwatch (méthode Object.unwatch), valueOf (méthode Object.valueOf), watch (méthode Object.watch)`

# constructeur BlurFilter()

`public BlurFilter([blurX:Number], [blurY:Number], [quality:Number])`

Initialise le filtre avec les paramètres spécifiés. Les valeurs par défaut créent une image douce, n'ayant pas le focus.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

## Paramètres

*`blurX`*`:Number` [facultatif] - Quantité de flou à appliquer horizontalement. Les valeurs valides sont comprises entre 0 et 255 (valeur en virgule flottante). La valeur par défaut est 4. Les valeurs multiples de 2 (telles que 2, 4, 8, 16 et 32) sont optimisées pour donner un rendu plus rapide que les autres valeurs.

*`blurY`*`:Number` [facultatif] - Quantité de flou à appliquer verticalement. Les valeurs valides sont comprises entre 0 et 255 (valeur en virgule flottante). La valeur par défaut est 4. Les valeurs multiples de 2 (telles que 2, 4, 8, 16 et 32) sont optimisées pour donner un rendu plus rapide que les autres valeurs.

*`quality`*`:Number` [facultatif] - Nombre de fois que le filtre doit s'appliquer. La valeur par défaut est 1, ce qui correspond à la qualité inférieure. Une valeur de 2 est une qualité moyenne et une valeur de 3 est la qualité supérieure s'approchant du flou gaussien.

## Exemple

L'exemple suivant instancie un nouveau constructeur `BlurFilter` et l'applique à un rectangle plat :

```
import flash.filters.BlurFilter;
var rect:MovieClip = createRectangle(100, 100, 0x003366,
  "BlurFilterExample");

var blurX:Number = 30;
var blurY:Number = 30;
var quality:Number = 3;

var filter:BlurFilter = new BlurFilter(blurX, blurY, quality);
var filterArray:Array = new Array();
filterArray.push(filter);
rect.filters = filterArray;

function createRectangle(w:Number, h:Number, bgColor:Number,
  name:String):MovieClip {
  var mc:MovieClip = this.createEmptyMovieClip(name,
  this.getNextHighestDepth());
  mc.beginFill(bgColor);
  mc.lineTo(w, 0);
```

```
  mc.lineTo(w, h);
  mc.lineTo(0, h);
  mc.lineTo(0, 0);
  mc._x = 20;
  mc._y = 20;
  return mc;
}
```

# blurX (propriété BlurFilter.blurX)

```
public blurX : Number
```

Le montant de flou horizontal. Les valeurs valides sont comprises entre 0 et 255 (virgule flottante). La valeur par défaut est 4. Les valeurs multiples de 2 (telles que 2, 4, 8, 16 et 32) sont optimisées pour donner un rendu plus rapide que les autres valeurs.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

## Exemple

L'exemple suivant change la propriété `blurX` sur une occurrence de clip existant quand un utilisateur clique dessus.

```
import flash.filters.BlurFilter;
var mc:MovieClip = createBlurFilterRectangle("BlurFilterBlurX");
mc.onRelease = function() {
  var filter:BlurFilter = this.filters[0];
  filter.blurX = 200;
  this.filters = new Array(filter);
}

function createBlurFilterRectangle(name:String):MovieClip {
  var rect:MovieClip = this.createEmptyMovieClip(name,
  this.getNextHighestDepth());
  var w:Number = 100;
  var h:Number = 100;
  rect.beginFill(0x003366);
  rect.lineTo(w, 0);
  rect.lineTo(w, h);
  rect.lineTo(0, h);
  rect.lineTo(0, 0);
  rect._x = 20;
  rect._y = 20;

  var filter:BlurFilter = new BlurFilter(30, 30, 2);
  var filterArray:Array = new Array();
  filterArray.push(filter);
  rect.filters = filterArray;
  return rect;
}
```

# blurY (propriété BlurFilter.blurY)

```
public blurY : Number
```

Le montant de flou vertical. Les valeurs valides sont comprises entre 0 et 255 (virgule flottante). La valeur par défaut est 4. Les valeurs multiples de 2 (telles que 2, 4, 8, 16 et 32) sont optimisées pour donner un rendu plus rapide que les autres valeurs.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

## Exemple

L'exemple suivant change la propriété `blurY` sur une occurrence de clip existant quand un utilisateur clique dessus.

```
import flash.filters.BlurFilter;
var mc:MovieClip = createBlurFilterRectangle("BlurFilterBlurY");
mc.onRelease = function() {
  var filter:BlurFilter = this.filters[0];
  filter.blurY = 200;
  this.filters = new Array(filter);
}

function createBlurFilterRectangle(name:String):MovieClip {
  var rect:MovieClip = this.createEmptyMovieClip(name,
  this.getNextHighestDepth());
  var w:Number = 100;
  var h:Number = 100;
  rect.beginFill(0x003366);
  rect.lineTo(w, 0);
  rect.lineTo(w, h);
  rect.lineTo(0, h);
  rect.lineTo(0, 0);
  rect._x = 20;
  rect._y = 20;

  var filter:BlurFilter = new BlurFilter(30, 30, 2);
  var filterArray:Array = new Array();
  filterArray.push(filter);
  rect.filters = filterArray;
  return rect;
}
```

# clone (méthode BitmapFilter.clone )

```
public clone() : BlurFilter
```

Renvoie une copie de cet objet filtre.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

## Valeur renvoyée

`flash.filters.BlurFilter` - Nouvelle occurrence BlurFilter dont toutes les propriétés sont identiques à celles de l'occurrence BlurFilter d'origine.

## Exemple

L'exemple suivant crée trois objets BlurFilter et les compare. Vous pouvez créer l'objet `filter_1` à l'aide du constructeur `BlurFilter`. Vous pouvez créer l'objet `filter_2` en lui attribuant des réglages égaux à ceux de `filter_1`. Vous pouvez créer l'objet `clonedFilter` en clonant `filter_1`. Veuillez noter que `filter_2` est considéré comme égal à `filter_1`, `clonedFilter` ne l'est pas, même s'il contient les mêmes valeurs que `filter_1`.

```
import flash.filters.BlurFilter;

var filter_1:BlurFilter = new BlurFilter(30, 30, 2);
var filter_2:BlurFilter = filter_1;
var clonedFilter:BlurFilter = filter_1.clone();

trace(filter_1 == filter_2); // true
trace(filter_1 == clonedFilter); // false

for(var i in filter_1) {
  trace(">> " + i + ": " + filter_1[i]);
  // >> clone: [type Function]
  // >> quality: 2
  // >> blurY: 30
  // >> blurX: 30
}

for(var i in clonedFilter) {
  trace(">> " + i + ": " + clonedFilter[i]);
  // >> clone: [type Function]
  // >> quality: 2
  // >> blurY: 30
  // >> blurX: 30
}
```

Pour démontrer davantage les relations entre `filter_1`, `filter_2`, et `clonedFilter`, l'exemple suivant modifie la propriété `quality` de `filter_1`. La modification de `quality` démontre que la méthode `clone()` crée une nouvelle occurrence basée sur les valeurs de `filter_1` au lieu de faire référence à ces valeurs.

```
import flash.filters.BlurFilter;

var filter_1:BlurFilter = new BlurFilter(30, 30, 2);
var filter_2:BlurFilter = filter_1;
var clonedFilter:BlurFilter = filter_1.clone();

trace(filter_1.quality); // 2
```

```
trace(filter_2.quality); // 2
trace(clonedFilter.quality); // 2

filter_1.quality = 1;

trace(filter_1.quality); // 1
trace(filter_2.quality); // 1
trace(clonedFilter.quality); // 2
```

# quality (propriété BlurFilter.quality)

```
public quality : Number
```

Le nombre d'applications du flou. Les valeurs valides sont 0 à 15. La valeur par défaut est 1, ce qui équivaut à une qualité inférieure. Une valeur de 2 est une qualité moyenne. Une valeur de 3 est une haute qualité et se rapproche d'un flou gaussien.

Pour la plupart des applications, une valeur de `quality` de 1, 2, ou 3 est suffisante. Vous pouvez cependant utiliser les valeurs numériques jusqu'à 15 pour augmenter le nombre de fois où le flou est appliqué pour obtenir un effet de flou plus important. Mais les valeurs les plus hautes donnent un rendu plus lent. Plutôt que d'augmenter la valeur de `quality`, vous pouvez souvent obtenir un effet similaire avec un rendu plus rapide en augmentant simplement les valeurs de `blurX` et `blurY`.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

## Exemple

L'exemple suivant crée un rectangle et lui applique un filtre de flou avec une valeur de `quality` de 1. Lorsque vous cliquez sur le rectangle, la valeur de `quality` augmente à 3, et le rectangle devient plus flou.

```
import flash.filters.BlurFilter;
var mc:MovieClip = createBlurFilterRectangle("BlurFilterQuality");
mc.onRelease = function() {
  var filter:BlurFilter = this.filters[0];
  filter.quality = 3;
  this.filters = new Array(filter);
}

function createBlurFilterRectangle(name:String):MovieClip {
  var rect:MovieClip = this.createEmptyMovieClip(name,
  this.getNextHighestDepth());
  var w:Number = 100;
  var h:Number = 100;
  rect.beginFill(0x003366);
  rect.lineTo(w, 0);
  rect.lineTo(w, h);
  rect.lineTo(0, h);
```

```
  rect.lineTo(0, 0);
  rect._x = 20;
  rect._y = 20;

  var filter:BlurFilter = new BlurFilter(30, 30, 1);
  var filterArray:Array = new Array();
  filterArray.push(filter);
  rect.filters = filterArray;
  return rect;
}
```

# Boolean

```
Object
  |
  +-Boolean
```

public class *Boolean*
extends Object

La classe Boolean est une enveloppe disposant des mêmes fonctionnalités que l'objet JavaScript Boolean standard. Utilisez la classe Boolean pour extraire le type de données primitif ou la représentation d'un objet booléen sous forme de chaîne.

Vous devez utiliser le constructeur new `Boolean()` pour créer un objet Boolean avant d'appeler ses méthodes.

**Disponibilité :** ActionScript 1.0 ; Flash Player 5

## Résumé des propriétés

*Propriétés héritées de la classe Object*

| |
|---|
| constructeur (propriété Object.constructor), __proto__ (Object.__proto__ property), prototype (Object.prototype, propriété), __resolve (Object.__resolve, propriété) |

## Résumé des constructeurs

| Signature | Description |
|---|---|
| `Boolean([value:Object])` | Crée un objet Boolean. |

### Résumé de la méthode

| Modificateurs | Signature | Description |
|---|---|---|
| | `toString() : String` | Renvoie la représentation de l'objet booléen sous forme de chaîne (`"true"` ou `"false"`). |
| | `valueOf() : Boolean` | Renvoie `true` si le type de valeur primitif de l'objet booléen spécifié est true ; `false` sinon. |

*Méthodes héritées de la classe Object*

`addProperty (méthode Object.addProperty)`, `hasOwnProperty (méthode Object.hasOwnProperty)`, `isPropertyEnumerable (méthode Object.isPropertyEnumerable)`, `isPrototypeOf (méthode Object.isPrototypeOf)`, `registerClass (méthode Object.registerClass)`, `toString (méthode Object.toString)`, `unwatch (méthode Object.unwatch)`, `valueOf (méthode Object.valueOf)`, `watch (méthode Object.watch)`

## constructeur Boolean()

```
public Boolean([value:Object])
```

Crée un objet Boolean. Si vous omettez le paramètre `value`, l'objet booléen est initialisé avec une valeur `false`. Si vous spécifiez une valeur pour le paramètre `value`, la méthode l'évalue et renvoie le résultat sous forme de valeur booléenne conformément aux règles de la fonction globale `Boolean()`.

**Disponibilité :** ActionScript 1.0 ; Flash Player 5

### Paramètres

*`value`*`:Object` [facultatif] - Toute expression. La valeur par défaut est `false`.

### Exemple

Le code suivant crée un nouvel objet booléen vide intitulé `myBoolean` :

```
var myBoolean:Boolean = new Boolean();
```

## toString (méthode Boolean.toString)

```
public toString() : String
```

Renvoie la représentation de l'objet booléen sous forme de chaîne (`"true"` ou `"false"`).

**Disponibilité :** ActionScript 1.0 ; Flash Player 5

### Valeur renvoyée

`String` - Chaîne; `"true"` ou `"false"`.

### Exemple

Cet exemple crée une variable de type Boolean et utilise la méthode `toString()` pour convertir la valeur en chaîne à utiliser dans l'instruction trace :

```
var myBool:Boolean = true;
trace("The value of the Boolean myBool is: " + myBool.toString());
myBool = false;
trace("The value of the Boolean myBool is: " + myBool.toString());
```

## valueOf (méthode Boolean.valueOf)

```
public valueOf() : Boolean
```

Renvoie `true` si le type de valeur primitif de l'objet booléen spécifié est true ; `false` sinon.

**Disponibilité :** ActionScript 1.0 ; Flash Player 5

### Valeur renvoyée

`Boolean` - Valeur booléenne.

### Exemple

L'exemple suivant indique le mode de fonctionnement de cette méthode et montre également que le type de valeur primitif d'un nouvel objet booléen est `false` :

```
var x:Boolean = new Boolean();
trace(x.valueOf());   // false
x = (6==3+3);
trace(x.valueOf());   // true
```

# Button

```
Object
  |
  +-Button
```

```
public class Button
extends Object
```

Tous les symboles de bouton présents dans un fichier SWF sont des occurrences de l'objet `Button`. Vous pouvez donner un nom d'occurrence à un bouton dans l'inspecteur des propriétés, puis utiliser les méthodes et les propriétés de la classe Button pour manipuler les boutons avec ActionScript. Les noms d'occurrence de boutons s'affichent dans l'explorateur d'animations et dans la boîte de dialogue Insérer un chemin cible du panneau Actions.

La classe Button hérite de la classe Object.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

## Voir également

`Object`

Résumé des propriétés

| Modificateurs | Propriété | Description |
|---|---|---|
| | `_alpha:Number` | La valeur de transparence alpha du bouton spécifié par `my_btn`. |
| | `blendMode:Object` | Le mode de fondu de cet objet. |
| | `cacheAsBitmap:Boolean` | Si défini sur `true`, Flash Player place en mémoire cache une version bitmap interne du bouton. |
| | `enabled:Boolean` | Une valeur booléenne spécifiant si un bouton est activé. |
| | `filters:Array` | Un tableau indexé contenant tous les objets filtre associés au bouton. |
| | `_focusrect:Boolean` | Une valeur booléenne indiquant si un bouton est entouré d'un rectangle jaune lorsqu'il a le focus clavier. |
| | `_height:Number` | La hauteur du bouton, en pixels. |
| | `_highquality:Number` | *Déconseillé*depuis Flash Player 7. Cette propriété a été déconseillée en faveur de`Button._quality`. Spécifie le niveau d'anti-aliasing appliqué au fichier SWF actuel. |
| | `menu:ContextMenu` | Associe l'objet `ContextMenu`, `contextMenu` à l'objet `my_button`. |
| | `_name:String` | Nom d'occurrence du bouton spécifié par `my_btn`. |
| | `_parent:MovieClip` | Référence au clip ou à l'objet contenant le clip ou l'objet actuel. |
| | `_quality:String` | Propriété (globale) ; définit ou récupère la qualité de rendu utilisée pour un fichier SWF. |
| | `_rotation:Number` | La rotation du bouton, en degrés, à partir de son orientation d'origine. |
| | `scale9Grid:Rectangle` | La zone rectangulaire qui définit les neuf zones de redimensionnement du bouton. |
| | `_soundbuftime:Number` | Propriété qui spécifie le nombre de secondes pendant lequel les sons sont chargés en mémoire tampon avant d'être diffusés en continu. |
| | `tabEnabled:Boolean` | Spécifie si `my_btn` est inclus dans l'ordre de tabulation automatique. |
| | `tabIndex:Number` | Permet de personnaliser l'ordre de tabulation des objets dans un fichier SWF. |

| Modificateurs | Propriété | Description |
|---|---|---|
| | `_target:String` [lecture seule] | Renvoie le chemin cible de l'occurrence de bouton spécifiée par `my_btn`. |
| | `trackAsMenu:Boolean` | Valeur booléenne indiquant si d'autres boutons ou clips peuvent recevoir des événements de relâchement de souris. |
| | `_url:String` [lecture seule] | Récupère l'URL du fichier SWF qui a créé le bouton. |
| | `useHandCursor:Boolean` | Valeur booléenne qui, lorsqu'elle est définie sur `true` (par défaut), indique si un curseur en forme de main s'affiche lorsque la souris survole un bouton. |
| | `_visible:Boolean` | Une valeur booléenne indiquant si le bouton spécifié par `my_btn` est visible. |
| | `_width:Number` | La largeur du bouton, en pixels. |
| | `_x:Number` | Un entier qui définit la coordonnée x d'un bouton par rapport aux coordonnées locales du clip parent. |
| | `_xmouse:Number` [lecture seule] | Renvoie la coordonnée *x* de la position de la souris par rapport au bouton. |
| | `_xscale:Number` | Le redimensionnement horizontal du bouton tel qu'il est appliqué à partir du point d'alignement du bouton, exprimé en pourcentage. |
| | `_y:Number` | La coordonnée *y* du bouton par rapport aux coordonnées locales du clip parent. |
| | `_ymouse:Number` [lecture seule] | Indique la coordonnée *y* de la position de la souris par rapport au bouton. |
| | `_yscale:Number` | Le redimensionnement vertical du bouton tel qu'il est appliqué à partir du point d'alignement du bouton, exprimé en pourcentage. |

*Propriétés héritées de la classe Object*

```
constructeur (propriété Object.constructor), __proto__ (Object.__proto__
property), prototype (Object.prototype, propriété), __resolve
(Object.__resolve, propriété)
```

## Résumé des événements

| Evénement | Description |
|---|---|
| `onDragOut = function() {}` | Invoqué lorsque vous cliquez sur le bouton à l'aide du bouton de la souris, puis faites glisser le pointeur hors du bouton. |
| `onDragOver = function() {}` | Invoqué lorsque l'utilisateur appuie sur le bouton de la souris, le fait glisser hors du bouton, puis sur le bouton. |
| `onKeyDown = function() {}` | Invoqué lorsqu'un bouton reçoit le focus clavier et lorsque l'utilisateur appuie sur une touche. |
| `onKeyUp = function() {}` | Invoqué lorsqu'un bouton reçoit le focus d'entrée et lorsque l'utilisateur relâche une touche. |
| `onKillFocus = function(newFocus:Object) {}` | Invoqué lorsqu'un bouton perd le focus clavier. |
| `onPress = function() {}` | Invoqué lorsqu'un bouton est enfoncé. |
| `onRelease = function() {}` | Invoqué lorsqu'un bouton est relâché. |
| `onReleaseOutside = function() {}` | Invoqué lorsque l'utilisateur relâche la souris tandis que le pointeur se trouve hors du bouton après avoir appuyé sur celui-ci lorsque le pointeur se trouvait à l'intérieur du bouton. |
| `onRollOut = function() {}` | Invoqué lorsque le pointeur se déplace hors d'une zone du bouton. |
| `onRollOver = function() {}` | Invoqué lorsque le pointeur se déplace sur une zone du bouton. |
| `onSetFocus = function(oldFocus:Object) {}` | Invoqué lorsqu'un bouton reçoit le focus clavier. |

### Résumé de la méthode

| Modificateurs | Signature | Description |
|---|---|---|
| | `getDepth() : Number` | Renvoie la profondeur d'une occurrence de bouton. |

*Méthodes héritées de la classe Object*

`addProperty (méthode Object.addProperty)`, `hasOwnProperty (méthode Object.hasOwnProperty)`, `isPropertyEnumerable (méthode Object.isPropertyEnumerable)`, `isPrototypeOf (méthode Object.isPrototypeOf)`, `registerClass (méthode Object.registerClass)`, `toString (méthode Object.toString)`, `unwatch (méthode Object.unwatch)`, `valueOf (méthode Object.valueOf)`, `watch (méthode Object.watch)`

# _alpha (propriété Button._alpha)

```
public _alpha : Number
```

La valeur de transparence alpha du bouton spécifié par `my_btn`. Les valeurs possibles sont comprises entre 0 (entièrement transparent) et 100 (entièrement opaque). La valeur par défaut est 100. Les objets d'un bouton dont la propriété `_alpha` est définie sur 0 sont actifs, même s'ils sont invisibles.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

## Exemple

Le code suivant définit la propriété `_alpha` d'un bouton intitulé myBtn_btn sur 50 % lorsque l'utilisateur clique sur le bouton : D'abord, ajoutez une occurrence de Button sur la scène. Ensuite, donnez lui un nom d'occurrence de myBtn_btn. Pour terminer, l'image 1 étant sélectionnée, placez le code suivant dans le panneau Actions :

```
myBtn_btn.onRelease = function(){
  this._alpha = 50;
};
```

## Voir également

`_alpha (propriété MovieClip._alpha)`, `_alpha (propriété TextField._alpha)`

# blendMode (propriété Button.blendMode)

```
public blendMode : Object
```

Le mode de fondu de cet objet. Le mode fondu modifie l'apparence du bouton quand il est sur un calque au dessus d'un autre objet à l'écran.

Flash Player applique la propriété `blendMode` à chaque pixel du bouton. Chaque pixel est composé de trois couleurs élémentaires (rouge, vert et bleu), chacune de ces couleurs ayant une valeur située entre 0x00 et 0xFF. Flash Player compare chaque couleur élémentaire d'un pixel du bouton avec la couleur correspondante du pixel de l'arrière-plan. Par exemple, si `blendMode` est réglé sur `"lighten"`, Flash Player compare la valeur de rouge du bouton avec la valeur de rouge de l'arrière-plan et utilise la plus légère des deux comme valeur du composant rouge de la couleur affichée.

Le tableau suivant répertorie les réglages `blendMode`. Pour définir la propriété `blendMode`, vous pouvez utiliser un entier compris entre 1 et 14 ou une chaîne. Les illustrations du tableau montrent `blendMode` appliqué sur un bouton (2) quand il est superposé à un autre objet à l'écran (1).

| Valeur de l'entier | Valeur de chaîne | Illustration | Description |
|---|---|---|---|
| 1 | `"normal"` |  | Le bouton apparaît devant l'arrière-plan. Les valeurs de pixels du bouton écrasent celles de l'arrière-plan. Si le bouton est transparent, l'arrière-plan est visible. |
| 2 | `"layer"` |  | Force la création d'un tampon temporaire pour la précomposition du bouton. Ceci se fait automatiquement s'il existe plus d'un objet enfant dans un bouton et un réglage `blendMode` autre que `"normal"` est choisi pour l'enfant. |

| Valeur de l'entier | Valeur de chaîne | Illustration | Description |
| --- | --- | --- | --- |
| 3 | `"multiply"` |  | Multiplie les valeurs des couleurs primaires du bouton par celles de la couleur d'arrière-plan, et ensuite normalise en divisant par 0xFF, ce qui donne des couleurs plus foncées. Utilisé de façon commune pour les effets d'ombre et de profondeur.<br>Par exemple, si une couleur primaire (telle que le rouge) d'un pixel du bouton et que la couleur correspondante du pixel de l'arrière-plan ont toutes les deux la valeur 0x88, le résultat de la multiplication est 0x4840. La division par 0xFF donne une valeur de 0x48 pour cette couleur primaire, ce qui est une ombre plus foncée que celle du bouton ou de l'arrière-plan. |

| Valeur de l'entier | Valeur de chaîne | Illustration | Description |
|---|---|---|---|
| 4 | `"screen"` |  | Multiplie le complément (l'inverse) de la couleur du bouton par le complément de la couleur d'arrière-plan, ce qui donne un effet de blanchissement. Ce réglage est habituellement utilisé pour les surbrillances ou pour éliminer les zones noires du bouton. |
| 5 | `"lighten"` |  | Choisit la plus claire des couleurs primaires du bouton et celle de l'arrière-plan (celles qui ont les plus grandes valeurs). Ce réglage est habituellement utilisé pour les superpositions.<br>Par exemple, si le bouton a un pixel ayant une valeur RVB de 0xFFCC33, et le pixel d'arrière-plan une valeur RVB de 0xDDF800, la valeur RVB résultante pour le pixel affiché est de 0xFFF833 (parce que 0xFF > 0xDD, 0xCC < 0xF8, et 0x33 > 0x00 = 33). |

| Valeur de l'entier | Valeur de chaîne | Illustration | Description |
| --- | --- | --- | --- |
| 6 | "darken" |  | Choisit la plus foncée des couleurs primaires du bouton et celle de l'arrière-plan (celles qui ont les plus petites valeurs). Ce réglage est habituellement utilisé pour les superpositions.<br>Par exemple, si le bouton a un pixel ayant une valeur RVB de 0xFFCC33, et le pixel d'arrière-plan une valeur RVB de 0xDDF800, la valeur RVB résultante pour le pixel affiché est de 0xDDCC00 (parce que 0xFF › 0xDD, 0xCC ‹ 0xF8, et 0x33 › 0x00 = 33). |

| Valeur de l'entier | Valeur de chaîne | Illustration | Description |
|---|---|---|---|
| 7 | `"difference"` |  | Compare les couleurs primaires du bouton avec celles de son arrière-plan et soustrait la plus foncée des deux couleurs de la plus claire. Ce réglage est habituellement utilisé pour obtenir des couleurs plus vibrantes.<br>Par exemple, si le bouton a un pixel ayant une valeur RVB de 0xFFCC33, et le pixel d'arrière-plan une valeur RVB de 0xDDF800, la valeur RVB résultante pour le pixel affiché est de 0x222C33 (parce que 0xFF - 0xDD = 0x22, 0xF8 - 0xCC = 0x2C et 0x33 - 0x00 = 0x33). |

| Valeur de l'entier | Valeur de chaîne | Illustration | Description |
|---|---|---|---|
| 8 | "add" |  | Ajoute les couleurs primaires du bouton à celles de son arrière-plan, et applique un plafond de 0xFF. Ce réglage est habituellement utilisé pour animer un fondu d'éclaircissement entre deux objets.<br>Par exemple, si le bouton a un pixel ayant une valeur RVB de 0xAAA633, et le pixel d'arrière-plan une valeur RVB de 0xDD2200, la valeur RVB résultante pour le pixel affiché est de 0xFFC833 (parce que 0xAA + 0xDD › 0xFF, 0xA6 + 0x22 = 0xC8, et 0x33 + 0x00 = 0x33). |

| Valeur de l'entier | Valeur de chaîne | Illustration | Description |
|---|---|---|---|
| 9 | `"subtract"` |  | Soustrait les couleurs primaires du bouton de celles de son arrière-plan, et applique un plancher de 0. Ce réglage est habituellement utilisé pour animer un fondu d'assombrissement entre deux objets. Par exemple, si le bouton a un pixel ayant une valeur RVB de 0xAA2233, et le pixel d'arrière-plan une valeur RVB de 0xDDA600, la valeur RVB résultante pour le pixel affiché est de 0x338400 (parce que 0xDD - 0xAA = 0x33, 0xA6 - 0x22 = 0x84, et 0x00 - 0x33 < 0x00). |
| 10 | `"invert"` |  | Inverse l'arrière-plan. |
| 11 | `"alpha"` |  | Applique la valeur alpha de chaque pixel du bouton à l'arrière-plan. Ceci requiert que le "calque" `blendMode` soit appliqué à un bouton parent. Par exemple, sur l'illustration, le bouton parent, qui est un arrière-plan blanc, a la propriété `blendMode = "layer"`. |

| Valeur de l'entier | Valeur de chaîne | Illustration | Description |
|---|---|---|---|
| 12 | `"erase"` |  | Efface l'arrière plan sur la base de la valeur alpha du bouton. Ceci requiert que le réglage `"layer" blendMode` soit appliqué à un bouton parent. Par exemple, sur l'illustration, le bouton parent, qui est un arrière-plan blanc, a la propriété `blendMode = "layer"`. |

| Valeur de l'entier | Valeur de chaîne | Illustration | Description |
|---|---|---|---|
| 13 | `"overlay"` |  | Ajuste la couleur de chaque bitmap sur la base de l'obscurité de l'arrière-plan. Si l'arrière-plan est plus clair qu'un gris à 50 %, les couleurs du bouton et de l'arrière-plan sont masquées, ce qui donne une couleur plus claire. Si l'arrière-plan est plus foncé qu'un gris à 50 %, les couleurs sont multipliées, ce qui donne une couleur plus sombre. Ce réglage est habituellement utilisé pour les effets d'ombrage. |
| 14 | `"hardlight"` |  | Ajuste la couleur de chaque bitmap sur la base de l'obscurité du bouton. Si le bouton est plus clair qu'un gris à 50 %, les couleurs du bouton et de l'arrière-plan sont masquées, ce qui donne une couleur plus claire. Si l'arrière-plan est plus foncé qu'un gris à 50 %, les couleurs sont multipliées, ce qui donne une couleur plus sombre. Ce réglage est habituellement utilisé pour les effets d'ombrage. |

Si vous tentez de définir la propriété `blendMode`sur une autre valeur, Flash la définit sur `"normal"`.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

### Exemple

L'exemple suivant vous montrera que si vous définissez la propriété sur un entier, Flash convertit immédiatement cette valeur en une chaîne correspondante :

```
my_button.blendMode = 8;
trace (my_button.blendMode) // add
```

Pour un exemple similaire, veuillez consulter la description de la propriété `blendMode` de la classe MovieClip.

### Voir également

`blendMode (propriété MovieClip.blendMode)`

## cacheAsBitmap (propriété Button.cacheAsBitmap)

```
public cacheAsBitmap : Boolean
```

Si défini sur `true`, Flash Player place en mémoire cache une version bitmap interne du bouton. Cela permet d'améliorer les performances des boutons qui intègrent du contenu vectoriel complexe.

Pour un bouton ayant sa propriété `cacheAsBitmap` sur `true`, Flash Player stocke une représentation de bitmap pour chacun des quatre états du bouton.

Toutes les données vectorielles d'un bouton contenant une bitmap en mémoire cache sont tracées sur la bitmap et non pas sur la scène principale. Cette bitmap est ensuite copiée sur la scène principale sous forme de pixels sans étirement ou rotation et accrochés aux limites de pixels les plus proches. Les correspondances des pixels avec l'objet parent se font selon un rapport de 1 à 1. Si les limites de la bitmap changent, elle est recréée au lieu d'être étirée.

Aucune bitmap interne n'est créée sauf si la propriété `cacheAsBitmap` est définie sur `true`.

Après avoir défini la propriété `cacheAsBitmap` du bouton sur `true`, le rendu ne change pas, bien que le bouton procède automatiquement à l'accrochage aux pixels. La vitesse d'animation peut être beaucoup plus importante selon la complexité du contenu vectoriel.

La propriété `cacheAsBitmap` est définie automatiquement sur `true` lorsque vous appliquez un filtre à un bouton (lorsque son tableau `filter` n'est pas vide). Lorsqu'un bouton est filtré, `cacheAsBitmap` renvoie la valeur `true` pour ce bouton, même si vous l'avez définie sur `false`. Si vous supprimez tous les filtres d'un bouton, le réglage `cacheAsBitmap` à sa position précédente.

Dans les cas suivants, les boutons n'utilisent pas de bitmap, même si la propriété `cacheAsBitmap` est définie sur `true` et procède au rendu à partir de données vectorielles :

- Quand la bitmap est trop grande, c'est à dire supérieure à 2800 pixels dans l'un des deux sens.
- Quand la bitmap manque de mémoire allouée (en raison d'une erreur mémoire)

La propriété `cacheAsBitmap` est utilisée de préférence avec les boutons dont le contenu est principalement statique et qui n'est ni redimensionné, ni pivoté de façon fréquente. Avec ce genre de bouton, `cacheAsBitmap` peut entraîner une augmentation de performance quand le bouton est déplacé (quand ses positions *x* et *y* sont changées).

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

### Exemple

L'exemple suivant applique une ombre portée à une occurence de bouton existant sous le nom de `myButton`. Il trace ensuite la valeur de `cacheAsBitmap`, qui est réglée sur `true` lors de l'application d'un filtre.

```
import flash.filters.DropShadowFilter;
trace(myButton.cacheAsBitmap); // false
var dropShadow:DropShadowFilter = new DropShadowFilter(6, 45, 0x000000, 50,
  5, 5, 1, 2, false, false, false);
myButton.filters = new Array(dropShadow);
trace(myButton.cacheAsBitmap); // true
```

## enabled (propriété Button.enabled)

```
public enabled : Boolean
```

Une valeur booléenne spécifiant si un bouton est activé. Lorsqu'il est désactivé (la propriété enabled est définie sur `false`), le bouton est visible mais vous ne pouvez pas cliquer sur celui-ci. La valeur par défaut est `true`. Cette propriété s'avère utile si vous souhaitez désactiver certains des boutons de navigation ; par exemple, il peut être souhaitable de désactiver un bouton dans la page actuellement affichée afin d'empêcher tout clic sur celui-ci et d'empêcher de recharger la page.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

### Exemple

L'exemple suivant démontre comment vous pouvez désactiver et activer le clic de boutons. Deux boutons, `myBtn1_btn` et `myBtn2_btn`,, se trouvent sur la scène et le code ActionScript suivant est ajouté afin que l'utilisateur ne puisse pas cliquer sur le bouton `myBtn2_btn` : D'abord, ajoutez deux occurence de boutons sur la scène. Ensuite, attribuez leur les noms d'occurence `myBtn1_btn` et `myBtn2_btn`. Pour finir, placez le code suivant sur l'image 1 pour activer ou désactiver les boutons.

```
myBtn1_btn.enabled = true;
myBtn2_btn.enabled = false;

//button code
// the following function will not get called
// because myBtn2_btn.enabled was set to false
myBtn1_btn.onRelease = function() {
  trace( "you clicked : " + this._name );
};
myBtn2_btn.onRelease = function() {
  trace( "you clicked : " + this._name );
};
```

## filters (propriété Button.filters)

```
public filters : Array
```

Un tableau indexé contenant tous les objets filtre associés au bouton. Le package `flash.filters` contient plusieurs classes qui définissent des filtres spécifiques.

Ces filtres peuvent s'appliquer dans l'outil de programmation de Flash pendant la phase de conception ou d'exécution du code ActionScript. Pour appliquer un filtre avec ActionScript, vous devez créer une copie temporaire de l'intégralité du tableau `Button.filters`, modifier le tableau temporaire, puis reporter les valeurs de ce tableau temporaire dans le tableau `Button.filters`. Vous ne pouvez pas appliquer directement un nouvel objet filtre au tableau `Button.filters`. Le code suivant n'a aucun effet sur le bouton cible, appelé `myButton` :

```
myButton.filters[0].push(myDropShadow);
```

Pour ajouter un filtre avec ActionScript, vous devez suivre les étapes ci-dessous (dans cet exemple le bouton cible est appelé `myButton`) :

- Créez un objet filtre avec la fonction constructeur de la classe de filtre retenue.
- Assignez la valeur du tableau `myButton.filters` à un tableau temporaire, tel que celui qui est nommé `myFilters`.
- Ajoutez le nouvel objet filtre au tableau temporaire, `myFilters`.
- Affectez la valeur du tableau temporaire au tableau `myButton.filters`.

Si le tableau `filters` est vide, il n'est pas nécessaire d'utiliser un tableau temporaire. Par contre, vous pouvez affecter directement un littéral de tableau contenant un ou plusieurs des objets filter que vous avez créés.

Pour modifier un objet filtre existant, que ce dernier ait été créé pendant la phase de conception ou d'exécution, vous devez appliquer la technique de modification d'une copie du tableau `filters` :

- Assignez la valeur du tableau `myButton.filters` à un tableau temporaire, tel que celui qui est nommé `myFilters`.
- Modifiez la propriété avec le tableau temporaire, `myFilters`. Par exemple, si vous souhaitez définir la propriété `quality` du premier filtre du tableau, utilisez le code suivant : `myList[0].quality = 1;`
- Affectez la valeur du tableau temporaire au tableau `myButton.filters`.

Pour supprimer les filtres d'un bouton, définissez `filters` par un tableau vide (`[]`).

Lors du chargement, si un bouton est associé à un filtre, ce bouton se place en mémoire cache en tant que bitmap transparent. A partir de ce stade, tant que le bouton possède une liste de filtres valide, le lecteur place le clip en mémoire cache au format bitmap. Cette bitmap source est ensuite reprise en tant qu'image source pour les effets de filtrage. Habituellement, chaque bouton a deux jeux de bitmaps : Un avec le bouton d'origine non filtré et un autre pour les images finales après filtrage (dans chacun des quatre états du bouton). L'image finale est utilisée pour le rendu. Tant que le bouton ne change pas, l'image source ne nécessite aucune mise à jour.

Si vous manipulez un tableau `filters` contenant plusieurs filtres et devez suivre le type de filtre affecté à chaque index de tableau, vous pouvez conserver votre propre tableau `filters` et utiliser une structure de données distincte pour suivre le type de filtre associé à chaque index de tableau. Il n'existe aucune méthode simple permettant de déterminer le type de filtre associé à chaque index de tableau `filters`.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

## Exemple

L'exemple suivant ajoute un filtre d'ombre portée à un bouton appelé `myButton`.

```
import flash.filters.DropShadowFilter;
var myDropFilter:DropShadowFilter = new DropShadowFilter(6, 45, 0x000000,
  50, 5, 5, 1, 2, false, false, false);
var myFilters:Array = myButton.filters;
myFilters.push(myDropFilter);
myButton.filters = myFilters;
```

L'exemple suivant donne au paramètre `quality` du premier filtre du tableau la valeur 15 (cet exemple ne peut fonctionner que si au moins un objet filtre a été associé au champ texte `myButton`).

```
var myList:Array = myButton.filters;
myList[0].quality = 15;
myButton.filters = myList;
```

### Voir également

, `cacheAsBitmap (propriété Button.cacheAsBitmap)`

## _focusrect (propriété Button._focusrect)

```
public _focusrect : Boolean
```

Une valeur booléenne indiquant si un bouton est entouré d'un rectangle jaune lorsqu'il a le focus clavier. Cette propriété peut annuler la propriété `_focusrect` globale. Par défaut, la propriété `_focusrect` d'une occurrence de bouton est nulle, ce qui signifie que l'occurrence de bouton n'annule pas la propriété globale`_focusrect`. Si la propriété `_focusrect` d'une occurrence de bouton est définie sur `true` ou `false`, elle annule le paramètre de la propriété globale `_focusrect` de l'occurrence unique de bouton .

Dans les fichiers SWF de Flash Player 4 ou Flash Player 5, la propriété `_focusrect` contrôle la propriété globale `_focusrect`. Il s'agit d'une valeur booléenne. Ce comportement a été modifié dans Flash Player 6 et les versions ultérieures afin de pouvoir personnaliser la propriété `_focusrect` sur un clip individuel.

Si la propriété `_focusrect` est définie sur `false`, la navigation au clavier se limite à la touche Tab pour ce bouton. Toutes les autres touches, ce qui inclut la touche Entrée et les touches directionnelles, sont ignorées. Pour restaurer l'intégralité de l'accès clavier, vous devez définir `_focusrect` sur `true`.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

### Exemple

Cet exemple démontre comment masquer le rectangle jaune autour d'une occurrence de bouton spécifiée d'un fichier SWF lorsqu'elle a le focus dans une fenêtre de navigateur. Créez trois boutons intitulés `myBtn1_btn`, `myBtn2_btn` et `myBtn3_btn`, puis ajoutez le code ActionScript suivant à l'image 1 du scénario :

```
myBtn2_btn._focusrect = false;
```

Définissez les paramètres de publication sur Flash Player 6, puis testez le fichier SWF dans une fenêtre de navigateur en pointant sur Fichier > Aperçu avant publication > HTML. Attribuez le focus au fichier SWF en cliquant sur celui-ci dans la fenêtre de navigateur, puis utilisez la touche Tab pour appliquer le focus à chaque occurrence. Vous ne pourrez pas exécuter le code de ce bouton en appuyant sur la touche Entrée ou Espace lorsque la propriété `_focusrect` est désactivée.

# getDepth (méthode Button.getDepth)

```
public getDepth() : Number
```

Renvoie la profondeur d'une occurrence de bouton.

Tout clip, bouton et champ texte est associé à une profondeur unique qui détermine l'aspect de l'objet devant ou derrière d'autres objets. Les objets dont la profondeur est la plus importante s'affichent au premier plan.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

## Valeur renvoyée

`Number` - Profondeur d'une occurrence de bouton.

## Exemple

Si vous créez `myBtn1_btn` et `myBtn2_btn` sur la scène, vous pouvez suivre leur profondeur à l'aide du code ActionScript suivant :

```
trace(myBtn1_btn.getDepth());
trace(myBtn2_btn.getDepth());
```

Si vous chargez un fichier SWF intitulé buttonMovie.swf dans ce document, vous pouvez suivre la profondeur d'un bouton, `myBtn4_btn`, dans ce fichier SWF à l'aide d'un autre bouton du fichier SWF principal :

```
this.createEmptyMovieClip("myClip_mc", 999);
myClip_mc.loadMovie("buttonMovie.swf");
myBtn3_btn.onRelease = function(){
  trace(myClip_mc.myBtn4_btn.getDepth());
};
```

Vous remarquerez que deux de ces boutons ont la même valeur de profondeur, l'un dans le fichier SWF principal et l'autre dans le fichier SWF chargé. Cela peut vous induire en erreur car buttonMovie.swf a été chargé à la profondeur 999, ce qui signifie que le bouton qu'il contient aura également une profondeur de 999 par rapport aux boutons du fichier SWF principal. N'oubliez pas que chaque clip dispose de son propre ordre z interne, ce qui signifie que chaque clip possède son propre jeu de valeurs de profondeur. Les deux boutons peuvent avoir la même valeur de profondeur mais les valeurs ne sont significatives que par rapport aux autres objets du même ordre z. Dans ce cas, les boutons ont la même valeur de profondeur mais les valeurs se rapportent à des clips différents. la valeur de profondeur du bouton dans le fichier SWF principal se rapporte à l'ordre z du scénario principal, tandis que la valeur de profondeur du bouton du fichier SWF chargé se rapporte à l'ordre z interne du clip `myClip_mc`.

### Voir également

`getDepth (méthode MovieClip.getDepth)`, `getDepth (méthode TextField.getDepth)`,

## _height (propriété Button._height)

```
public _height : Number
```

La hauteur du bouton, en pixels.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

### Exemple

L'exemple suivant définit la hauteur et la largeur d'un bouton intitulé `my_btn` sur des valeurs spécifiées.

```
my_btn._width = 500;
my_btn._height = 200;
```

## _highquality (propriété Button._highquality)

```
public _highquality : Number
```

*Déconseillé*depuis Flash Player 7. Cette propriété a été déconseillée en faveur de`Button._quality`.

Spécifie le niveau d'anti-aliasing appliqué au fichier SWF actuel. Spécifiez 2 (meilleure qualité) pour bénéficier de la meilleure qualité possible et activer le lissage de façon permanente. Spécifiez 1 (haute qualité) pour procéder à l'anti-aliasing ; ceci permet de lisser les bitmaps si le fichier SWF ne contient pas d'animation et constitue la valeur par défaut. Spécifiez 0 (faible qualité) pour empêcher l'anti-aliasing.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

### Exemple

Ajoutez une occurrence de bouton sur la scène et appelez-la myBtn_btn. Tracez un ovale sur la scène à l'aide de l'outil Ovale ayant une couleur de trait et de remplissage. Sélectionnez l'image 1 et ajoutez le code ActionScript suivant via le panneau Actions :

```
myBtn_btn.onRelease = function() {
 myBtn_btn._highquality = 0;
};
```

Lorsque vous cliquez sur myBtn_btn, le trait du cercle est irrégulier. Vous pouvez ajouter le code ActionScript suivant pour affecter l'ensemble du fichier SWF :

```
_quality = 0;
```

### Voir également

`_quality (propriété Button._quality)`, `_quality, propriété`

## menu (propriété Button.menu)

```
public menu : ContextMenu
```

Associe l'objet `ContextMenu`, `contextMenu` à l'objet bouton `my_button`. La classe ContextMenu permet de modifier le menu contextuel qui s'affiche lorsque l'utilisateur clique avec le bouton droit de la souris (Windows) ou en appuyant sur la touche Contrôle (Macintosh) dans Flash Player.

**Disponibilité :** ActionScript 1.0 ; Flash Player 7

### Exemple

L'exemple suivant affecte un objet ContextMenu à une occurrence de bouton intitulée `myBtn_btn`. L'objet ContextMenu contient un élément de menu unique (intitulé « Save... ») incluant une fonction de gestionnaire de rappel intitulée `doSave`.

Ajoutez l'occurrence de bouton sur la scène et appelez-la myBtn_btn.

```
var menu_cm:ContextMenu = new ContextMenu();
menu_cm.customItems.push(new ContextMenuItem("Save...", doSave));
function doSave(menu:Object, obj:Object):Void {
  trace( " You selected the 'Save...' menu item ");
```

```
}
myBtn_btn.menu = menu_cm;
```

Pointez sur Contrôle > Tester l'animation pour tester le fichier SWF. Après avoir placé le pointeur sur `myBtn_btn`, cliquez avec le bouton droit de la souris ou maintenez la touche Contrôle enfoncée. Le menu contextuel incluant le bouton Enregistrer apparaît dans le menu. Lorsque vous cliquez sur Enregistrer dans le menu, le panneau de sortie s'affiche.

### Voir également

`ContextMenu`, `ContextMenuItem`, `menu (propriété MovieClip.menu)`, `menu (propriété TextField.menu)`

## _name (propriété Button._name)

```
public _name : String
```

Nom d'occurrence du bouton spécifié par `my_btn`.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

### Exemple

L'exemple suivant présente tous les noms d'occurrence des occurrences Button dans le scénario actuel d'un fichier SWF.

```
for (i in this) {
  if (this[i] instanceof Button) {
  trace(this[i]._name);
  }
}
```

## onDragOut (gestionnaire Button.onDragOut)

```
onDragOut = function() {}
```

Invoqué lorsque vous cliquez sur le bouton à l'aide du bouton de la souris, puis faites glisser le pointeur hors du bouton. Vous devez définir une fonction qui s'exécute lorsque le gestionnaire d'événements est appelé.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

### Exemple

L'exemple suivant démontre comment vous pouvez exécuter des instructions lorsque le pointeur ne se trouve plus sur un bouton. Créez un bouton intitulé `my_btn` sur la scène et entrez le code ActionScript suivant dans une image du scénario :

```
my_btn.onDragOut = function() {
```

```
  trace("onDragOut: "+this._name);
};
my_btn.onDragOver = function() {
  trace("onDragOver: "+this._name);
};
```

# onDragOver (gestionnaire Button.onDragOver)

## onDragOver = function() {}

Invoqué lorsque l'utilisateur appuie sur le bouton de la souris, le fait glisser hors du bouton, puis sur le bouton. Vous devez définir une fonction qui s'exécute lorsque le gestionnaire d'événements est appelé.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

### Exemple

L'exemple suivant définit une fonction pour le gestionnaire onDragOver qui envoie une instruction `trace()` au panneau de sortie. Créez un bouton intitulé `my_btn` sur la scène et entrez le code ActionScript suivant sur le scénario :

```
my_btn.onDragOut = function() {
  trace("onDragOut: "+this._name);
};
my_btn.onDragOver = function() {
  trace("onDragOver: "+this._name);
};
```

Lorsque vous testez le fichier SWF, éloignez le pointeur de l'occurrence de bouton en le faisant glisser. Ensuite, tout en maintenant le bouton de la souris enfoncé, faites-le glisser vers l'occurrence de bouton à nouveau. Vous pouvez constater que le panneau de sortie suit vos mouvements.

### Voir également

`onDragOut (gestionnaire Button.onDragOut)`

# onKeyDown (gestionnaire Button.onKeyDown)

## onKeyDown = function() {}

Invoqué lorsqu'un bouton reçoit le focus clavier et lorsque l'utilisateur appuie sur une touche. Le gestionnaire d'événements `onKeyDown` est appelé sans paramètre. Vous pouvez utiliser les méthodes `Key.getAscii()` et `Key.getCode()` afin d'identifier la touche sur laquelle l'utilisateur a appuyé. Vous devez définir une fonction qui s'exécute lorsque le gestionnaire d'événements est appelé.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

### Exemple

Dans l'exemple suivant, une fonction qui envoie du texte vers le panneau de sortie est définie pour le gestionnaire `onKeyDown`. Créez un bouton intitulé `my_btn` sur la scène et entrez le code ActionScript suivant dans une image du scénario :

```
my_btn.onKeyDown = function() {
  trace("onKeyDown: "+this._name+" (Key: "+getKeyPressed()+")");
};
function getKeyPressed():String {
  var theKey:String;
  switch (Key.getAscii()) {
  case Key.BACKSPACE :
    theKey = "BACKSPACE";
    break;
  case Key.SPACE :
    theKey = "SPACE";
    break;
  default :
    theKey = chr(Key.getAscii());
  }
  return theKey;
}
```

Pointez sur Contrôle > Tester l'animation pour tester le fichier SWF. Assurez-vous de sélectionner Contrôle > Désactivez les raccourcis clavier dans l'environnement de test. Ensuite, appuyez sur la touche Tab jusqu'à ce que le bouton ait le focus (un rectangle jaune entoure l'occurrence `my_btn`) et commencez à appuyer sur les touches de votre clavier. Lorsque vous appuyez sur les touches, elles s'affichent dans le panneau de sortie.

### Voir également

`onKeyUp (gestionnaire Button.onKeyUp)`, `getAscii (méthode Key.getAscii)`, `getCode (méthode Key.getCode)`

## onKeyUp (gestionnaire Button.onKeyUp)

### onKeyUp = function() {}

Invoqué lorsqu'un bouton reçoit le focus d'entrée et lorsque l'utilisateur relâche une touche. Le gestionnaire d'événements `onKeyUp` est appelé sans paramètre. Vous pouvez utiliser les méthodes `Key.getAscii()` et `Key.getCode()` afin d'identifier la touche sur laquelle l'utilisateur a appuyé.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

### Exemple

Dans l'exemple suivant, une fonction qui envoie du texte vers le panneau de sortie est définie pour le gestionnaire `onKeyDown handler`. Créez un bouton intitulé `my_btn` sur la scène et entrez le code ActionScript suivant dans une image du scénario :

```
my_btn.onKeyDown = function() {
  trace("onKeyDown: "+this._name+" (Key: "+getKeyPressed()+")");
};
my_btn.onKeyUp = function() {
  trace("onKeyUp: "+this._name+" (Key: "+getKeyPressed()+")");
};
function getKeyPressed():String {
  var theKey:String;
  switch (Key.getAscii()) {
  case Key.BACKSPACE :
    theKey = "BACKSPACE";
    break;
  case Key.SPACE :
    theKey = "SPACE";
    break;
  default :
    theKey = chr(Key.getAscii());
  }
  return theKey;
}
```

Appuyez sur Ctrl+Entrée pour tester le fichier SWF. Assurez-vous de sélectionner Contrôle > Désactivez les raccourcis clavier dans l'environnement de test. Ensuite, appuyez sur la touche Tab jusqu'à ce que le bouton ait le focus (un rectangle jaune entoure l'occurrence `my_btn`) et commencez à appuyer sur les touches de votre clavier. Lorsque vous appuyez sur les touches, elles s'affichent dans le panneau de sortie.

### Voir également

`onKeyDown (gestionnaire Button.onKeyDown)`, `getAscii (méthode Key.getAscii)`, `getCode (méthode Key.getCode)`

## onKillFocus (gestionnaire Button.onKillFocus)

### onKillFocus = function(newFocus:Object) {}

Invoqué lorsqu'un bouton perd le focus clavier. La gestionnaire `onKillFocus` reçoit un paramètre, `newFocus` : il s'agit d'un objet représentant le nouvel objet recevant le focus. Si aucun objet ne reçoit le focus, `newFocus` contient la valeur `null`.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

### Paramètres

*newFocus*:Object - Objet recevant le focus.

### Exemple

L'exemple suivant démontre comment exécuter des instructions lorsqu'un bouton perd le focus. Créez une occurrence de bouton intitulée my_btn sur la scène et ajoutez le code ActionScript suivant à l'image 1 du scénario :

```
this.createTextField("output_txt", this.getNextHighestDepth(), 0, 0, 300,
  200);
output_txt.wordWrap = true;
output_txt.multiline = true;
output_txt.border = true;
my_btn.onKillFocus = function() {
  output_txt.text = "onKillFocus: "+this._name+newline+output_txt.text;
};
```

Testez le fichier SWF dans une fenêtre de navigateur et essayez d'utiliser la touche Tab pour faire défiler les éléments dans la fenêtre. Lorsque l'occurrence de bouton perd le focus, le texte est envoyé vers le champ de texte output_txt.

La méthode MovieClip.getNextHighestDepth() utilisée dans cet exemple requiert Flash Player 7 ou version ultérieure. Si votre fichier SWF comprend un composant de version 2, utilisez la classe DepthManager avec composants version 2 plutôt que la méthode MovieClip.getNextHighestDepth()

## onPress (gestionnaire Button.onPress)

### onPress = function() {}

Invoqué lorsqu'un bouton est enfoncé. Vous devez définir une fonction qui s'exécute lorsque le gestionnaire d'événements est appelé.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

### Exemple

Dans l'exemple suivant, une fonction envoyant une instruction trace() vers le panneau de sortie est définie pour le gestionnaire onPress :

```
my_btn.onPress = function () {
  trace ("onPress called");
};
```

# onRelease (gestionnaire Button.onRelease)

onRelease = function() {}

Invoqué lorsqu'un bouton est relâché. Vous devez définir une fonction qui s'exécute lorsque le gestionnaire d'événements est appelé.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

## Exemple

Dans l'exemple suivant, une fonction envoyant une instruction trace() vers le panneau de sortie est définie pour le gestionnaire `onRelease` :

```
my_btn.onRelease = function () {
  trace ("onRelease called");
};
```

# onReleaseOutside (gestionnaire Button.onReleaseOutside)

onReleaseOutside = function() {}

Invoqué lorsque l'utilisateur relâche la souris tandis que le pointeur se trouve hors du bouton après avoir appuyé sur celui-ci lorsque le pointeur se trouvait à l'intérieur du bouton. Vous devez définir une fonction qui s'exécute lorsque le gestionnaire d'événements est appelé.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

## Exemple

Dans l'exemple suivant, une fonction envoyant une instruction trace() vers le panneau de sortie est définie pour le gestionnaire `onReleaseOutside` :

```
my_btn.onReleaseOutside = function () {
  trace ("onReleaseOutside called");
};
```

# onRollOut (gestionnaire Button.onRollOut)

onRollOut = function() {}

Invoqué lorsque le pointeur se déplace hors d'une zone du bouton. Vous devez définir une fonction qui s'exécute lorsque le gestionnaire d'événements est appelé.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

### Exemple

Dans l'exemple suivant, une fonction envoyant une instruction trace() vers le panneau de sortie est définie pour le gestionnaire `onRollOut` :

```
my_btn.onRollOut = function () {
  trace ("onRollOut called");
};
```

## onRollOver (gestionnaire Button.onRollOver)

### onRollOver = function() {}

Invoqué lorsque le pointeur se déplace sur une zone du bouton. Vous devez définir une fonction qui s'exécute lorsque le gestionnaire d'événements est appelé.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

### Exemple

Dans l'exemple suivant, une fonction envoyant une instruction trace() vers le panneau de sortie est définie pour le gestionnaire `onRollOver` :

```
my_btn.onRollOver = function () {
  trace ("onRollOver called");
};
```

## onSetFocus (gestionnaire Button.onSetFocus)

### onSetFocus = function(oldFocus:Object) {}

Invoqué lorsqu'un bouton reçoit le focus clavier. Le paramètre `oldFocus` est l'objet qui perd le focus. Par exemple, si l'utilisateur appuie sur la touche Tab pour déplacer le focus d'entrée d'un champ de texte vers un bouton, le paramètre oldFocus contient l'occurrence de champ de texte.

Si aucun objet n'avait précédemment reçu le focus, le paramètre `oldFocus` contient une valeur null.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

### Paramètres

*`oldFocus`*`:Object` - Objet perdant le focus du clavier.

### Exemple

L'exemple suivant démontre comment vous pouvez exécuter des instructions lorsque l'utilisateur d'un fichier SWF déplace le focus d'un bouton vers un autre. Créez deux boutons, `btn1_btn` et `btn2_btn`, puis entrez le code ActionScript suivant dans l'image 1 du scénario :

```
Selection.setFocus(btn1_btn);
trace(Selection.getFocus());
btn2_btn.onSetFocus = function(oldFocus) {
  trace(oldFocus._name + "lost focus");
};
```

Testez le fichier SWF en appuyant sur Ctrl+Entrée. Assurez-vous de sélectionner Contrôle > Désactivez les raccourcis clavier si vous ne l'avez pas déjà fait. Le focus est défini sur `btn1_btn`. Lorsque `btn1_btn` perd le focus au détriment de `btn2_btn`, les informations s'affichent dans le panneau de sortie.

## _parent (propriété Button._parent)

```
public _parent : MovieClip
```

Référence au clip ou à l'objet contenant le clip ou l'objet actuel. L'objet actuel est l'objet qui contient le code ActionScript faisant référence à `_parent`.

Utilisez `_parent` pour spécifier un chemin relatif vers les clips ou les objets situés au-dessus du clip ou de l'objet actuel. Vous pouvez utiliser `_parent` pour remonter de plusieurs niveaux dans l'arborescence de la liste d'affichage, comme dans l'exemple suivant :

```
this._parent._parent._alpha = 20;
```

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

### Exemple

Dans l'exemple suivant, un bouton intitulé `my_btn` est placé dans un clip intitulé `my_mc`. Le code suivant montre comment utiliser la propriété `_parent` pour obtenir une référence au clip `my_mc` :

```
trace(my_mc.my_btn._parent);
```

Le panneau de sortie affiche le code suivant :

```
_level0.my_mc
```

### Voir également

`_parent (propriété MovieClip._parent)`, `_target (propriété MovieClip._target)`, `_root, propriété`

# _quality (propriété Button._quality)

```
public _quality : String
```

Propriété (globale) ; définit ou récupère la qualité de rendu utilisée pour un fichier SWF. Les polices de périphérique sont toujours aliasées, ce qui implique qu'elles ne sont pas affectées par la propriété `_quality`.

La propriété `_quality` peut être définie sur les valeurs suivantes :

- "`LOW` Qualité de rendu inférieure. Les images ne sont pas anti-aliasées et les bitmaps ne sont pas lissées.
- "`MEDIUM` Qualité de rendu moyenne. Les images sont anti-aliasées selon une grille de 2 x 2 pixels, mais les bitmaps ne sont pas lissées. Ce niveau de qualité convient aux animations qui ne contiennent pas de texte.
- "`HIGH` Qualité de rendu supérieure. Les images sont anti-aliasées en appliquant une grille de 4 x 4 pixels et les bitmaps sont lissées lorsque l'animation est statique. Il s'agit du paramètre de qualité de rendu par défaut de Flash.
- "`BEST` Très haute qualité de rendu. Les graphiques sont anti-aliasés selon une grille de 4 x 4 pixels et les bitmaps sont toujours lissés.

*Remarque :* Bien que vous puissiez spécifier cette propriété pour un objet Button, il s'agit en fait d'une propriété globale : il vous suffit donc de définir sa valeur sur `_quality`.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

## Exemple

Cet exemple définit la qualité de rendu d'un bouton intitulé `my_btn` sur `LOW` :

```
my_btn._quality = "LOW";
```

# _rotation (propriété Button._rotation)

```
public _rotation : Number
```

La rotation du bouton, en degrés, à partir de son orientation d'origine. Les valeurs comprises entre 0 et 180 représentent la rotation en sens horaire ; les valeurs comprises entre 0 et -180 représentent la rotation en sens anti-horaire. Les valeurs hors de cette plage sont ajoutées ou soustraites de 360 pour obtenir une valeur comprise dans la plage. Par exemple, l'instruction `my_btn._rotation = 450` est identique à `my_btn._rotation = 90`.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

### Exemple

L'exemple suivant fait pivoter deux boutons sur la scène. Créez deux boutons intitulés `control_btn` et `my_btn` sur la scène. Assurez-vous que `my_btn` n'est pas parfaitement arrondi afin que vous puissiez le voir pivoter. Entrez ensuite le code ActionScript suivant dans l'image 1 du scénario :

```
var control_btn:Button;
var my_btn:Button;
control_btn.onRelease = function() {
  my_btn._rotation += 10;
};
```

Créez maintenant un autre bouton intitulé `myOther_btn` sur la scène, en veillant à ce qu'il ne soit pas parfaitement arrondi (afin que vous puissiez le voir pivoter). Entrez le code ActionScript suivant dans l'image 1 du scénario.

```
var myOther_btn:Button;
this.createEmptyMovieClip("rotater_mc", this.getNextHighestDepth());
rotater_mc.onEnterFrame = function() {
  myOther_btn._rotation += 2;
};
```

La méthode `MovieClip.getNextHighestDepth()` utilisée dans cet exemple requiert Flash Player 7 ou version ultérieure. Si votre fichier SWF comprend un composant de version 2, utilisez la classe DepthManager avec composants version 2 plutôt que la méthode `MovieClip.getNextHighestDepth()`

### Voir également

`_rotation (propriété MovieClip._rotation)`, `_rotation (propriété TextField._rotation)`

## scale9Grid (propriété Button.scale9Grid)

```
public scale9Grid : Rectangle
```

La zone rectangulaire qui définit les neuf zones de redimensionnement du bouton. Si la valeur est définie sur `null`, le bouton tout entier est alors dimensionné normalement lorsqu'une transformation par redimensionnement est appliquée.

Lorsque vous définissez une propriété `scale9Grid` pour un bouton, le bouton est divisé dans une grille comportant neuf zones, en fonction du rectangle `scale9Grid`, qui définit le centre de la grille. La grille est constituée des huit autres zones suivantes :

- La zone située dans le coin supérieur gauche, en dehors du rectangle.
- La zone située au-dessus du rectangle
- La zone située dans le coin supérieur droit, en dehors du rectangle.

- La zone située à gauche du rectangle
- La zone située à droite du rectangle
- La zone située dans le coin inférieur gauche, en dehors du rectangle.
- La zone située en dessous du rectangle
- La zone située dans le coin inférieur droit, en dehors du rectangle.

Les huit zones entourant la partie centrale (définie par le rectangle) peuvent être conçues comme un cadre qui bénéficie de règles spécifiques de redimensionnement.

Lorsque la propriété `scale9Grid` est définie et qu'un bouton est redimensionné, la totalité du texte et les dégradés sont dimensionnés normalement ; toutefois, les règles suivantes s'appliquent pour les autres types d'objets :

- Le contenu de la zone centrale est redimensionné normalement.
- Le contenu apparaissant dans les coins n'est pas redimensionné.
- Le contenu apparaissant dans les zones supérieures et inférieures est redimensionné horizontalement uniquement. Le contenu apparaissant dans les zones de gauche et de droite est redimensionné verticalement uniquement.

Si vous faites pivoter un bouton, tout redimensionnement effectué ultérieurement est normal et la propriété `scale9Grid` est ignorée.

L'un des modes d'utilisation de la propriété `scale9Grid` le plus répandu consiste à configurer un bouton dont les contours conservent la même largeur lorsque celui-ci est redimensionné.

Vous obtiendrez davantage d'informations, y compris des illustrations et un exemple s'y rapportant en consultant MovieClip.scale9Grid.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

### Voir également

`Rectangle (flash.geom.Rectangle)`, `scale9Grid (propriété MovieClip.scale9Grid)`

## _soundbuftime (propriété Button._soundbuftime)

```
public _soundbuftime : Number
```

Propriété qui spécifie le nombre de secondes pendant lequel les sons sont chargés en mémoire tampon avant d'être diffusés en continu.

*Remarque :* Bien que vous puissiez spécifier cette propriété pour un objet Button, il s'agit en fait d'une propriété globale qui s'applique à tous les sons chargés : il vous suffit donc de définir sa valeur sur `_soundbuftime`. La définition de cette propriété pour un objet Button permet de définir la propriété globale.

Pour plus d'informations et un exemple, consultez la propriété globale `_soundbuftime`.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

### Voir également

`_soundbuftime, propriété`

## tabEnabled (propriété Button.tabEnabled)

```
public tabEnabled : Boolean
```

Spécifie si `my_btn` est inclus dans l'ordre de tabulation automatique. La valeur par défaut est `undefined`.

Si la propriété `tabEnabled` est définie sur `undefined` ou `true`, l'objet est inclus dans l'ordre de tabulation automatique. Si la propriété `tabIndex` est également définie sur une valeur, l'objet est également inclus dans l'ordre de tabulation personnalisé. Si la propriété `tabEnabled` est définie sur `false`, l'objet n'est pas inclus dans l'ordre de tabulation automatique ou personnalisé, même si la propriété `tabIndex` est définie.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

### Exemple

Le code ActionScript suivant est utilisé pour définir la propriété `tabEnabled` sur `false` pour l'un des quatre boutons. Cependant, les quatre boutons (`one_btn`, `two_btn`, `three_btn`, et `four_btn`) sont placés dans un ordre de tabulation personnalisé à l'aide de `tabIndex`. Bien que la propriété `tabIndex` soit définie pour le bouton `three_btn`, ce dernier n'est pas inclus dans un ordre de tabulation personnalisé ou automatique car la propriété `tabEnabled` est définie sur `false` pour cette occurrence. Pour définir l'ordre de tabulation des quatre boutons, ajoutez le code ActionScript suivant à l'image 1 du scénario :

```
three_btn.tabEnabled = false;
two_btn.tabIndex = 1;
four_btn.tabIndex = 2;
three_btn.tabIndex = 3;
one_btn.tabIndex = 4;
```

Veillez à désactiver les raccourcis clavier lorsque vous testez le fichier SWF. Pour ce faire, il vous suffit de pointer sur Contrôle > Désactiver les raccourcis clavier dans l'environnement de test.

### Voir également

`tabIndex (propriété Button.tabIndex), tabEnabled (propriété MovieClip.tabEnabled), tabEnabled (propriété TextField.tabEnabled)`

# tabIndex (propriété Button.tabIndex)

```
public tabIndex : Number
```

Permet de personnaliser l'ordre de tabulation des objets dans un fichier SWF. Vous pouvez définir la propriété `tabIndex` sur un bouton, un clip ou une occurrence de champ texte ; sa valeur par défaut est `undefined`.

Si un objet actuellement affiché dans le fichier SWF contient une propriété `tabIndex`, l'ordre de tabulation automatique est désactivé : l'ordre de tabulation est alors calculé à partir des propriétés `tabIndex` des objets contenus dans le fichier SWF. L'ordre de tabulation personnalisé inclut uniquement des objets dotés de propriétés `tabIndex`.

La propriété `tabIndex` peut être un entier non négatif. Les objets sont triés selon leurs propriétés `tabIndex`, par ordre croissant. Un objet possédant une valeur `tabIndex` de `1` précède un objet ayant une valeur `tabIndex` de `2`. Si deux objets ont la même valeur `tabIndex`, celui qui précède l'autre dans l'ordre de tabulation est `undefined` (non défini).

L'ordre de tabulation personnalisé défini par la propriété `tabIndex` est *plat*. Cela signifie qu'on ne prête aucune attention aux relations hiérarchiques des objets contenus dans le fichier SWF. Tous les objets du fichier SWF dotés de propriétés `tabIndex` sont placés dans l'ordre de tabulation, qui est déterminé par l'ordre des valeurs `tabIndex`. Si deux objets ont la même valeur `tabIndex`, celui qui apparaît en premier est `undefined`(non défini). Il est recommandé de ne pas affecter la même valeur `tabIndex` à plusieurs objets.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

## Exemple

Le code ActionScript suivant est utilisé pour définir la propriété `tabEnabled` sur `false` pour l'un des quatre boutons. Cependant, les quatre boutons (`one_btn`, `two_btn`, `three_btn`, et `four_btn`) sont placés dans un ordre de tabulation personnalisé à l'aide de `tabIndex`. Bien que la propriété `tabIndex` soit définie pour le bouton `three_btn`, ce dernier n'est pas inclus dans un ordre de tabulation personnalisé ou automatique car la propriété `tabEnabled` est définie sur `false` pour cette occurrence. Pour définir l'ordre de tabulation des quatre boutons, ajoutez le code ActionScript suivant à l'image 1 du scénario :

```
three_btn.tabEnabled = false;
two_btn.tabIndex = 1;
four_btn.tabIndex = 2;
three_btn.tabIndex = 3;
one_btn.tabIndex = 4;
```

Veillez à désactiver les raccourcis clavier lorsque vous testez le fichier SWF. Pour ce faire, il vous suffit de pointer sur Contrôle > Désactiver les raccourcis clavier dans l'environnement de test.

### Voir également

tabEnabled (propriété Button.tabEnabled), tabChildren (propriété MovieClip.tabChildren), tabEnabled (propriété MovieClip.tabEnabled), tabIndex (propriété MovieClip.tabIndex), tabIndex (propriété TextField.tabIndex)

## _target (propriété Button._target)

```
public _target : String [lecture seule]
```

Renvoie le chemin cible de l'occurrence de bouton spécifiée par `my_btn`.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

### Exemple

Ajoutez une occurrence de bouton intitulée `my_btn` sur la scène, puis ajoutez le code suivant à l'image 1 du scénario :

```
trace(my_btn._target); //displays /my_btn
```

Sélectionnez my_btn et convertissez-le en clip. Attribuez au nouveau clip un nom d'occurrence `my_mc`. Supprimez le code ActionScript existant dans l'image 1 du scénario et remplacez-le par le code suivant :

```
my_mc.my_btn.onRelease = function(){
  trace(this._target); //displays /my_mc/my_btn
};
```

Pour convertir la notation avec barre oblique en notation avec point, modifiez l'exemple de code précédent comme suit :

```
my_mc.my_btn.onRelease = function(){
  trace(eval(this._target)); //displays _level0.my_mc.my_btn
};
```

Ceci vous permet d'accéder aux méthodes et paramètres de l'objet cible, tels que :

```
my_mc.my_btn.onRelease = function(){
  var target_btn:Button = eval(this._target);
trace(target_btn._name); //displays my_btn
};
```

### Voir également

_target (propriété MovieClip._target)

# trackAsMenu (propriété Button.trackAsMenu)

```
public trackAsMenu : Boolean
```

Valeur booléenne indiquant si d'autres boutons ou clips peuvent recevoir des événements de relâchement de souris. Si vous faites glisser un bouton, puis relâchez un deuxième bouton, l'événement `onRelease` est enregistré pour le deuxième bouton. Cette opération vous permet de créer des menus. Vous pouvez définir la propriété `trackAsMenu` sur n'importe quel bouton ou objet de clip. Si la propriété `trackAsMenu` n'a pas été définie, la valeur du comportement par défaut est `false`.

Vous pouvez modifier la propriété `trackAsMenu` à tout moment ; le bouton modifié accepte immédiatement le nouveau comportement.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

## Exemple

L'exemple suivant démontre comment identifier deux boutons en tant que menu. Placez deux occurrences de bouton intitulées `one_btn` et `two_btn` sur la scène. Entrez le code ActionScript suivant dans le scénario :

```
var one_btn:Button;
var two_btn:Button;
one_btn.trackAsMenu = true;
two_btn.trackAsMenu = true
one_btn.onRelease = function() {
  trace("clicked one_btn");
};
two_btn.onRelease = function() {
  trace("clicked two_btn");
};
```

Testez le fichier SWF. Pour ce faire, il vous suffit de cliquer sur `one_btn` dans la scène, en maintenant le bouton de la souris enfoncé, puis de le relâcher sur `two_btn`. Essayez ensuite de commenter les deux lignes du code ActionScript contenant `trackAsMenu` et testez à nouveau le fichier SWF pour voir la différence de comportement du bouton.

## Voir également

`trackAsMenu (propriété MovieClip.trackAsMenu)`

# _url (propriété Button._url)

```
public _url : String [lecture seule]
```

Récupère l'URL du fichier SWF qui a créé le bouton.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

### Exemple

Créez deux occurrences de bouton intitulées `one_btn` et `two_btn` sur la scène. Entrez le code ActionScript suivant dans l'image 1 du scénario :

```
var one_btn:Button;
var two_btn:Button;
this.createTextField("output_txt", 999, 0, 0, 100, 22);
output_txt.autoSize = true;
one_btn.onRelease = function() {
  trace("clicked one_btn");
  trace(this._url);
};
two_btn.onRelease = function() {
  trace("clicked "+this._name);
  var url_array:Array = this._url.split("/");
  var my_str:String = String(url_array.pop());
  output_txt.text = unescape(my_str);
};
```

Lorsque vous cliquez sur chaque bouton, le nom du fichier SWF contenant les boutons s'affiche dans le panneau de sortie.

## useHandCursor (propriété Button.useHandCursor)

```
public useHandCursor : Boolean
```

Valeur booléenne qui, lorsqu'elle est définie sur `true` (par défaut), indique si un curseur en forme de main s'affiche lorsque la souris survole un bouton. Si cette propriété est définie sur `false`, le pointeur flèche est utilisé.

Vous pouvez modifier la propriété `useHandCursor` à tout moment ; le bouton modifié accepte immédiatement le nouveau comportement du curseur. La propriété `useHandCursor` peut être extraite d'un objet prototype.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

### Exemple

Créez deux boutons portant les noms d'occurrence `myBtn1_btn` et `myBtn2_btn` sur la scène. Entrez le code ActionScript suivant dans l'image 1 du scénario :

```
myBtn1_btn.useHandCursor = false;
myBtn1_btn.onRelease = buttonClick;
myBtn2_btn.onRelease = buttonClick;
function buttonClick() {
  trace(this._name);
}
```

Lorsque le curseur de la souris survole `myBtn1_btn` et clique sur celui-ci, aucun curseur en forme de main n'apparaît. En revanche, le curseur en forme de main apparaît lorsque le bouton survole `myBtn2_btn` et clique sur celui-ci.

# _visible (propriété Button._visible)

```
public _visible : Boolean
```

Une valeur booléenne indiquant si le bouton spécifié par `my_btn` est visible. Les boutons qui ne sont pas visibles (propriété `_visible` définie sur `false`) sont désactivés.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

## Exemple

Créez deux boutons portant les noms d'occurrence `myBtn1_btn` et `myBtn2_btn` sur la scène. Entrez le code ActionScript suivant dans l'image 1 du scénario :

```
myBtn1_btn.onRelease = function() {
  this._visible = false;
  trace("clicked "+this._name);
};
myBtn2_btn.onRelease = function() {
  this._alpha = 0;
  trace("clicked "+this._name);
};
```

Vous remarquez que vous pouvez toujours cliquer sur `myBtn2_btn` lorsque la valeur alpha est définie sur 0.

## Voir également

`_visible (propriété MovieClip._visible)`, `_visible (propriété TextField._visible)`

# _width (propriété Button._width)

```
public _width : Number
```

La largeur du bouton, en pixels.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

### Exemple

L'exemple suivant augmente la valeur de la propriété width d'un bouton intitulé `my_btn` et affiche la largeur dans le panneau de sortie. Entrez le code ActionScript suivant dans l'image 1 du scénario :

```
my_btn.onRelease = function() {
  trace(this._width);
  this._width ~= 1.1;
};
```

### Voir également

`_width (propriété MovieClip._width)`

## _x (propriété Button._x)

```
public _x : Number
```

Un entier qui définit la coordonnée x d'un bouton par rapport aux coordonnées locales du clip parent. Si un bouton se trouve sur le scénario principal, son système de coordonnées se réfère alors au coin supérieur gauche de la scène : (0, 0). Si le bouton est imbriqué dans un clip subissant des transformations, le bouton se trouve dans le système de coordonnées local du clip qui l'encadre. Ainsi, dans le cas d'un clip qui a effectué une rotation à 90 degrés en sens anti-horaire, le bouton imbriqué hérite d'un système de coordonnées ayant effectué une rotation à 90 degrés en sens anti-horaire. Les coordonnées du bouton renvoient à la position du point d'alignement.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

### Exemple

L'exemple suivant définit les coordonnées de `my_btn` sur 0 sur la scène. Créez un bouton intitulé `my_btn` et entrez le code ActionScript suivant dans l'image 1 du scénario :

```
my_btn._x = 0;
my_btn._y = 0;
```

### Voir également

`_xscale (propriété Button._xscale)`, `_y (propriété Button._y)`, `_yscale (propriété Button._yscale)`

# _xmouse (propriété Button._xmouse)

```
public _xmouse : Number [lecture seule]
```

Renvoie la coordonnée *x* de la position de la souris par rapport au bouton.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

## Exemple

L'exemple suivant affiche la position xmouse pour la scène et un bouton intitulé `my_btn` placé sur celle-ci. Entrez le code ActionScript suivant dans l'image 1 du scénario :

```
this.createTextField("mouse_txt", 999, 5, 5, 150, 40);
mouse_txt.html = true;
mouse_txt.wordWrap = true;
mouse_txt.border = true;
mouse_txt.autoSize = true;
mouse_txt.selectable = false;
//
var mouseListener:Object = new Object();
mouseListener.onMouseMove = function() {
  var table_str:String = "<textformat tabstops='[50,100]'>";
  table_str += "<b>Stage</b>\t"+"x:"+_xmouse+"\t"+"y:"+_ymouse+newline;
  table_str += "<b>Button</
  b>\t"+"x:"+my_btn._xmouse+"\t"+"y:"+my_btn._ymouse+newline;
  table_str += "</textformat>";
  mouse_txt.htmlText = table_str;
};
Mouse.addListener(mouseListener);
```

## Voir également

_ymouse (propriété Button._ymouse)

# _xscale (propriété Button._xscale)

```
public _xscale : Number
```

Le redimensionnement horizontal du bouton tel qu'il est appliqué à partir du point d'alignement du bouton, exprimé en pourcentage. Le point d'alignement par défaut est (0,0).

Le redimensionnement du système de coordonnées local affecte les paramètres des propriétés `_x` et `_y`, définis en pixels. Par exemple, si le clip parent est redimensionné à 50 %, le paramètrage de la propriété `_x` déplace un objet sur le bouton selon un nombre de pixels réduit de moitié par rapport à celui qui serait appliqué si le fichier SWF était défini sur 100 %.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

### Exemple

L'exemple suivant redemandions un bouton intitulé m_btn. Lorsque vous cliquez sur le bouton et le relâchez, sa taille augmente de 10 % sur l'axe *x* et *y*. Entrez le code ActionScript suivant dans l'image 1 du scénario :

```
my_btn.onRelease = function(){
  this._xscale ~= 1.1;
  this._yscale ~= 1.1;
};
```

### Voir également

`_x (propriété Button._x)`, `_y (propriété Button._y)`, `_yscale (propriété Button._yscale)`

## _y (propriété Button._y)

```
public _y : Number
```

La coordonnée *y* du bouton par rapport aux coordonnées locales du clip parent. Si un bouton se trouve dans le scénario principal, son système de coordonnées se réfère au coin supérieur gauche de la scène : (0, 0). Si le bouton est imbriqué dans un autre clip subissant des transformations, le bouton se trouve dans le système de coordonnées local du clip qui l'encadre. Ainsi, dans le cas d'un clip qui a effectué une rotation à 90 degrés en sens anti-horaire, le bouton imbriqué hérite d'un système de coordonnées ayant effectué une rotation à 90 degrés en sens anti-horaire. Les coordonnées du bouton renvoient à la position du point d'alignement.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

### Exemple

L'exemple suivant définit les coordonnées de `my_btn` sur 0 sur la scène. Créez un bouton intitulé `my_btn` et entrez le code ActionScript suivant dans l'image 1 du scénario :

```
my_btn._x = 0;
my_btn._y = 0;
```

### Voir également

`_x (propriété Button._x)`, `_xscale (propriété Button._xscale)`, `_yscale (propriété Button._yscale)`

# _ymouse (propriété Button._ymouse)

```
public _ymouse : Number [lecture seule]
```

Indique la coordonnée *y* de la position de la souris par rapport au bouton.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

## Exemple

L'exemple suivant affiche la position ymouse pour la scène et un bouton intitulé `my_btn` placé sur celle-ci. Entrez le code ActionScript suivant dans l'image 1 du scénario :

```
this.createTextField("mouse_txt", 999, 5, 5, 150, 40);
mouse_txt.html = true;
mouse_txt.wordWrap = true;
mouse_txt.border = true;
mouse_txt.autoSize = true;
mouse_txt.selectable = false;
//
var mouseListener:Object = new Object();
mouseListener.onMouseMove = function() {
  var table_str:String = "<textformat tabstops='[50,100]'>";
  table_str += "<b>Stage</b>\t"+"x:"+_xmouse+"\t"+"y:"+_ymouse+newline;
  table_str += "<b>Button</
  b>\t"+"x:"+my_btn._xmouse+"\t"+"y:"+my_btn._ymouse+newline;
  table_str += "</textformat>";
  mouse_txt.htmlText = table_str;
};
Mouse.addListener(mouseListener);
```

## Voir également

`_xmouse (propriété Button._xmouse)`

# _yscale (propriété Button._yscale)

```
public _yscale : Number
```

Le redimensionnement vertical du bouton tel qu'il est appliqué à partir du point d'alignement du bouton, exprimé en pourcentage. Le point d'alignement par défaut est (0,0).

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

## Exemple

L'exemple suivant redimensionne un bouton intitulé `my_btn`. Lorsque vous cliquez sur le bouton et le relâchez, sa taille augmente de 10 % sur l'axe *x* et *y*. Entrez le code ActionScript suivant dans l'image 1 du scénario :

```
my_btn.onRelease = function(){
```

```
  this._xscale ~= 1.1;
  this._yscale ~ 1.1;
};
```

### Voir également

`_y (propriété Button._y)`, `_x (propriété Button._x)`, `_xscale (propriété Button._xscale)`

# Camera

```
Object
  |
  +-Camera
```

`public class` ***Camera***
`extends Object`

La classe Camera est principalement dédiée à Macromedia Flash Communication Server, mais peut être utilisée de façon restreinte sans le serveur.

La classe Camera vous permet de capturer de la vidéo à partir d'une caméra vidéo reliée à l'ordinateur qui exécute Macromedia Flash Player, par exemple, pour surveiller une vidéo à partir d'une caméra Web reliée à votre système local. (Flash est doté de fonctionnalités audio similaires ; pour plus d'informations, consultez l'entrée de la classe Microphone.)

*Avertissement :* Lorsqu'un fichier SWF tente d'accéder à la caméra renvoyée par `Camera.get()`, Flash Player affiche une boîte de dialogue Confidentialité permettant à l'utilisateur d'autoriser ou de refuser l'accès à la caméra. (Assurez-vous que la taille de votre scène est d'au moins 215 x 138 pixels pour les exemples de la classe Camera ; il s'agit de la taille minimale requise par Flash pour afficher la boîte de dialogue.) Les utilisateurs finaux et administratifs peuvent également désactiver l'accès à la caméra emplacement par emplacement ou de manière globale.

Pour créer ou référencer un objet Camera, utilisez la méthode `Camera.get()`.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

## Résumé des propriétés

| Modificateurs | Propriété | Description |
|---|---|---|
| | `activityLevel:Number` [lecture seule] | Une valeur numérique spécifiant la quantité de mouvement détectée par la caméra. |
| | `bandwidth:Number` [lecture seule] | Un entier spécifiant la quantité maximale de bande passante pouvant être utilisée par la vidéo sortante actuelle, en octets. |
| | `currentFps:Number` [lecture seule] | Cadence à laquelle la caméra capture des données, en images par seconde. |
| | `fps:Number` [lecture seule] | Cadence maximale à laquelle vous voulez que la caméra capture des données, en images par seconde. |
| | `height:Number` [lecture seule] | La hauteur de capture actuelle, en pixels. |
| | `index:Number` [lecture seule] | Entier de base zéro spécifiant l'indice de la caméra, tel qu'indiqué dans le tableau renvoyé par `Camera.names`. |
| | `motionLevel:Number` [lecture seule] | Valeur numérique spécifiant l'intensité de mouvement requise pour appeler `Camera.onActivity(true)`. |
| | `motionTimeOut:Number` [lecture seule] | Nombre de millisecondes entre le moment où la caméra arrête la détection du mouvement et le moment où `Camera.onActivity(false)` est appelé. |
| | `muted:Boolean` [lecture seule] | Valeur booléenne spécifiant si l'utilisateur a refusé l'accès à la caméra (`true`) ou autorisé l'accès (`false`) dans le panneau Paramètres de contrôle de Flash Player. |
| | `name:String` [lecture seule] | Chaîne spécifiant le nom de la caméra actuelle, tel que renvoyé par le matériel. |
| `static` | `names:Array` [lecture seule] | Récupère un tableau de chaînes reflétant les noms de toutes les caméras disponibles sans afficher le panneau Paramètres de contrôle de Flash Player. |
| | `quality:Number` [lecture seule] | Entier spécifiant le niveau de qualité d'image requis, tel que déterminé par le taux de compression appliqué à chaque image vidéo. |
| | `width:Number` [lecture seule] | La largeur de capture actuelle, en pixels. |

*Propriétés héritées de la classe Object*

| |
|---|
| constructeur (propriété Object.constructor), __proto__ (Object.__proto__ property), prototype (Object.prototype, propriété), __resolve (Object.__resolve, propriété) |

## Résumé des événements

| Evénement | Description |
|---|---|
| `onActivity = function(active:Boolean) {}` | Gestionnaire d'événements : invoqué lorsque la caméra commence ou arrête la détection du mouvement. |
| `onStatus = function(infoObject:Object) {}` | Gestionnaire d'événements : invoqué lorsque l'utilisateur autorise ou refuse l'accès à la caméra. |

## Résumé de la méthode

| Modificateurs | Signature | Description |
|---|---|---|
| `static` | `get([index:Number]) : Camera` | Renvoie une référence à un objet Camera pour capturer de la vidéo. |
| | `setMode([width:Number], [height:Number], [fps:Number], [favorArea:Boolean]) : Void` | Définit le mode de capture de la caméra sur le mode natif qui remplit le mieux les conditions requises. |
| | `setMotionLevel([motionLevel:Number], [timeOut:Number]) : Void` | Spécifie la quantité de mouvement requise pour appeler `Camera.onActivity(true)`. |
| | `setQuality([bandwidth:Number], [quality:Number]) : Void` | Définit le montant maximum de bande passante par seconde ou la qualité d'image requise des données vidéo sortantes actuelles. |

addProperty (méthode Object.addProperty), hasOwnProperty (méthode Object.hasOwnProperty), isPropertyEnumerable (méthode Object.isPropertyEnumerable), isPrototypeOf (méthode Object.isPrototypeOf), registerClass (méthode Object.registerClass), toString (méthode Object.toString), unwatch (méthode Object.unwatch), valueOf (méthode Object.valueOf), watch (méthode Object.watch)

# activityLevel (propriété Camera.activityLevel)

`public activityLevel : Number [lecture seule]`

Une valeur numérique spécifiant la quantité de mouvement détectée par la caméra. Les valeurs sont comprises entre 0 (aucun mouvement n'est détecté) et 100 (une grande intensité de mouvement est détectée). La valeur de cette propriété peut vous aider à déterminer s'il est nécessaire de transmettre un paramètre à `Camera.setMotionLevel()`.

Si la caméra est disponible mais n'est pas encore utilisée car `Video.attachVideo()` n'a pas été appelé, cette propriété est définie sur -1.

Si vous effectuez uniquement l'émission d'une vidéo non compressée en local, cette propriété est définie uniquement si vous avez affecté une fonction au gestionnaire d'événements `Camera.onActivity`. Dans le cas contraire, elle est non définie.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

## Exemple

Cet exemple indique la quantité de mouvement détectée par la caméra à l'aide de la propriété `activityLevel` et d'une occurrence ProgressBar. Créez une nouvelle occurrence vidéo en sélectionnant Nouvelle vidéo dans le menu d'options de la bibliothèque. Ajoutez une occurrence intitulée `my_video` à la scène. Ajoutez une occurrence de composant ProgressBar intitulée `activity_pb` à la scène. Ajoutez ensuite le code ActionScript suivant à l'image 1 du scénario :

```
// video instance on the Stage.
var my_video:Video;
var activity_pb:mx.controls.ProgressBar;
var my_cam:Camera = Camera.get();
my_video.attachVideo(my_cam);
activity_pb.mode = "manual";
activity_pb.label = "Activity %3%%";

this.onEnterFrame = function() {
  activity_pb.setProgress(my_cam.activityLevel, 100);
};
```

```
my_cam.onActivity = function(isActive:Boolean) {
  var themeColor:String = (isActive) ? "haloGreen" : "haloOrange";
  activity_pb.setStyle("themeColor", themeColor);
};
```

### Voir également

motionLevel (propriété Camera.motionLevel), setMotionLevel (méthode Camera.setMotionLevel )

## bandwidth (propriété Camera.bandwidth)

```
public bandwidth : Number [lecture seule]
```

Un entier spécifiant la quantité maximale de bande passante pouvant être utilisée par la vidéo sortante actuelle, en octets. Une valeur de 0 signifie que la vidéo Flash peut utiliser autant de bande passante que nécessaire pour conserver la qualité d'image voulue.

Pour configurer cette propriété, appelez `Camera.setQuality()`.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

### Exemple

L'exemple suivant modifie la quantité maximale de bande passante utilisée par la caméra. Créez une nouvelle occurrence vidéo en sélectionnant Nouvelle vidéo dans le menu d'options de la bibliothèque. Ajoutez une occurrence intitulée `my_video` à la scène. Ajoutez une occurrence de composant NumericStepper intitulée `bandwidth_nstep` à la scène. Ajoutez ensuite le code ActionScript suivant à l'image 1 du scénario :

```
var bandwidth_nstep:mx.controls.NumericStepper;
var my_video:Video;
var my_cam:Camera = Camera.get();
my_video.attachVideo(my_cam);
this.createTextField("bandwidth_txt", this.getNextHighestDepth(), 0, 0,
  100, 22);
bandwidth_txt.autoSize = true;
this.onEnterFrame = function() {
  bandwidth_txt.text = "Camera is currently using "+my_cam.bandwidth+"
  bytes ("+Math.round(my_cam.bandwidth/1024)+" KB) bandwidth.";
};
//
bandwidth_nstep.minimum = 0;
bandwidth_nstep.maximum = 128;
bandwidth_nstep.stepSize = 16;
bandwidth_nstep.value = my_cam.bandwidth/1024;
function changeBandwidth(evt:Object) {
  my_cam.setQuality(evt.target.value 1024, 0);
}
```

```
bandwidth_nstep.addEventListener("change", changeBandwidth);
```

La méthode `MovieClip.getNextHighestDepth()` utilisée dans cet exemple requiert Flash Player 7 ou version ultérieure. Si votre fichier SWF comprend un composant de version 2, utilisez la classe DepthManager avec composants version 2 plutôt que la méthode `MovieClip.getNextHighestDepth()`

### Voir également

`setQuality (méthode Camera.setQuality)`

## currentFps (propriété Camera.currentFps)

```
public currentFps : Number [lecture seule]
```

Cadence à laquelle la caméra capture des données, en images par seconde. Cette propriété ne peut pas être définie ; toutefois, vous pouvez utiliser la méthode `Camera.setMode()` pour définir une propriété connexe, `Camera.fps`, spécifiant la cadence maximale à laquelle vous souhaitez que la caméra capture les données.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

### Exemple

L'exemple suivant détecte la cadence, en images par seconde, à laquelle la caméra capture des données à l'aide de la propriété `currentFps` et d'une occurrence ProgressBar. Créez une nouvelle occurrence vidéo en sélectionnant Nouvelle vidéo dans le menu d'options de la bibliothèque. Ajoutez une occurrence intitulée `my_video` à la scène. Ajoutez une occurrence de composant ProgressBar intitulée fps_pb à la scène. Ajoutez ensuite le code ActionScript suivant à l'image 1 du scénario :

```
var my_video:Video;
var fps_pb:mx.controls.ProgressBar;
var my_cam:Camera = Camera.get();
my_video.attachVideo(my_cam);
this.onEnterFrame = function() {
  fps_pb.setProgress(my_cam.fps-my_cam.currentFps, my_cam.fps);
};

fps_pb.setStyle("fontSize", 10);
fps_pb.setStyle("themeColor", "haloOrange");
fps_pb.labelPlacement = "top";
fps_pb.mode = "manual";
fps_pb.label = "FPS: %2 (%3%% dropped)";
```

### Voir également

`setMode (méthode Camera.setMode)`, `fps (propriété Camera.fps)`

# fps (propriété Camera.fps)

`public fps : Number [lecture seule]`

Cadence maximale à laquelle vous voulez que la caméra capture des données, en images par seconde. La cadence maximale possible dépend des fonctionnalités de la caméra : si la caméra ne prend pas en charge la valeur définie ici, cette cadence ne sera pas atteinte.

- Pour définir une valeur souhaitée pour cette propriété, utilisez `Camera.setMode()`.
- Pour déterminer la cadence à laquelle la caméra capture actuellement les données, utilisez la propriété `Camera.currentFps`.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

## Exemple

L'exemple suivant détecte la cadence, en images par seconde, à laquelle la caméra capture des données à l'aide de la propriété `currentFps` et d'une occurrence ProgressBar. Créez une nouvelle occurrence vidéo en sélectionnant Nouvelle vidéo dans le menu d'options de la bibliothèque. Ajoutez une occurrence intitulée `my_video` à la scène. Ajoutez une occurrence de composant ProgressBar intitulée fps_pb à la scène. Ajoutez ensuite le code ActionScript suivant à l'image 1 du scénario :

```
var my_video:Video;
var fps_pb:mx.controls.ProgressBar;
var my_cam:Camera = Camera.get();
my_video.attachVideo(my_cam);
this.onEnterFrame = function() {
  fps_pb.setProgress(my_cam.fps-my_cam.currentFps, my_cam.fps);
};

fps_pb.setStyle("fontSize", 10);
fps_pb.setStyle("themeColor", "haloOrange");
fps_pb.labelPlacement = "top";
fps_pb.mode = "manual";
fps_pb.label = "FPS: %2 (%3%% dropped)";
```

*Remarque :* La fonction `setMode()` ne garantit pas de fournir le paramètre fps requis ; elle définit la propriété fps que vous avez demandée ou la plus rapide parmi celles disponibles.

## Voir également

`currentFps (propriété Camera.currentFps)`, `setMode (méthode Camera.setMode)`

# get (méthode Camera.get)

```
public static get([index:Number]) : Caméra
```

Renvoie une référence à un objet Camera pour capturer de la vidéo. Pour commencer la capture de la vidéo, vous devez relier l'objet Camera à un objet vidéo (consultez `Video.attachVideo()`).

Contrairement aux objets que vous pouvez créer à l'aide du constructeur `new`, plusieurs appels de `Camera.get()` font référence à la même caméra. Ainsi, si votre script contient les lignes `first_cam = Camera.get()` et `second_cam = Camera.get()`, à la fois `first_cam` et `second_cam` font référence à la même caméra (par défaut).

En général, évitez de transmettre une valeur pour *`index`* ; contentez-vous d'utiliser `Camera.get()` pour renvoyer une référence à la caméra par défaut. Via le panneau Paramètres de la caméra (comme indiqué plus bas dans cette section), l'utilisateur peut spécifier la caméra que Flash doit utiliser par défaut. Si vous transmettez une valeur pour *`index`*, vous pouvez essayer de référencer une caméra autre que celle que l'utilisateur préfère utiliser. Vous pouvez utiliser *`index`* en de rares occasions, par exemple si votre application capture de la vidéo à partir de deux caméras simultanément.

Lorsqu'un fichier SWF tente d'accéder à la caméra renvoyée par `Camera.get()`, Flash Player affiche une boîte de dialogue Confidentialité permettant à l'utilisateur d'autoriser ou de refuser l'accès à la caméra. (Assurez-vous que la taille de votre scène est d'au moins 215 x 138 pixels ; il s'agit de la taille minimale requise par Flash pour afficher la boîte de dialogue.)

Lorsque l'utilisateur répond à cette boîte de dialogue, le gestionnaire d'événements `Camera.onStatus` renvoie un objet d'informations qui indique la réponse de l'utilisateur. Pour déterminer si l'utilisateur a refusé ou autorisé l'accès à la caméra sans traiter ce gestionnaire d'événements, utilisez la propriété `Camera.muted`.

L'utilisateur peut également spécifier des paramètres de confidentialité permanents pour un domaine spécifique. Pour ce faire, il lui suffit de cliquer avec le bouton droit (Windows) ou d'appuyer sur la touche Contrôle (Macintosh) lors de la lecture d'un fichier SWF, de pointer sur Paramètres, d'ouvrir le panneau Contrôle de l'accès, puis de sélectionner Mémoriser.

Vous ne pouvez pas utiliser ActionScript pour définir la valeur Autoriser ou Refuser d'un utilisateur, mais vous pouvez afficher le panneau Confidentialité pour l'utilisateur via `System.showSettings(0)`. Si l'utilisateur sélectionne Mémoriser, Flash Player n'affiche plus la boîte de dialogue Confidentialité pour les fichiers SWF de ce domaine.

Si `Camera.get` renvoie la valeur `null`, cela signifie que la caméra est utilisée par une autre application ou qu'aucune caméra n'est installée sur le système. Pour déterminer si une caméra est installée, utilisez `Camera.names.length`. Pour afficher le panneau de paramètres Flash Player Camera, qui permet à l'utilisateur de choisir la caméra à référencer par `Camera.get()`, utilisez `System.showSettings(3)`.

L'analyse du matériel des caméras nécessite un certain temps. Lorsque Flash détecte au moins une caméra, le matériel n'est plus analysé pendant la durée de vie de l'occurrence du lecteur. Cependant, si Flash ne détecte aucune caméra, il effectuera une analyse à chaque fois que `Camera.get` est appelé. Cela est particulièrement utile si un utilisateur a oublié de connecter la caméra ; si votre fichier SWF contient un bouton Réessayer permettant d'appeler `Camera.get`, Flash peut rechercher la caméra sans que l'utilisateur ne doive redémarrer le fichier SWF.

*Remarque :* La syntaxe correcte est `Camera.get()`. Pour affecter l'objet Camera à une variable, utilisez une syntaxe comme `active_cam = Camera.get()` :

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

## Paramètres

*`index`*`:Number` [facultatif] - Entier de base zéro spécifiant la caméra à sélectionner, comme déterminé dans le tableau renvoyé par la propriété `Camera.names`. Pour obtenir la caméra par défaut (ce qui est recommandé pour la plupart des applications), omettez ce paramètre.

## Valeur renvoyée

`Camera` - Si `index` n'est pas spécifié, cette méthode renvoie une référence à la caméra par défaut ou, si elle est utilisée par une autre application, à la première caméra disponible. (Si plusieurs caméras sont installées, l'utilisateur peut spécifier la caméra par défaut dans le panneau Paramètres de la caméra de Flash Player.) Si aucune caméra n'est disponible ou installée, la méthode renvoie `null`. Si `index` est spécifié, cette méthode renvoie une référence à la caméra demandée ou `null` si elle n'est pas disponible.

## Exemple

L'exemple suivant vous permet de sélectionner une caméra active à utiliser à partir d'une occurrence ComboBox. La caméra actuellement active s'affiche dans une occurrence du composant Label. Créez une nouvelle occurrence vidéo en sélectionnant Nouvelle vidéo dans le menu d'options de la bibliothèque. Ajoutez une occurrence intitulée `my_video` à la scène. Ajoutez une occurrence du composant Label intitulée camera_lbl à la scène, ainsi qu'une occurrence du composant ComboBox intitulée `cameras_cb`. Ajoutez ensuite le code ActionScript suivant à l'image 1 du scénario :

```
var my_cam:Camera = Camera.get();
```

```
var my_video:Video;
my_video.attachVideo(my_cam);
var camera_lbl:mx.controls.Label;
var cameras_cb:mx.controls.ComboBox;
camera_lbl.text = my_cam.name;
cameras_cb.dataProvider = Camera.names;
function changeCamera():Void {
  my_cam = Camera.get(cameras_cb.selectedIndex);
  my_video.attachVideo(my_cam);
  camera_lbl.text = my_cam.name;
}
cameras_cb.addEventListener("change", changeCamera);
camera_lbl.setStyle("fontSize", 9);
cameras_cb.setStyle("fontSize", 9);
```

### Voir également

index (propriété Camera.index), muted (propriété Camera.muted), names (propriété Camera.names), onStatus (gestionnaire Camera.onStatus), setMode (méthode Camera.setMode), showSettings (méthode System.showSettings), attachVideo (méthode Video.attachVideo)

## height (propriété Camera.height)

```
public height : Number [lecture seule]
```

La hauteur de capture actuelle, en pixels. Pour définir une valeur souhaitée pour cette propriété, utilisez `Camera.setMode()`.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

### Exemple

Le code suivant affiche la largeur, la hauteur et la valeur FPS actuelles d'une occurrence vidéo dans une occurrence du composant Label sur la scène. Créez une nouvelle occurrence vidéo en sélectionnant Nouvelle vidéo dans le menu d'options de la bibliothèque. Ajoutez une occurrence intitulée `my_video` à la scène. Ajoutez une occurrence du composant Label intitulée dimensions_lbl à la scène. Ajoutez ensuite le code ActionScript suivant à l'image 1 du scénario :

```
var my_cam:Camera = Camera.get();
var my_video:Video;
my_video.attachVideo(my_cam);
var dimensions_lbl:mx.controls.Label;
dimensions_lbl.setStyle("fontSize", 9);
dimensions_lbl.setStyle("fontWeight", "bold");
dimensions_lbl.setStyle("textAlign", "center");
```

```
dimensions_lbl.text = "width: "+my_cam.width+", height: "+my_cam.height+",
  FPS: "+my_cam.fps;
```

Consultez également l'exemple relatif à `Camera.setMode()`.

### Voir également

largeur (propriété Camera.width), setMode (méthode Camera.setMode)

## index (propriété Camera.index)

```
public index : Number [lecture seule]
```

Entier de base zéro spécifiant l'indice de la caméra, tel qu'indiqué dans le tableau renvoyé par `Camera.names`.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

### Exemple

L'exemple suivant affiche un tableau de caméras dans un champ de texte créé lors de l'exécution et vous indique quelle caméra vous utilisez actuellement. Créez une nouvelle occurrence vidéo en sélectionnant Nouvelle vidéo dans le menu d'options de la bibliothèque. Ajoutez une occurrence intitulée `my_video` à la scène. Ajoutez une occurrence du composant Label intitulée camera_lbl à la scène. Ajoutez ensuite le code ActionScript suivant à l'image 1 du scénario :

```
var camera_lbl:mx.controls.Label;
var my_cam:Camera = Camera.get();
var my_video:Video;
my_video.attachVideo(my_cam);

camera_lbl.text = my_cam.index+". "+my_cam.name;
this.createTextField("cameras_txt", this.getNextHighestDepth(), 25, 160,
  160, 80);
cameras_txt.html = true;
cameras_txt.border = true;
cameras_txt.wordWrap = true;
cameras_txt.multiline = true;
for (var i = 0; i<Camera.names.length; i++) {
  cameras_txt.htmlText += "<li><u><a
  href=\"asfunction:changeCamera,"+i+"\">"+Camera.names[i]+"</a></u></
  li>";
}
function changeCamera(index:Number) {
  my_cam = Camera.get(index);
  my_video.attachVideo(my_cam);
  camera_lbl.text = my_cam.index+". "+my_cam.name;
}
```

La méthode `MovieClip.getNextHighestDepth()` utilisée dans cet exemple requiert Flash Player 7 ou version ultérieure. Si votre fichier SWF comprend un composant de version 2, utilisez la classe DepthManager avec composants version 2 plutôt que la méthode `MovieClip.getNextHighestDepth()`

### Voir également

`names (propriété Camera.names)`, `get (méthode Camera.get)`

## motionLevel (propriété Camera.motionLevel)

`public motionLevel : Number [lecture seule]`

Valeur numérique spécifiant l'intensité de mouvement requise pour appeler `Camera.onActivity(true)`. Les valeurs acceptables sont comprises entre 0 et 100. La valeur par défaut est 50.

La vidéo peut être affichée quelle que soit la valeur de la propriété `motionLevel`. Pour plus d'informations, reportez-vous à `Camera.setMotionLevel()`.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

### Exemple

L'exemple suivant détecte continuellement le niveau de mouvement d'une caméra. Créez une nouvelle occurrence vidéo en sélectionnant Nouvelle vidéo dans le menu d'options de la bibliothèque. Ajoutez une occurrence intitulée `my_video` à la scène. Ajoutez une occurrence du composant Label intitulée motionLevel_lbl, une occurrence du composant NumericStepper intitulée `motionLevel_nstep` et une occurrence du composant ProgressBar intitulée motion_pb à la scène. Ajoutez ensuite le code ActionScript suivant à l'image 1 du scénario :

```
var my_cam:Camera = Camera.get();
var my_video:Video;
my_video.attachVideo(my_cam);

// configure the ProgressBar component instance
var motion_pb:mx.controls.ProgressBar;
motion_pb.mode = "manual";
motion_pb.label = "Motion: %3%%";

var motionLevel_lbl:mx.controls.Label;
// configure the NumericStepper component instance
var motionLevel_nstep:mx.controls.NumericStepper;
motionLevel_nstep.minimum = 0;
motionLevel_nstep.maximum = 100;
motionLevel_nstep.stepSize = 5;
```

```
motionLevel_nstep.value = my_cam.motionLevel;

// Continuously update the progress of the ProgressBar component instance to
   the activityLevel
// of the current Camera instance, which is defined in my_cam
this.onEnterFrame = function() {
   motion_pb.setProgress(my_cam.activityLevel, 100);
};

// When the level of activity goes above or below the number defined in
   Camera.motionLevel,
// trigger the onActivity event handler.
my_cam.onActivity = function(isActive:Boolean) {
   // If isActive equals true, set the themeColor variable to "haloGreen".
   // Otherwise set the themeColor to "haloOrange".
   var themeColor:String = (isActive) ? "haloGreen" : "haloOrange";
   motion_pb.setStyle("themeColor", themeColor);
};

function changeMotionLevel() {
   // Set the motionLevel property for my_cam Camera instance to the value
   of the NumericStepper
   // component instance. Maintain the current motionTimeOut value of the
   my_cam Camera instance.
   my_cam.setMotionLevel(motionLevel_nstep.value, my_cam.motionTimeOut);
}
motionLevel_nstep.addEventListener("change", changeMotionLevel);
```

### Voir également

onActivity (gestionnaire Camera.onActivity), onStatus (gestionnaire Camera.onStatus), setMotionLevel (méthode Camera.setMotionLevel ), activityLevel (propriété Camera.activityLevel)

## motionTimeOut (propriété Camera.motionTimeOut)

```
public motionTimeOut : Number [lecture seule]
```

Nombre de millisecondes entre le moment où la caméra arrête la détection du mouvement et le moment où `Camera.onActivity(false)` est appelé. La valeur par défaut est 2 000 (2 secondes).

Pour configurer cette propriété, appelez `Camera.setMotionLevel()`.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

## Exemple

Dans l'exemple suivant, l'occurrence ProgressBar change sa couleur de thème Halo lorsque le niveau de l'activité est inférieur au niveau de mouvement. Vous pouvez définir le nombre de secondes pour la propriété `motionTimeout` à l'aide d'une occurrence NumericStepper. Créez une nouvelle occurrence vidéo en sélectionnant Nouvelle vidéo dans le menu d'options de la bibliothèque. Ajoutez une occurrence intitulée `my_video` à la scène. Ajoutez une occurrence du composant Label intitulée motionLevel_lbl, une occurrence du composant NumericStepper intitulée motionTimeOut_nstep et une occurrence du composant ProgressBar intitulée motion_pb à la scène. Ajoutez ensuite le code ActionScript suivant à l'image 1 du scénario :

```
var motionLevel_lbl:mx.controls.Label;
var motion_pb:mx.controls.ProgressBar;
var motionTimeOut_nstep:mx.controls.NumericStepper;
var my_cam:Camera = Camera.get();
var my_video:Video;
my_video.attachVideo(my_cam);

this.onEnterFrame = function() {
  motionLevel_lbl.text = "activityLevel: "+my_cam.activityLevel;
};

motion_pb.indeterminate = true;
my_cam.onActivity = function(isActive:Boolean) {
  if (isActive) {
  motion_pb.setStyle("themeColor", "haloGreen");
  motion_pb.label = "Motion is above "+my_cam.motionLevel;
  } else {
  motion_pb.setStyle("themeColor", "haloOrange");
  motion_pb.label = "Motion is below "+my_cam.motionLevel;
  }
};
function changeMotionTimeOut() {
  my_cam.setMotionLevel(my_cam.motionLevel, motionTimeOut_nstep.value
  1000);
}
motionTimeOut_nstep.addEventListener("change", changeMotionTimeOut);
motionTimeOut_nstep.value = my_cam.motionTimeOut/1000;
```

## Voir également

`setMotionLevel (méthode Camera.setMotionLevel )`, `onActivity (gestionnaire Camera.onActivity)`

# muted (propriété Camera.muted)

```
public muted : Boolean [lecture seule]
```

Valeur booléenne spécifiant si l'utilisateur a refusé l'accès à la caméra (`true`) ou autorisé l'accès (`false`) dans le panneau Paramètres de contrôle de Flash Player. Lorsque cette valeur change, `Camera.onStatus` est appelé. Pour plus d'informations, reportez-vous à `Camera.get()`.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

## Exemple

Dans l'exemple suivant, un message d'erreur peut s'afficher si `my_cam.muted` renvoie true. Créez une nouvelle occurrence vidéo en sélectionnant Nouvelle vidéo dans le menu d'options de la bibliothèque. Ajoutez une occurrence intitulée `my_video` à la scène. Ajoutez ensuite le code ActionScript suivant à l'image 1 du scénario :

```
var my_cam:Camera = Camera.get();
var my_video:Video;
my_video.attachVideo(my_cam);
my_cam.onStatus = function(infoObj:Object) {
  if (my_cam.muted) {
  // If user is denied access to their Camera, you can display an error
  message here. You can display the user's Camera/Privacy settings again
  using System.showSettings(0);
  trace("User denied access to Camera");
  System.showSettings(0);
  }
};
```

## Voir également

`get (méthode Camera.get)`, `onStatus (gestionnaire Camera.onStatus)`

# name (propriété Camera.name)

```
public name : String [lecture seule]
```

Chaîne spécifiant le nom de la caméra actuelle, tel que renvoyé par le matériel.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

### Exemple

L'exemple suivant affiche le nom de la caméra par défaut dans un champ de texte. Sous Windows, ce nom est identique à celui du périphérique répertorié dans le panneau de configuration Scanneurs et appareils photo. Créez une nouvelle occurrence vidéo en sélectionnant Nouvelle vidéo dans le menu d'options de la bibliothèque. Ajoutez une occurrence intitulée `my_video` à la scène. Ajoutez ensuite le code ActionScript suivant à l'image 1 du scénario :

```
var my_cam:Camera = Camera.get();
var my_video:Video;
my_video.attachVideo(my_cam);

this.createTextField("name_txt", this.getNextHighestDepth(), 0, 0, 100,
  22);
name_txt.autoSize = true;
name_txt.text = my_cam.name;
```

La méthode `MovieClip.getNextHighestDepth()` utilisée dans cet exemple requiert Flash Player 7 ou version ultérieure. Si votre fichier SWF comprend un composant de version 2, utilisez la classe DepthManager avec composants version 2 plutôt que la méthode `MovieClip.getNextHighestDepth()`

### Voir également

`get (méthode Camera.get)`, `names (propriété Camera.names)`

## names (propriété Camera.names)

```
public static names : Array [lecture seule]
```

Récupère un tableau de chaînes reflétant les noms de toutes les caméras disponibles sans afficher le panneau Paramètres de contrôle de Flash Player. Ce tableau se comporte de la même manière que tout autre tableau ActionScript, fournissant de façon implicite l'index basé sur zéro de chaque caméra et le nombre de caméras présentes sur le système (via `Camera.names.length`). Pour plus d'informations, consultez l'entrée de classe Array de `Camera.names`.

L'appel de la propriété `Camera.names` nécessite un examen minutieux du matériel et plusieurs secondes peuvent être nécessaires pour créer le tableau. Dans la plupart des cas, vous pouvez utiliser la caméra par défaut.

*Remarque :* La syntaxe correcte est `Camera.names`. Pour affecter la valeur renvoyée à une variable, utilisez une syntaxe comme *`cam_array`* `= Camera.names`: Pour déterminer le nom de la caméra en cours, utilisez *`active_cam`* `.name`.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

### Exemple

L'exemple suivant utilise la caméra par défaut sauf si plusieurs caméras sont disponibles. Dans ce cas, l'utilisateur peut sélectionner la caméra qu'il souhaite définir par défaut. Si une seule caméra est disponible, alors la caméra par défaut est utilisée. Créez une nouvelle occurrence vidéo en sélectionnant Nouvelle vidéo dans le menu d'options de la bibliothèque. Ajoutez une occurrence intitulée `my_video` à la scène. Ajoutez ensuite le code ActionScript suivant à l'image 1 du scénario :

```
var my_video:Video;
var cam_array:Array = Camera.names;
if (cam_array.length>1) {
  System.showSettings(3);
}
var my_cam:Camera = Camera.get();
my_video.attachVideo(my_cam);
```

### Voir également

`get (méthode Camera.get)`, `index (propriété Camera.index)`, `name (propriété Camera.name)`

## onActivity (gestionnaire Camera.onActivity)

onActivity = function(active:Boolean) {}

Gestionnaire d'événements : invoqué lorsque la caméra commence ou arrête la détection du mouvement. Si vous souhaitez répondre à ce gestionnaire d'événements, vous devez créer une fonction pour traiter sa valeur d'activité.

Pour spécifier l'intensité de mouvement requise pour appeler `Camera.onActivity(true)` et la durée qui doit s'écouler sans activité avant d'appeler `Camera.on(false)`, utilisez `Camera.setMotionLevel()`.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

### Paramètres

*`active`*`:Boolean` - Valeur booléenne définie sur true quand la caméra commence à détecter du mouvement, sur false quand le mouvement s'arrête.

### Exemple

L'exemple suivant affiche la valeur true ou false dans le panneau de sortie selon que la caméra commence ou cesse la détection du mouvement :

```
// Assumes a Video object named "myVideoObject" is on the Stage
active_cam = Camera.get();
```

```
myVideoObject.attachVideo(active_cam);
active_cam.setMotionLevel(10, 500);
active_cam.onActivity = function(mode)
{
  trace(mode);
}
```

### Voir également

`setMotionLevel (méthode Camera.setMotionLevel )`

## onStatus (gestionnaire Camera.onStatus)

onStatus = function(infoObject:Object) {}

Gestionnaire d'événements : invoqué lorsque l'utilisateur autorise ou refuse l'accès à la caméra. Si vous souhaitez répondre à ce gestionnaire d'événements, vous devez créer une fonction pour traiter l'objet d'informations généré par la caméra.

Lorsqu'un fichier SWF tente d'accéder à la caméra, Flash Player affiche une boîte de dialogue Confidentialité permettant à l'utilisateur d'autoriser ou de refuser l'accès.

- Si l'utilisateur autorise l'accès, la propriété `Camera.muted` est définie sur false, et ce gestionnaire est appelé avec un objet d'informations dont la propriété de code est « Camera.Unmuted » et la propriété de niveau « Status ».
- Si l'utilisateur refuse l'accès, la propriété `Camera.muted` est définie sur true, et ce gestionnaire est appelé avec un objet d'informations dont la propriété de code est « Camera.Muted » et la propriété de niveau « Status ».

Pour déterminer si l'utilisateur a refusé ou autorisé l'accès à la caméra sans traiter ce gestionnaire d'événements, utilisez la propriété `Camera.muted`.

Remarque : Si l'utilisateur choisit d'autoriser ou de refuser définitivement l'accès à tous les fichiers SWF d'un domaine spécifié, ce gestionnaire n'est pas appelé pour les fichiers SWF de ce domaine sauf si l'utilisateur modifie ultérieurement le paramètre de confidentialité. Pour plus d'informations, reportez-vous à `Camera.get()`.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

### Paramètres

*`infoObject`*`:Object` - Paramètre défini selon le message de statut.

### Exemple

Le code ActionScript suivant est utilisé pour afficher un message à chaque fois que l'utilisateur autorise ou refuse l'accès à la caméra :

```
var my_cam:Camera = Camera.get();
var my_video:Video;
my_video.attachVideo(my_cam);
my_cam.onStatus = function(infoObj:Object) {
  switch (infoObj.code) {
  case 'Camera.Muted' :
  trace("Camera access is denied");
  break;
  case 'Camera.Unmuted' :
  trace("Camera access granted");
  break;
  }
}
```

### Voir également

`get (méthode Camera.get)`, `muted (propriété Camera.muted)`, `showSettings (méthode System.showSettings)`, `onStatus (gestionnaire System.onStatus)`

## qualité (propriété Camera.quality)

```
public quality : Number [lecture seule]
```

Entier spécifiant le niveau de qualité requis pour l'image, tel que déterminé par le taux de compression appliqué à chaque image vidéo. Les valeurs de qualité acceptables sont comprises entre 1 (qualité la plus médiocre, compression maximale) et 100 (qualité optimale, pas de compression). La valeur par défaut est 0, ce qui signifie que la qualité de l'image peut varier si nécessaire pour éviter de dépasser la bande passante disponible.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

### Exemple

L'exemple suivant utilise une occurrence NumericStepper pour spécifier le taux de compression appliqué à la caméra. Créez une nouvelle occurrence vidéo en sélectionnant Nouvelle vidéo dans le menu d'options de la bibliothèque. Ajoutez une occurrence intitulée `my_video` à la scène. Ajoutez une occurrence NumericStepper intitulée quality_nstep. Ajoutez ensuite le code ActionScript suivant à l'image 1 du scénario :

```
var quality_nstep:mx.controls.NumericStepper;

var my_cam:Camera = Camera.get();
var my_video:Video;
my_video.attachVideo(my_cam);
```

```
quality_nstep.minimum = 0;
quality_nstep.maximum = 100;
quality_nstep.stepSize = 5;
quality_nstep.value = my_cam.quality;

function changeQuality() {
  my_cam.setQuality(my_cam.bandwidth, quality_nstep.value);
}
quality_nstep.addEventListener("change", changeQuality);
```

### Voir également

setQuality (méthode Camera.setQuality)

## setMode (méthode Camera.setMode)

```
public setMode([width:Number], [height:Number], [fps:Number],
  [favorArea:Boolean]) : Void
```

Définit le mode de capture de la caméra sur le mode natif qui remplit le mieux les conditions requises. Si la caméra ne dispose pas d'un mode natif correspondant à tous les paramètres que vous transmettez, Flash sélectionne un mode de capture qui synthétise le mieux le mode demandé. Cette manipulation peut nécessiter le découpage de l'image et l'omission d'images.

Par défaut, Flash omet des images si nécessaire pour conserver la taille de l'image. Pour réduire le nombre d'images omises, même si cela passe par une réduction de la taille de l'image, définissez le paramètre `favorArea` sur `false`.

Lorsqu'il choisit un mode natif, Flash essaie de conserver les proportions demandées dans la mesure du possible. Par exemple, si vous appelez la commande *`active_cam`*`.setMode(400, 400, 30)`, et si les valeurs de largeur et de hauteur maximales disponibles sur la caméra sont 320 et 288, Flash définit la largeur et la hauteur sur 288 ; en attribuant la même valeur à ces propriétés, Flash conserve la proportion 1:1 que vous avez demandée.

Pour déterminer les valeurs affectées à ces propriétés une fois la sélection du mode qui correspond le mieux aux valeurs que vous avez demandées par Flash, utilisez `Camera.width`, `Camera.height`, et `Camera.fps`.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

### Paramètres

*`width`*`:Number` [facultatif] - Largeur de capture demandée, en pixels. La valeur par défaut est 160.

*`height`*`:Number` [facultatif] - Hauteur de capture demandée, en pixels. La valeur par défaut est 120.

*fps*:Number [facultatif] - Cadence à laquelle la caméra doit capturer des données, en images par seconde. La valeur par défaut est 15.

*favorArea*:Boolean [facultatif] - Valeur booléenne indiquant comment manipuler la largeur, la hauteur et la cadence si la caméra n'est pas dotée d'un mode natif qui remplit les conditions requises. La valeur par défaut est `true`, ce qui signifie que le maintien de la taille de capture est favorisé ; l'utilisation de ce paramètre permet de sélectionner le mode qui correspond le mieux aux valeurs *width* et *height* , même si cela affecte les performances en réduisant la cadence. Pour optimiser la cadence au détriment de la hauteur et de la largeur de la caméra, définissez le paramètre `favorArea` sur `false`.

## Exemple

L'exemple suivant définit le mode de capture de la caméra. Vous pouvez entrer une cadence dans une occurrence TextInput et appuyer sur Entrée ou sur Retour pour l'appliquer. Créez une nouvelle occurrence vidéo en sélectionnant Nouvelle vidéo dans le menu d'options de la bibliothèque. Ajoutez une occurrence intitulée `my_video` à la scène. Ajoutez une occurrence du composant TextInput intitulée fps_ti. Ajoutez ensuite le code ActionScript suivant à l'image 1 du scénario :

```
var my_cam:Camera = Camera.get();
var my_video:Video;
my_video.attachVideo(my_cam);

fps_ti.maxChars = 2;
fps_ti.restrict = [0-9];
fps_lbl.text = "Current: "+my_cam.fps+" fps";

function changeFps():Void {
  my_cam.setMode(my_cam.width, my_cam.height, fps_ti.text);
  fps_lbl.text = "Current: "+my_cam.fps+" fps";
  fps_ti.text = my_cam.fps;
  Selection.setSelection(0,2);
}
fps_ti.addEventListener("enter", changeFps);
```

## Voir également

fps (propriété Camera.fps), height (propriété Camera.height), largeur (propriété Camera.width), currentFps (propriété Camera.currentFps)

# setMotionLevel (méthode Camera.setMotionLevel )

`public setMotionLevel([motionLevel:Number], [timeOut:Number]) : Void`

Spécifie la quantité de mouvement requise pour appeler `Camera.onActivity(true)`. Peut éventuellement définir le nombre de millisecondes qui doit s'écouler sans activité avant que Flash considère que le mouvement a cessé et puisse appeler `Camera.onActivity(false)`.

*Remarque :* La vidéo peut être affichée quelle que soit la valeur du paramètre `sensitivity`. Ce paramètre détermine uniquement à quel moment et dans quelles circonstances `Camera.onActivity` est appelé (ce n'est pas le cas lorsque la vidéo est capturée ou affichée).

- Pour empêcher la caméra de détecter le mouvement, définissez le paramètre *`sensitivity`* sur la valeur 100 ; `Camera.onActivity` n'est jamais appelé. (Vous utiliserez probablement cette valeur à des fins de tests uniquement : par exemple, pour désactiver temporairement des actions qui doivent se produire lorsque `Camera.onActivity` est appelé.)
- Pour déterminer l'intensité de mouvement actuellement détectée par la caméra, utilisez la propriété `Camera.activityLevel`.
- Les valeurs de sensibilité de mouvement correspondent directement aux valeurs d'activité. La valeur d'activité d'une absence de mouvement totale est 0. La valeur d'activité d'un mouvement constant est 100. Votre valeur d'activité est inférieure à votre valeur de sensibilité de mouvement lorsque vous n'effectuez pas de déplacement ; lorsque vous effectuez un déplacement, les valeurs d'activité dépassent fréquemment votre valeur de sensibilité de mouvement.
- L'objectif de cette méthode est similaire à celui de `Microphone.setSilenceLevel()` ; les deux méthodes sont utilisées pour spécifier à quel moment il convient d'appeler le gestionnaire d'événements `onActivity`. Toutefois, l'impact de ces méthodes sur la publication des flux continus diffère de manière significative :
- `Microphone.setSilenceLevel()` est conçu pour optimiser la bande passante. Lorsqu'un flux continu est considéré comme étant silencieux, aucune donnée audio n'est envoyée. En revanche, un message unique est envoyé, indiquant le début du silence.
- `Camera.setMotionLevel()` est conçu pour détecter le mouvement et n'a aucune incidence sur l'utilisation de la bande passante. Même si un flux vidéo ne détecte pas le mouvement, la vidéo est toujours envoyée.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

## Paramètres

*`motionLevel`*`:Number` [facultatif] - Valeur numérique spécifiant l'intensité de mouvement requise pour appeler `Camera.onActivity(true)`. Les valeurs acceptables sont comprises entre 0 et 100. La valeur par défaut est 50.

*timeOut*:Number [facultatif] - Paramètre numérique spécifiant le nombre de millisecondes qui doit s'écouler sans activité avant que Flash considère que l'activité a cessé et puisse appeler le gestionnaire d'événement `Camera.onActivity(false)`. La valeur par défaut est 2 000 (2 secondes).

### Exemple

L'exemple suivant envoie des messages au panneau de Sortie lorsque l'activité vidéo commence ou s'arrête. La valeur de sensibilité de mouvement étant de 30, augmentez-la ou diminuez-la pour voir comment différentes valeurs affectent la détection de mouvement.

```
// Assumes a Video object named "myVideoObject" is on the Stage
active_cam = Camera.get();
x = 0;
function motion(mode) {
  trace(x + ": " + mode);
  x++;
}
active_cam.onActivity = function(mode) {
  motion(mode);
}
active_cam.setMotionLevel(30, 500);
myVideoObject.attachVideo(active_cam);
```

### Voir également

`motionLevel (propriété Camera.motionLevel)`, `motionTimeOut (propriété Camera.motionTimeOut)`, `onActivity (gestionnaire Camera.onActivity)`, `activityLevel (propriété Camera.activityLevel)`

## setQuality (méthode Camera.setQuality)

```
public setQuality([bandwidth:Number], [quality:Number]) : Void
```

Définit le montant maximum de bande passante par seconde ou la qualité d'image requise des données vidéo sortantes actuelles. Cette méthode ne peut être généralement appliquée que si vous transmettez la vidéo via Flash Communication Server.

Utilisez cette méthode pour spécifier l'élément de la vidéo sortante le plus important pour votre application, utilisation de bande passante ou qualité de l'image.

- Pour indiquer que l'utilisation de la bande passante est prioritaire, transmettez une valeur à *bandwidth* et attribuez la valeur 0 à *frameQuality*. Flash transmet une vidéo de qualité optimale dans la bande passante spécifiée. Si nécessaire, Flash réduit la qualité de l'image afin d'éviter de dépasser la bande passante spécifiée. En général, plus le mouvement augmente, plus la qualité diminue.

- Pour indiquer que la qualité est prioritaire, transmettez la valeur 0 à *bandwidth* et une valeur numérique à *frameQuality*. Flash utilise autant de bande passante que nécessaire pour conserver la qualité spécifiée. Si nécessaire, Flash réduit la cadence pour conserver la qualité de l'image. En général, plus le mouvement augmente, plus l'utilisation de la bande passante augmente également.
- Pour spécifier que la bande passante et la qualité sont aussi importantes l'une que l'autre, transmettez des valeurs numériques aux deux paramètres. Flash transmet alors de la vidéo permettant de respecter la qualité requise et ne dépassant pas la bande passante spécifiée. Si nécessaire, Flash réduit la cadence pour conserver la qualité de l'image sans dépasser la bande passante spécifiée.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

## Paramètres

*bandwidth*:Number [facultatif] - Entier spécifiant la quantité maximale de bande passante pouvant être utilisée par la vidéo sortante actuelle, en octets par seconde. Pour spécifier que la vidéo Flash peut utiliser autant de bande passante que nécessaire pour conserver la valeur de *frameQuality*, attribuez la valeur 0 à *bandwidth*. La valeur par défaut est 16 384.

*quality*:Number [facultatif] - Entier spécifiant le niveau de qualité d'image requis, tel que déterminé par le taux de compression appliqué à chaque image vidéo. Les valeurs acceptables sont comprises entre 1 (qualité la plus médiocre, compression maximale) et 100 (qualité optimale, pas de compression). Pour spécifier que la qualité d'image peut varier autant que nécessaire afin d'éviter de dépasser la quantité de bande passante disponible, attribuez la valeur 0 à *quality*. La valeur par défaut est 0.

## Exemple

Les exemples suivants indiquent comment utiliser cette méthode afin de contrôler l'utilisation de la bande passante et la qualité d'image.

```
// Ensure that no more than 8192 (8K/second) is used to send video
active_cam.setQuality(8192,0);

// Ensure that no more than 8192 (8K/second) is used to send video
// with a minimum quality of 50
active_cam.setQuality(8192,50);

// Ensure a minimum quality of 50, no matter how much bandwidth it takes
active_cam.setQuality(0,50);
```

## Voir également

get (méthode Camera.get), qualité (propriété Camera.quality), bandwidth (propriété Camera.bandwidth)

## largeur (propriété Camera.width)

```
public quality : Number [lecture seule]
```

La largeur de capture actuelle, en pixels. Pour définir une valeur souhaitée pour cette propriété, utilisez `Camera.setMode()`.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

### Exemple

Le code suivant affiche la largeur, la hauteur et la valeur IPS (FPS) actuelles d'une occurrence vidéo dans une occurrence du composant Label sur la scène. Créez une nouvelle occurrence vidéo en sélectionnant Nouvelle vidéo dans le menu d'options de la bibliothèque. Ajoutez une occurrence intitulée `my_video` à la scène. Ajoutez une occurrence du composant Label intitulée dimensions_lbl à la scène. Ajoutez ensuite le code ActionScript suivant à l'image 1 du scénario :

```
var my_cam:Camera = Camera.get();
var my_video:Video;
my_video.attachVideo(my_cam);
var dimensions_lbl:mx.controls.Label;
dimensions_lbl.setStyle("fontSize", 9);
dimensions_lbl.setStyle("fontWeight", "bold");
dimensions_lbl.setStyle("textAlign", "center");
dimensions_lbl.text = "width: "+my_cam.width+", height: "+my_cam.height+",
  FPS: "+my_cam.fps;
```

Consultez aussi l'exemple relatif à `Camera.setMode()`.

### Voir également

`height (propriété Camera.height)`, `setMode (méthode Camera.setMode)`

# capabilities (System.capabilities)

```
Object
  |
  +-System.capabilities

public class capabilities
extends Object
```

La classe Capabilities permet de déterminer les fonctionnalités du système et le lecteur hébergeant un fichier SWF, vous permettant d'adapter le contenu à différents formats. Par exemple, l'écran d'un téléphone portable (noir et blanc, 100 pixels carrés) diffère de l'écran couleurs de 1 000 pixels carrés d'un PC. Pour fournir le contenu approprié au plus grand nombre d'utilisateurs possible, vous pouvez utiliser l'objet `System.capabilities` afin de déterminer le type de périphérique dont dispose un utilisateur. Vous pouvez ensuite demander au serveur d'envoyer différents fichiers SWF en fonction des fonctionnalités propres à chaque périphérique ou indiquer au fichier SWF de modifier sa présentation en fonction des fonctionnalités du périphérique.

Vous pouvez envoyer les informations relatives aux fonctionnalités à l'aide de la méthode HTTP `GET` ou `POST`. L'exemple suivant montre une chaîne de serveur pour un ordinateur prenant en charge les fichiers MP3, doté d'une résolution de 1 600 x 1 200 pixels, exécutant Windows XP et Flash Player 8 (8.0.0.0) :

```
A=t&SA=t&SV=t&EV=t&MP3=t&AE=t&VE=t&ACC=f&PR=t&SP=t&

SB=f&DEB=t&V=WIN%208%2C0%2C0%2C0&M=Macromedia%20Windows&

R=1600x1200&DP=72&COL=color&AR=1.0&OS=Windows%20XP&

L=en&PT=External&AVD=f&LFD=f&WD=f"
```

Toutes les propriétés de l'objet `System.capabilities` sont en lecture seule.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

## Résumé des propriétés

| Modificateurs | Propriété | Description |
|---|---|---|
| static | avHardwareDisable:Boolean [lecture seule] | Valeur booléenne spécifiant si l'accès à la caméra et au microphone de l'utilisateur est interdit administrativement (true) ou autorisé (false). |
| static | hasAccessibility:Boolean [lecture seule] | Valeur booléenne définie sur true si le lecteur s'exécute dans un environnement qui prend en charge la communication entre Flash Player et les options d'accessibilité ; sinon définie sur false. |
| static | hasAudio:Boolean [lecture seule] | Spécifie si le système est doté de fonctionnalités audio. |
| static | hasAudioEncoder:Boolean [lecture seule] | Spécifie si Flash Player peut coder un flux audio continu. |
| static | hasEmbeddedVideo:Boolean [lecture seule] | Valeur booléenne définie sur true si le lecteur s'exécute sur un système qui prend en charge la vidéo intégrée ; sinon définie sur false. |
| static | hasIME:Boolean [lecture seule] | Indique si le système est doté d'un IME, (éditeur de méthode d'entrée). |
| static | hasMP3:Boolean [lecture seule] | Spécifie si le système est doté d'un décodeur MP3. |
| static | hasPrinting:Boolean [lecture seule] | Valeur booléenne définie sur true si le lecteur s'exécute sur un système qui prend en charge l'impression ; définie sur false sinon. |
| static | hasScreenBroadcast:Boolean [lecture seule] | Valeur booléenne définie sur true si le lecteur prend en charge le développement des applications de diffusion sur écran devant être exécutées via Flash Communication Server ; définie sur false sinon. |
| static | hasScreenPlayback:Boolean [lecture seule] | Valeur booléenne définie sur true si le lecteur prend en charge la lecture des applications de diffusion sur écran exécutées via Flash Communication Server ; définie sur false sinon. |
| static | hasStreamingAudio:Boolean [lecture seule] | Valeur booléenne définie sur true si le lecteur peut lire les sons en flux continu ; définie sur false sinon. |
| static | hasStreamingVideo:Boolean [lecture seule] | Valeur booléenne définie sur true si le lecteur peut lire les vidéos en flux continu ; sinon définie sur false. |
| static | hasVideoEncoder:Boolean [lecture seule] | Spécifie si Flash Player peut coder un flux vidéo. |

| Modificateurs | Propriété | Description |
|---|---|---|
| static | isDebugger:Boolean [lecture seule] | Valeur booléenne indiquant si le lecteur est une version officielle (false) ou une version de débogage spéciale (true). |
| static | language:String [lecture seule] | Indique la langue du système sur lequel s'exécute le lecteur. |
| static | localFileReadDisable :Boolean [lecture seule] | Valeur booléenne indiquant si l'accès en lecture au disque dur de l'utilisateur est interdit administrativement (true) ou autorisé (false). |
| static | manufacturer:String [lecture seule] | Chaîne indiquant le fabricant de Flash Player, au format "Macromedia *OSName*" (*OSName* peut être "Windows", "Macintosh", "Linux", ou "Other OS Name"). |
| static | os:String [lecture seule] | Chaîne indiquant le système d'exploitation actuel. |
| static | pixelAspectRatio:Num ber [lecture seule] | Entier indiquant les proportions de l'écran, en pixels. |
| static | playerType:String [lecture seule] | Chaîne indiquant le type de lecteur. |
| static | screenColor:String [lecture seule] | Chaîne indiquant la couleur d'écran. |
| static | screenDPI:Number [lecture seule] | Nombre indiquant la résolution en points par pouce (ppp) de l'écran, en pixels. |
| static | screenResolutionX:Nu mber [lecture seule] | Entier indiquant la résolution horizontale maximale de l'écran. |
| static | screenResolutionY:Nu mber [lecture seule] | Entier indiquant la résolution verticale maximale de l'écran. |
| static | serverString:String [lecture seule] | Chaîne de code URL spécifiant les valeurs de chaque propriété System.capabilities. |
| static | version:String [lecture seule] | Chaîne contenant la plate-forme Flash Player et les informations sur la version (par exemple, "WIN 8,0,0,0"). |

*Propriétés héritées de la classe Object*

constructeur (propriété Object.constructor), __proto__ (Object.__proto__ property), prototype (Object.prototype, propriété), __resolve (Object.__resolve, propriété)

### Résumé de la méthode

*Méthodes héritées de la classe Object*

| |
|---|
| addProperty (méthode Object.addProperty), hasOwnProperty (méthode Object.hasOwnProperty), isPropertyEnumerable (méthode Object.isPropertyEnumerable), isPrototypeOf (méthode Object.isPrototypeOf), registerClass (méthode Object.registerClass), toString (méthode Object.toString), unwatch (méthode Object.unwatch), valueOf (méthode Object.valueOf), watch (méthode Object.watch) |

## avHardwareDisable (propriété capabilities.avHardwareDisable)

```
public static avHardwareDisable : Boolean [lecture seule]
```

Valeur booléenne spécifiant si l'accès à la caméra et au microphone de l'utilisateur est interdit administrativement (`true`) ou autorisé (`false`). La chaîne de serveur est `AVD`.

**Disponibilité :** ActionScript 1.0 ; Flash Player 7

### Exemple

L'exemple suivant présente la valeur de cette propriété en lecture seule :

```
trace(System.capabilities.avHardwareDisable);
```

### Voir également

get (méthode Camera.get), obtenir (méthode Microphone.get), showSettings (méthode System.showSettings)

## hasAccessibility (propriété capabilities.hasAccessibility)

```
public static hasAccessibility : Boolean [lecture seule]
```

Valeur booléenne définie sur `true` si le lecteur s'exécute dans un environnement qui prend en charge la communication entre Flash Player et les options d'accessibilité ; sinon définie sur `false`. La chaîne de serveur est `ACC`.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

### Exemple

L'exemple suivant présente la valeur de cette propriété en lecture seule :

```
trace(System.capabilities.hasAccessibility);
```

### Voir également

`isActive (méthode Accessibility.isActive)`, `updateProperties (méthode Accessibility.updateProperties)`,

## hasAudio (propriété capabilities.hasAudio)

`public static hasAudio : Boolean [lecture seule]`

Spécifie si le système est doté de fonctionnalités audio. Valeur booléenne définie sur `true` si le lecteur s'exécute sur un système doté de fonctionnalités audio ; sinon définie sur `false`. La chaîne de serveur est `A`.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

### Exemple

L'exemple suivant présente la valeur de cette propriété en lecture seule :

```
trace(System.capabilities.hasAudio);
```

## hasAudioEncoder (propriété capabilities.hasAudioEncoder)

`public static hasAudioEncoder : Boolean [lecture seule]`

Spécifie si Flash Player peut coder un flux continu. Valeur booléenne définie sur `true` si le lecteur peut coder un flux continu, tel que celui provenant d'un microphone ; sinon définie sur `false`. La chaîne de serveur est `AE`.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

### Exemple

L'exemple suivant présente la valeur de cette propriété en lecture seule :

```
trace(System.capabilities.hasAudioEncoder);
```

## hasEmbeddedVideo (propriété capabilities.hasEmbeddedVideo)

`public static hasEmbeddedVideo : Boolean [lecture seule]`

Valeur booléenne définie sur `true` si le lecteur s'exécute sur un système qui prend en charge la vidéo intégrée ; sinon définie sur `false`. La chaîne de serveur est `EV`.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6,0,65,0

### Exemple

L'exemple suivant présente la valeur de cette propriété en lecture seule :

```
trace(System.capabilities.hasEmbeddedVideo);
```

## hasIME (propriété capabilities.hasIME)

```
public static hasIME : Boolean [lecture seule]
```

Indique si le système est doté d'un IME (éditeur de méthode d'entrée). Une valeur `true` indique que le lecteur s'exécute sur un système doté d'un IME ; une valeur `false` indique qu'aucun IME n'est installé. La chaîne de serveur est `IME`.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

### Exemple

L'exemple suivant positionne l'IME sur `ALPHANUMERIC_FULL` si le lecteur utilise un système sur lequel un IME est installé.

```
if(System.capabilities.hasIME) {
  trace(System.IME.getConversionMode());
  System.IME.setConversionMode(System.IME.ALPHANUMERIC_FULL);
  trace(System.IME.getConversionMode());
}
```

## hasMP3 (propriété capabilities.hasMP3)

```
public static hasMP3 : Boolean [lecture seule]
```

Spécifie si le système est doté d'un décodeur MP3. Valeur booléenne définie sur `true` si le lecteur s'exécute sur un système doté d'un décodeur MP3 ; sinon définie sur `false`. La chaîne de serveur est `MP3`.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

### Exemple

L'exemple suivant présente la valeur de cette propriété en lecture seule :

```
trace(System.capabilities.hasMP3);
```

## hasPrinting (propriété capabilities.hasPrinting)

```
public static hasPrinting : Boolean [lecture seule]
```

Valeur booléenne définie sur `true` si le lecteur s'exécute sur un système qui prend en charge l'impression ; définie sur `false` sinon. La chaîne de serveur est `PR`.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6,0,65,0

### Exemple

L'exemple suivant présente la valeur de cette propriété en lecture seule :

```
trace(System.capabilities.hasPrinting);
```

## hasScreenBroadcast (propriété capabilities.hasScreenBroadcast)

```
public static hasScreenBroadcast : Boolean [lecture seule]
```

Valeur booléenne définie sur `true` si le lecteur prend en charge le développement des applications de diffusion sur écran devant être exécutées via Flash Communication Server ; définie sur `false` sinon. La chaîne de serveur est `SB`.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6,0,79,0

### Exemple

L'exemple suivant présente la valeur de cette propriété en lecture seule :

```
trace(System.capabilities.hasScreenBroadcast);
```

## hasScreenPlayback (propriété capabilities.hasScreenPlayback)

```
public static hasScreenPlayback : Boolean [lecture seule]
```

Valeur booléenne définie sur `true` si le lecteur prend en charge la lecture des applications de diffusion sur écran exécutées via Flash Communication Server ; définie sur `false` sinon. La chaîne de serveur est `SP`.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6,0,79,0

### Exemple

L'exemple suivant présente la valeur de cette propriété en lecture seule :

```
trace(System.capabilities.hasScreenPlayback);
```

## hasStreamingAudio (propriété capabilities.hasStreamingAudio)

```
public static hasStreamingAudio : Boolean [lecture seule]
```

Valeur booléenne définie sur `true` si le lecteur peut lire les sons en flux continu ; sinon définie sur `false`. La chaîne de serveur est `SA`.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6,0,65,0

### Exemple

L'exemple suivant présente la valeur de cette propriété en lecture seule :

```
trace(System.capabilities.hasStreamingAudio);
```

## hasStreamingVideo (propriété capabilities.hasStreamingVideo)

```
public static hasStreamingVideo : Boolean [lecture seule]
```

Valeur booléenne définie sur `true` si le lecteur peut lire les vidéos en flux continu ; sinon définie sur `false`. La chaîne de serveur est `SV`.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6,0,65,0

### Exemple

L'exemple suivant présente la valeur de cette propriété en lecture seule :

```
trace(System.capabilities.hasStreamingVideo);
```

## hasVideoEncoder (propriété capabilities.hasVideoEncoder)

```
public static hasVideoEncoder : Boolean [lecture seule]
```

Spécifie si Flash Player peut coder un flux vidéo. Valeur booléenne définie sur `true` si le lecteur peut coder un flux vidéo, tel que celui provenant d'une caméra Web ; définie sur `false` sinon. La chaîne de serveur est `VE`.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

### Exemple

L'exemple suivant présente la valeur de cette propriété en lecture seule :

```
trace(System.capabilities.hasVideoEncoder);
```

## isDebugger (propriété capabilities.isDebugger)

```
public static isDebugger : Boolean [lecture seule]
```

Valeur booléenne indiquant si le lecteur est une version officielle (`false`) ou une version de débogage spéciale (`true`). La chaîne de serveur est `DEB`.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

### Exemple

L'exemple suivant présente la valeur de cette propriété en lecture seule :

```
trace(System.capabilities.isDebugger);
```

## langage (propriété capabilities.language)

```
public static language : String [lecture seule]
```

Indique la langue du système sur lequel s'exécute le lecteur. Cette propriété est spécifiée sous forme de code de langue à deux lettres en minuscules selon ISO 639-1. Pour le chinois, une balise secondaire de code pays à deux lettres en majuscules supplémentaire selon ISO 3166 permet de faire la distinction entre le chinois simplifié et traditionnel. Les langues, elles-mêmes, sont nommées avec des balises en anglais. Par exemple, `fr` signifie Français.

Cette propriété a été modifiée en deux points pour Flash Player 7. Premièrement, le code de langue des systèmes en anglais n'inclut plus le code pays. Dans Flash Player 6, tous les systèmes en anglais renvoyaient le code de langue et la balise secondaire de code pays à deux lettres (`en-US`). Dans Flash Player 7, les systèmes en anglais renvoient uniquement le code de langue (`en`). Deuxièmement, sur les systèmes Microsoft Windows, cette propriété renvoie désormais la langue de l'interface utilisateur (IU). Dans Flash Player 6 sur la plate-forme Microsoft Windows, `System.capabilities.language` renvoie les paramètres régionaux utilisateur, permettant de sélectionner les paramètres de mise en forme des dates, heures, symboles monétaires et nombres élevés. Dans Flash Player 7 sur la plate-forme Microsoft Windows, cette propriété renvoie désormais la langue de l'interface utilisateur, qui se réfère à la langue utilisée pour tous les menus, boîtes de dialogue, messages d'erreur et fichiers d'aide. Le tableau suivant répertorie les valeurs possibles :

| **Langue** | **Balise** |
|---|---|
| Tchèque | cs |
| Danois | da |
| Néerlandais | nl |
| Anglais | en |
| Finnois | fi |
| Français | fr |
| Allemand | de |
| Hongrois | hu |
| Italien | it |
| Japonais | ja |

| Langue | Balise |
|---|---|
| Coréen | ko |
| Norvégien | no |
| Autre/inconnu | xu |
| Polonais | pl |
| Portugais | pt |
| Russe | ru |
| Chinois simplifié | zh-CN |
| Espagnol | es |
| Suédois | sv |
| Chinois traditionnel | zh-TW |
| Turc | tr |

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

### Exemple

L'exemple suivant présente la valeur de cette propriété en lecture seule :

```
trace(System.capabilities.language);
```

## localFileReadDisable (propriété capabilities.localFileReadDisable)

```
public static localFileReadDisable : Boolean [lecture seule]
```

Valeur booléenne indiquant si l'accès en lecture au disque dur de l'utilisateur est interdit administrativement (`true`) ou autorisé (`false`). Si la propriété est définie sur `true`, , Flash Player ne peut pas lire de fichiers (y compris le premier fichier SWF de démarrage de Flash Player) sur le disque dur de l'utilisateur. Par exemple, toute tentative de lecture d'un fichier sur le disque dur de l'utilisateur à l'aide de `XML.load()`, `LoadMovie()`, ou `LoadVars.load()` échouera si cette propriété est définie sur `true`.

La lecture de bibliothèques partagées à l'exécution sera également bloquée si cette propriété est définie sur `true` ; en revanche, la lecture d'objets partagés localement est autorisée, indépendamment de la valeur de cette propriété. La chaîne de serveur est `LFD`.

**Disponibilité :** ActionScript 1.0 ; Flash Player 7

### Exemple

L'exemple suivant présente la valeur de cette propriété en lecture seule :

```
trace(System.capabilities.localFileReadDisable);
```

## manufacturer (propriété capabilities.manufacturer)

```
public static manufacturer : String [lecture seule]
```

Chaîne indiquant le fabricant de Flash Player, au format "`Macromedia OSName`" (*`OSName`* peut être "`Windows`", "`Macintosh`", "`Linux`", ou "`Other OS Name`"). La chaîne de serveur est `M`.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

### Exemple

L'exemple suivant présente la valeur de cette propriété en lecture seule :

```
trace(System.capabilities.manufacturer);
```

## os (propriété capabilities.os)

```
public static os : String [lecture seule]
```

Chaîne indiquant le système d'exploitation actuel. La propriété `os` peut renvoyer les chaînes suivantes : "`Windows XP`", "`Windows 2000`", "`Windows NT`", "`Windows 98/ME`", "`Windows 95`", "`Windows CE`" (disponible seulement en version Flash Player SDK, et pas en version de bureau), "`Linux`", et "`MacOS`". La chaîne de serveur est `OS`.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

### Exemple

L'exemple suivant présente la valeur de cette propriété en lecture seule :

```
trace(System.capabilities.os);
```

## pixelAspectRatio (propriété capabilities.pixelAspectRatio)

```
public static pixelAspectRatio : Number [lecture seule]
```

Entier indiquant les proportions de l'écran, en pixels. La chaîne de serveur est `AR`.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

### Exemple

L'exemple suivant présente la valeur de cette propriété en lecture seule :

```
trace(System.capabilities.pixelAspectRatio);
```

## playerType (propriété capabilities.playerType)

```
public static playerType : String [lecture seule]
```

Chaîne indiquant le type de lecteur. Cette propriété peut avoir l'une des valeurs suivantes :

- `"StandAlone"` pour le Flash StandAlone Player
- `"External"` pour la version Flash Player utilisée par le lecteur externe, ou en mode Tester l'animation..
- `"PlugIn"` pour le module externe du navigateur Flash Player
- `"ActiveX"` pour le contrôle ActiveX de Flash Player utilisé par Microsoft Internet Explorer

La chaîne de serveur est `PT`.

**Disponibilité :** ActionScript 1.0 ; Flash Player 7

### Exemple

L'exemple suivant présente la valeur de cette propriété en lecture seule :

```
trace(System.capabilities.playerType);
```

## screenColor (propriété capabilities.screenColor)

```
public static screenColor : String [lecture seule]
```

Chaîne indiquant la couleur d'écran. Cette propriété peut avoir la valeur `"color"`, `"gray"` ou `"bw"`, représentant respectivement la couleur, les niveaux de gris, et le noir et blanc. La chaîne de serveur est `COL`.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

### Exemple

L'exemple suivant présente la valeur de cette propriété en lecture seule :

```
trace(System.capabilities.screenColor);
```

## screenDPI (propriété capabilities.screenDPI)

```
public static screenDPI : Number [lecture seule]
```

Nombre indiquant la résolution en points par pouce (ppp) de l'écran, en pixels. La chaîne de serveur est `DP`.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

### Exemple

L'exemple suivant présente la valeur de cette propriété en lecture seule :

```
trace(System.capabilities.screenDPI);
```

## screenResolutionX (propriété capabilities.screenResolutionX)

```
public static screenResolutionX : Number [lecture seule]
```

Entier indiquant la résolution horizontale maximale de l'écran. La chaîne de serveur est `R` (qui renvoie la largeur et la hauteur de l'écran).

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

### Exemple

L'exemple suivant présente la valeur de cette propriété en lecture seule :

```
trace(System.capabilities.screenResolutionX);
```

## screenResolutionX (propriété capabilities.screenResolutionX)

```
public static screenResolutionY : Number [lecture seule]
```

Entier indiquant la résolution verticale maximale de l'écran. La chaîne de serveur est `R` (qui renvoie la largeur et la hauteur de l'écran).

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

### Exemple

L'exemple suivant présente la valeur de cette propriété en lecture seule :

```
trace(System.capabilities.screenResolutionY);
```

## serverString (propriété capabilities.serverString)

```
public static serverString : String [lecture seule]
```

Chaîne de code URL spécifiant les valeurs de chaque propriété `System.capabilities`.

L'exemple suivant illustre une chaîne de code URL :

```
A=t&SA=t&SV=t&EV=t&MP3=t&AE=t&VE=t&ACC=f&PR=t&SP=t&
 SB=f&DEB=t&V=WIN%208%2C0%2C0%2C0&M=Macromedia%20Windows&
 R=1600x1200&DP=72&COL=color&AR=1.0&OS=Windows%20XP&
 L=en&PT=External&AVD=f&LFD=f&WD=f
```

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

### Exemple

L'exemple suivant présente la valeur de cette propriété en lecture seule :

```
trace(System.capabilities.serverString);
```

## version (propriété capabilities.version)

```
public static version : String [lecture seule]
```

Chaîne contenant la plate-forme Flash Player et les informations sur la version (par exemple, `"WIN 8,0,0,0"`). La chaîne de serveur est `V`.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

### Exemple

L'exemple suivant présente la valeur de cette propriété en lecture seule :

```
trace(System.capabilities.version);
```

# Color

```
Object
  |
  +-Color
```

```
public class Color
extends Object
```

*Déconseillé* depuis Flash Player 8. La classe Color a été déconseillée en faveur de la classe flash.geom.ColorTransform.

La classe Color vous permet de définir la valeur d'une couleur RVB et la transformation de couleurs des clips, puis de récupérer ces valeurs une fois définies.

Vous devez utiliser le constructeur `new Color()` pour créer un objet Color avant d'appeler ses méthodes.

**Disponibilité :** ActionScript 1.0 ; Flash Player 5

## Résumé des propriétés

*Propriétés héritées de la classe Object*

| |
|---|
| `constructeur (propriété Object.constructor), __proto__ (Object.__proto__ property), prototype (Object.prototype, propriété), __resolve (Object.__resolve, propriété)` |

## Résumé des constructeurs

| Signature | Description |
|---|---|
| `Color(target:Object)` | *Classe déconseillée.*<br>Crée un objet Color pour le clip spécifié par le paramètre *`target_mc`*. |

## Résumé de la méthode

| Modificateurs | Signature | Description |
|---|---|---|
| | `getRGB() : Number` | *Classe déconseillée.*<br>Renvoie la combinaison R+V+B actuellement utilisée par l'objet Color. |
| | `getTransform() : Object` | *Classe déconseillée.*<br>Renvoie la valeur de transformation définie par le dernier appel `Color.setTransform()`. |
| | `setRGB(offset:Number) : Void` | *Classe déconseillée.*<br>Spécifie une couleur RVB pour un objet Color. |
| | `setTransform(transformObject:Object) : Void` | *Classe déconseillée.*<br>Définit les informations relatives à la transformation de couleurs pour un objet Color. |

addProperty (méthode Object.addProperty), hasOwnProperty (méthode Object.hasOwnProperty), isPropertyEnumerable (méthode Object.isPropertyEnumerable), isPrototypeOf (méthode Object.isPrototypeOf), registerClass (méthode Object.registerClass), toString (méthode Object.toString), unwatch (méthode Object.unwatch), valueOf (méthode Object.valueOf), watch (méthode Object.watch)

# constructeur Color

```
public Color(target:Object)
```

*La classe Color est déconseillée depuis Flash Player 8.*

Crée un objet Color pour le clip spécifié par le paramètre *target_mc*. Vous pouvez alors utiliser les méthodes de cet objet Color pour modifier la couleur du clip cible entier.

**Disponibilité :** ActionScript 1.0 ; Flash Player 5

## Paramètres

*target*:Object - Nom d'occurrence d'un clip.

## Exemple

L'exemple suivant crée un objet Color intitulé my_color pour le clip my_mc et définit sa valeur RVB sur orange :

```
var my_color:Color = new Color(my_mc);
my_color.setRGB(0xff9933);
```

# getRGB (méthode Color.getRGB)

```
public getRGB() : Number
```

*La classe Color est déconseillée depuis Flash Player 8.*

Renvoie la combinaison R+V+B actuellement utilisée par l'objet Color.

**Disponibilité :** ActionScript 1.0 ; Flash Player 5

## Valeur renvoyée

Number - Nombre représentant la valeur numérique RVB de la couleur spécifiée.

### Exemple

Le code suivant récupère la valeur RVB de l'objet Color intitulé `my_color`, convertit la valeur en chaîne hexadécimale et l'affecte à la variable `myValue` Pour voir ce code fonctionner, ajoutez une occurrence de clip intitulée `my_mc` à la scène :

```
var my_color:Color = new Color(my_mc);
// set the color
my_color.setRGB(0xff9933);
var myValue:String = my_color.getRGB().toString(16);
// trace the color value
trace(myValue); // traces ff9933
```

### Voir également

`setRGB (méthode Color.setRGB)`

## getTransform (méthode Color.getTransform)

```
public getTransform() : Object
```

*La classe Color est déconseillée depuis Flash Player 8.*

Renvoie la valeur de transformation définie par le dernier appel `Color.setTransform()`.

**Disponibilité :** ActionScript 1.0 ; Flash Player 5

### Valeur renvoyée

`Object` - Objet dont les propriétés contiennent les valeurs actuelles de décalage et de pourcentage de la couleur spécifiée.

### Exemple

L'exemple suivant lit l'objet Transform et définit les nouveaux pourcentages de couleurs et la valeur alpha de `my_mc` par rapport à leurs valeurs actuelles. Pour voir ce code fonctionner, placez un clip multicolore portant le nom d'occurrence `my_mc` sur la scène. Ensuite, insérez le code suivant sur l'image 1 du scénario principal et sélectionnez Contrôle > Tester l'animation :

```
var my_color:Color = new Color(my_mc);
var myTransform:Object = my_color.getTransform();
myTransform = { ra: 50, ba: 50, aa: 30};
my_color.setTransform(myTransform);
```

Pour obtenir une description des paramètres relatifs à l'objet de transformation de couleurs, consultez `Color.setTransform()`.

### Voir également

setTransform (méthode Color.setTransform)

## setRGB (méthode Color.setRGB)

```
public setRGB(offset:Number) : Void
```

*La classe Color est déconseillée depuis Flash Player 8.*

Spécifie une couleur RVB pour un objet Color. L'appel de cette méthode remplace tout paramètre `Color.setTransform()` précédent.

**Disponibilité :** ActionScript 1.0 ; Flash Player 5

### Paramètres

*offset*:Number - 0x *RRGGBB* La valeur hexadécimale ou la couleur RVB à définir. Les valeurs *RR*, *GG*, et *BB* se composent chacune de deux chiffres hexadécimaux qui spécifient le décalage de chaque composant de couleur. La valeur `0x` indique au compilateur ActionScript que le nombre est une valeur hexadécimale.

### Exemple

Cet exemple définit la valeur de couleur RVB pour le clip `my_mc`. Pour voir ce code fonctionner, placez sur la scène un clip portant le nom d'occurrence `my_mc`. Ensuite, insérez le code suivant sur l'image 1 du scénario principal et sélectionnez Contrôle > Tester l'animation :

```
var my_color:Color = new Color(my_mc);
my_color.setRGB(0xFF0000); // my_mc turns red
```

### Voir également

setTransform (méthode Color.setTransform)

## setTransform (méthode Color.setTransform)

```
public setTransform(transformObject:Object) : Void
```

*La classe Color est déconseillée depuis Flash Player 8.*

Définit les informations relatives à la transformation de couleurs pour un objet Color. Le paramètre *colorTransformObject* est un objet générique que vous créez à partir du constructeur `new Object`. Il dispose de paramètres spécifiant les valeurs de pourcentage et de décalage des composants rouge, vert, bleu et alpha (transparence) d'une couleur, saisies au format 0xRRGGBBAA.

Les paramètres d'un objet de transformation de couleurs correspondent à ceux de la boîte de dialogue Effet avancé et sont définis comme suit :

- *ra* est le pourcentage du composant rouge (-100 à 100).
- *rb* est le décalage du composant rouge (-255 à 255).
- *ga* est le pourcentage du composant vert (-100 à 100).
- *gb* est le décalage du composant vert (-255 à 255).
- *ba* est le pourcentage du composant bleu (-100 à 100).
- *bb* est le décalage du composant bleu (-255 à 255).
- *aa* est le pourcentage pour alpha (-100 à 100).
- *ab* est le décalage pour alpha (-255 à 255).

Pour créer un paramètre *colorTransformObject*, procédez comme suit :

```
var myColorTransform:Object = new Object();
myColorTransform.ra = 50;
myColorTransform.rb = 244;
myColorTransform.ga = 40;
myColorTransform.gb = 112;
myColorTransform.ba = 12;
myColorTransform.bb = 90;
myColorTransform.aa = 40;
myColorTransform.ab = 70;
```

Vous pouvez également utiliser la syntaxe suivante pour créer un paramètre *colorTransformObject*:

```
var myColorTransform:Object = { ra: 50, rb: 244, ga: 40, gb: 112, ba: 12, bb: 90, aa: 40, ab:
    70}
```

**Disponibilité :** ActionScript 1.0 ; Flash Player 5

## Paramètres

*transformObject*:Object - Objet créé à partir du constructeur `new Object`. Les propriétés de cette occurrence de la classe Object permettant de spécifier les valeurs de transformation de couleurs doivent être les suivantes : `ra`, `rb`, `ga`, `gb`, `ba`, `bb`, `aa`, `ab`. Ces propriétés sont expliquées ci-dessous.

### Exemple

Cet exemple crée un nouvel objet Color pour un fichier SWF cible, un objet générique intitulé `myColorTransform` doté des propriétés définies ci-dessus et utilise la méthode `setTransform()` pour transmettre la valeur *`colorTransformObject`* à un objet Color. Pour utiliser ce code dans un document Flash (FLA), insérez-le sur l'image 1 du scénario principal, puis placez un clip portant le nom d'occurrence `my_mc` sur la scène, de la manière suivante :

```
// Create a color object called my_color for the target my_mc
var my_color:Color = new Color(my_mc);
// Create a color transform object called myColorTransform using
// Set the values for myColorTransform
var myColorTransform:Object = { ra: 50, rb: 244, ga: 40, gb: 112, ba: 12,
  bb: 90, aa: 40, ab: 70};
// Associate the color transform object with the Color object
// created for my_mc
my_color.setTransform(myColorTransform);
```

### Voir également

`Object`

# ColorMatrixFilter (flash.filters.ColorMatrixFilter)

```
Object
  |
  +-flash.filters.BitmapFilter
    |
    +-flash.filters.ColorMatrixFilter
```

`public class` *ColorMatrixFilter*
`extends BitmapFilter`

La classe ColorMatrixFilter vous permet d'appliquer une transformation de matrice 4 x 5 aux valeurs de couleur RVBA et alpha de chaque pixel de l'image d'entrée afin d'obtenir un résultat intégrant un nouvel ensemble de valeurs de couleur RVBA et alpha. Elle permet d'effectuer des modifications de saturation, des rotations de teinte, de définir la luminance de l'alpha et de produire d'autres effets. Vous pouvez appliquer ce filtre aux bitmaps et aux occurrences MovieClip.

L'utilisation de filtres dépend de l'objet auquel vous appliquez le filtre.

- Pour appliquer les filtres lors de l'exécution du clip, utilisez la propriété `filters`. Lorsque vous définissez la propriété `filters` d'un objet, celui-ci n'est pas modifié. En outre, vous pouvez l'annuler en supprimant la propriété `filters`.

- Pour appliquer des filtres aux occurrences BitmapData, utilisez la méthode `BitmapData.applyFilter()` . L'appel `applyFilter()` sur un objet BitmapData, utilise l'objet BitmapData d'origine ainsi que l'objet filtre pour générer une image filtrée.

Vous pouvez également appliquer des effets de filtre aux images et aux données vidéo pendant la programmation. Pour plus d'informations, consultez la documentation relative à la programmation.

Si vous appliquez un filtre à un clip ou à un bouton, la propriété `cacheAsBitmap` du clip ou du bouton est définie sur `true`. Si vous supprimez tous les filtres, la valeur d'origine de `cacheAsBitmap` est restaurée.

Les formules suivantes sont utilisées où a[0] à a[19] correspondent aux entrées 0 à 19 dans la matrice de propriété de tableau du vingtième élément :

```
redResult = a[0] * srcR + a[1] * srcG + a[2] * srcB + a[3] * srcA + a[4]
greenResult = a[5] * srcR + a[6] * srcG + a[7] * srcB + a[8] * srcA + a[9]
blueResult = a[10] * srcR + a[11] * srcG + a[12] * srcB + a[13] * srcA +
 a[14]
alphaResult = a[15] * srcR + a[16] * srcG + a[17] * srcB + a[18] * srcA +
 a[19]
```

Ce filtre sépare chaque pixel d'origine en composants rouge, vert, bleu et alpha comme suit : srcR, srcG, srcB, srcA. Pour finir, il associe de nouveau chaque composant de couleur pour former un pixel unique et renvoie le résultat.

Les calculs sont effectués sur des valeurs de couleur non multipliées. Si le graphique d'entrée est constitué de valeurs de couleur prémultipliées, celles-ci sont automatiquement converties en valeurs de couleur non multipliées en vue de cette opération.

Les deux modes optimisés suivants sont disponibles.

*Alpha uniquement.* Quand vous passez au filtre une matrice qui ajuste uniquement le composant alpha, comme indiqué ici, le filtre optimise ses performances :

```
1 0 0 0 0
0 1 0 0 0
0 0 1 0 0
0 0 0 N 0 (where N is between 0.0 and 1.0)
```

*Version plus rapide*. Disponible uniquement sur les processeurs dotés d'un accélérateur SSE/ Altivec (tels que Pentium 3 et version ultérieure, Apple G4 et version ultérieure).
L'accélérateur est utilisé quand les termes de multiplicateur se trouvent dans la plage comprise entre -15,99 et 15,99 et les termes d'additionneur a[4], a[9], a[14] et a[19] se trouvent dans la plage comprise entre -8 000 et 8 000.

Un filtre ne peut s' appliquer si l'image résultante dépasse 2 880 pixels en largeur ou en hauteur. Par exemple, si vous faites un zoom avant sur un grand clip auquel le filtre est appliqué, le filtre est désactivé si l'image résultante atteint la limite de 2 880 pixels.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

## Exemple

L'exemple suivant utilise BitmapFilter pour manipuler la saturation en couleur d'une image située à l'emplacement du pointeur de la souris. Si vous positionnez le pointeur dans le coin supérieur gauche (0,0), l'image ne doit pas être modifiée. Quand vous déplacez le pointeur vers la droite, les canaux vert et bleu sont enlevés en même temps de l'image. Quand vous déplacez le pointeur vers le bas, le canal rouge est enlevé. Si le pointeur est placé à l'angle inférieur droit de la scène, l'image doit être entièrement noire. Cet exemple suppose que vous ayez dans votre librairie une image avec son Linkage Identifier réglé sur « YourImageLinkage ».

```
import flash.filters.BitmapFilter;
import flash.filters.ColorMatrixFilter;

var image:MovieClip = this.attachMovie("YourImageLinkage", "YourImage",
  this.getNextHighestDepth());
image.cacheAsBitmap = true;

var listener:Object = new Object();
listener.image = image;
listener.onMouseMove = function() {
  var xPercent:Number = 1 - (_xmouse/Stage.width);
  var yPercent:Number = 1 - (_ymouse/Stage.height);
  var matrix:Array = new Array();
  matrix = matrix.concat([yPercent, 0, 0, 0, 0]); // red
  matrix = matrix.concat([0, xPercent, 0, 0, 0]); // green
  matrix = matrix.concat([0, 0, xPercent, 0, 0]); // blue
  matrix = matrix.concat([0, 0, 0, 1, 0]); // alpha

  var filter:BitmapFilter = new ColorMatrixFilter(matrix);
  image.filters = new Array(filter);
}

Mouse.addListener(listener);
listener.onMouseMove();
```

## Voir également

getPixel (méthode BitmapData.getPixel), applyFilter (méthode BitmapData.applyFilter), filters (propriété MovieClip.filters), cacheAsBitmap (propriété MovieClip.cacheAsBitmap)

## Résumé des propriétés

| Modificateurs | Propriété | Description |
|---|---|---|
| | `matrix:Array` | Un tableau de 20 éléments pour la transformation de couleurs 4 x 5. |

*Propriétés héritées de la classe Object*

`constructeur (propriété Object.constructor), __proto__ (Object.__proto__ property), prototype (Object.prototype, propriété), __resolve (Object.__resolve, propriété)`

## Résumé des constructeurs

| Signature | Description |
|---|---|
| `ColorMatrixFilter(matrix:Array)` | Initialise une nouvelle occurrence ColorMatrixFilter avec les paramètres spécifiés. |

## Résumé de la méthode

| Modificateurs | Signature | Description |
|---|---|---|
| | `clone() : ColorMatrixFilter` | Renvoie une copie de cet objet filtre. |

*Méthodes héritées de la classe BitmapFilter*

`clone (méthode BitmapFilter.clone )`

*Méthodes héritées de la classe Object*

`addProperty (méthode Object.addProperty), hasOwnProperty (méthode Object.hasOwnProperty), isPropertyEnumerable (méthode Object.isPropertyEnumerable), isPrototypeOf (méthode Object.isPrototypeOf), registerClass (méthode Object.registerClass), toString (méthode Object.toString), unwatch (méthode Object.unwatch), valueOf (méthode Object.valueOf), watch (méthode Object.watch)`

# clone (méthode ColorMatrixFilter.clone)

`public clone() : ColorMatrixFilter`

Renvoie une copie de cet objet filtre.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

## Valeur renvoyée

`flash.filters.ColorMatrixFilter` - Nouvelle occurrence ColorMatrixFilter dont les propriétés sont toutes identiques à celles de l'occurrence d'origine.

## Exemple

L'exemple suivant crée une nouvelle occurrence de ColorMatrixFilter et ensuite la clone en utilisant la méthode `clone`. La propriété `matrix` ne peut pas être changée directement (par exemple, `clonedFilter.matrix[2] = 1;`). Vous devez plutôt donner une référence au tableau, effectuer le changement, et restaurer la valeur en utilisant `clonedFilter.matrix = changedMatrix`.

```
import flash.filters.ColorMatrixFilter;

var matrix:Array = new Array();
matrix = matrix.concat([1, 0, 0, 0, 0]); // red
matrix = matrix.concat([0, 1, 0, 0, 0]); // green
matrix = matrix.concat([0, 0, 1, 0, 0]); // blue
matrix = matrix.concat([0, 0, 0, 1, 0]); // alpha

var filter:ColorMatrixFilter = new ColorMatrixFilter(matrix);
trace("filter: " + filter.matrix);

var clonedFilter:ColorMatrixFilter = filter.clone();
matrix = clonedFilter.matrix;
matrix[2] = 1;
clonedFilter.matrix = matrix;
trace("clonedFilter: " + clonedFilter.matrix);
```

# constructeur ColorMatrixFilter()

`public ColorMatrixFilter(matrix:Array)`

Initialise une nouvelle occurrence ColorMatrixFilter avec les paramètres spécifiés.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

## Paramètres

*`matrix`*`:Array` - Tableau de 20 éléments organisés en matrice de 4 x 5.

## matrix (propriété ColorMatrixFilter.matrix)

```
public matrix : Array
```

Un tableau de 20 éléments pour la transformation de couleurs 4 x 5.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

### Exemple

L'exemple suivant crée une nouvelle occurrence de ColorMatrixFilter et ensuite change sa propriété `matrix`. La propriété `matrix` ne peut pas être changée directement en modifiant sa valeur (par exemple, `clonedFilter.matrix[2] = 1;`). Vous devez plutôt donner une référence au tableau, effectuer le changement de la référence, et restaurer la valeur en utilisant `clonedFilter.matrix = changedMatrix`.

```
import flash.filters.ColorMatrixFilter;

var matrix:Array = new Array();
matrix = matrix.concat([1, 0, 0, 0, 0]); // red
matrix = matrix.concat([0, 1, 0, 0, 0]); // green
matrix = matrix.concat([0, 0, 1, 0, 0]); // blue
matrix = matrix.concat([0, 0, 0, 1, 0]); // alpha

var filter:ColorMatrixFilter = new ColorMatrixFilter(matrix);
trace("filter: " + filter.matrix);
var changedMatrix:Array = filter.matrix;
changedMatrix[2] = 1;
filter.matrix = changedMatrix;
trace("filter: " + filter.matrix);
```

# ColorTransform (flash.geom.ColorTransform)

```
Object
  |
  +-flash.geom.ColorTransform
```

```
public class ColorTransform
extends Object
```

La classe ColorTransform vous permet de régler de façon mathématique l'ensemble des valeurs de couleur dans un clip. La fonction de réglage des couleurs ou *transformation de couleur* peut être appliquée aux quatre canaux : rouge, vert, bleu et transparence alpha.

Lorsqu'un objet ColorTransform est appliqué à un clip, une nouvelle valeur est calculée pour chaque canal de couleur de la manière suivante :

- Nouvelle valeur de rouge = (ancienne valeur de rouge * `redMultiplier`)) + `redOffset`
- Nouvelle valeur de vert = (ancienne valeur de vert * `greenMultiplier`)) + `greenOffset`
- Nouvelle valeur de bleu = (ancienne valeur de bleu * `blueMultiplier`)) + `blueOffset`
- Nouvelle valeur alpha = (ancienne valeur alpha * `alphaMultiplier`)) + `alphaOffset`

Si l'une des valeurs de canal de couleur est supérieure à 255 une fois le calcul effectué, elle est définie sur 255. Si elle est inférieure à zéro, elle est définie sur zéro.

Vous devez utiliser le constructeur `new ColorTransform()` pour créer un objet ColorTransform afin de pouvoir appeler les méthodes de l'objet ColorTransform.

Les transformations de couleurs ne s'appliquent pas à la couleur d'arrière-plan d'un clip (tel qu'un objet SWF chargé). Elles s'appliquent uniquement aux graphiques et symboles associés au clip.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

## Voir également

`getTransform (méthode Color.getTransform)`, `setTransform (méthode Color.setTransform)`, `Transform (flash.geom.Transform)`

## Résumé des propriétés

| Modificateurs | Propriété | Description |
|---|---|---|
| | `alphaMultiplier:Number` | Une valeur décimale multipliée par la valeur du canal de transparence alpha. |
| | `alphaOffset:Number` | Un nombre, compris entre -255 et 255, qui est ajouté à la valeur du canal de transparence alpha après avoir été multiplié par la valeur `alphaMultiplier`. |
| | `blueMultiplier:Number` | Une valeur décimale multipliée par la valeur du canal de bleu. |
| | `blueOffset:Number` | Un nombre, compris entre -255 et 255, qui est ajouté à la valeur du canal de bleu après avoir été multiplié par la valeur `blueMultiplier`. |
| | `greenMultiplier:Number` | Une valeur décimale multipliée par la valeur du canal de vert. |
| | `greenOffset:Number` | Un nombre, compris entre -255 et 255, qui est ajouté à la valeur du canal de vert après avoir été multiplié par la valeur `greenMultiplier`. |
| | `redMultiplier:Number` | Une valeur décimale multipliée par la valeur du canal de rouge. |
| | `redOffset:Number` | Un nombre, compris entre -255 et 255, qui est ajouté à la valeur du canal de rouge après avoir été multiplié par la valeur `redMultiplier`. |
| | `rgb:Number` | La valeur de couleur RVB d'un objet ColorTransform. |

*Propriétés héritées de la classe Object*

`constructeur (propriété Object.constructor), __proto__ (Object.__proto__ property), prototype (Object.prototype, propriété), __resolve (Object.__resolve, propriété)`

## Résumé des constructeurs

| Signature | Description |
|---|---|
| `ColorTransform([redMultiplier:Number], [greenMultiplier:Number], [blueMultiplier:Number], [alphaMultiplier:Number], [redOffset:Number], [greenOffset:Number], [blueOffset:Number], [alphaOffset:Number])` | Crée un objet ColorTransform pour un objet d'affichage avec les paramètres RVB et alpha spécifiés. |

## Résumé de la méthode

| Modificateurs | Signature | Description |
|---|---|---|
| | `concat(second:ColorTransform) : Void` | Applique au clip une deuxième transformation additive de couleur. |
| | `toString() : String` | Formate et renvoie une chaîne qui décrit l'ensemble des propriétés de l'objet ColorTransform. |

*Méthodes héritées de la classe Object*

`addProperty (méthode Object.addProperty), hasOwnProperty (méthode Object.hasOwnProperty), isPropertyEnumerable (méthode Object.isPropertyEnumerable), isPrototypeOf (méthode Object.isPrototypeOf), registerClass (méthode Object.registerClass), toString (méthode Object.toString), unwatch (méthode Object.unwatch), valueOf (méthode Object.valueOf), watch (méthode Object.watch)`

# alphaMultiplier (propriété ColorTransform.alphaMultiplier)

`public alphaMultiplier : Number`

Une valeur décimale multipliée par la valeur du canal de transparence alpha.

Si vous définissez la valeur de transparence alpha d'un clip directement à l'aide de la propriété `MovieClip._alpha`, celle-ci affecte la valeur de la propriété `alphaMultiplier` de l'objet `ColorTransform` de ce clip.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

## Exemple

L'exemple suivant crée l'objet ColorTransform `colorTrans` et ajuste sa valeur `alphaMultiplier` de 1 à 0,5.

```
import flash.geom.ColorTransform;
import flash.geom.Transform;

var colorTrans:ColorTransform = new ColorTransform();
trace(colorTrans.alphaMultiplier); // 1

colorTrans.alphaMultiplier = .5;
trace(colorTrans.alphaMultiplier); // .5

var rect:MovieClip = createRectangle(20, 80, 0x000000);
var trans:Transform = new Transform(rect);
trans.colorTransform = colorTrans;

function createRectangle(width:Number, height:Number, color:Number,
  scope:MovieClip):MovieClip {
  scope = (scope == undefined) ? this : scope;
  var depth:Number = scope.getNextHighestDepth();
  var mc:MovieClip = scope.createEmptyMovieClip("mc_" + depth, depth);
  mc.beginFill(color);
  mc.lineTo(0, height);
  mc.lineTo(width, height);
  mc.lineTo(width, 0);
  mc.lineTo(0, 0);
  return mc;
}
```

## Voir également

`_alpha (propriété MovieClip._alpha)`

# alphaOffset (propriété ColorTransform.alphaOffset)

```
public alphaOffset : Number
```

Un nombre, compris entre -255 et 255, qui est ajouté à la valeur du canal de transparence alpha après avoir été multiplié par la valeur `alphaMultiplier`.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

## Exemple

L'exemple suivant crée l'objet ColorTransform `colorTrans` et ajuste sa valeur `alphaOffset` de 0 à -128.

```
import flash.geom.ColorTransform;
import flash.geom.Transform;

var colorTrans:ColorTransform = new ColorTransform();
trace(colorTrans.alphaOffset); // 0

colorTrans.alphaOffset = -128;
trace(colorTrans.alphaOffset); // -128

var rect:MovieClip = createRectangle(20, 80, 0x000000);
var trans:Transform = new Transform(rect);
trans.colorTransform = colorTrans;

function createRectangle(width:Number, height:Number, color:Number,
  scope:MovieClip):MovieClip {
  scope = (scope == undefined) ? this : scope;
  var depth:Number = scope.getNextHighestDepth();
  var mc:MovieClip = scope.createEmptyMovieClip("mc_" + depth, depth);
  mc.beginFill(color);
  mc.lineTo(0, height);
  mc.lineTo(width, height);
  mc.lineTo(width, 0);
  mc.lineTo(0, 0);
  return mc;
}
```

# blueMultiplier (propriété ColorTransform.blueMultiplier)

```
public blueMultiplier : Number
```

Une valeur décimale multipliée par la valeur du canal de bleu.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

### Exemple

L'exemple suivant crée l'objet ColorTransform `colorTrans` et ajuste sa valeur `blueMultiplier` de 1 à 0,5.

```
import flash.geom.ColorTransform;
import flash.geom.Transform;

var colorTrans:ColorTransform = new ColorTransform();
trace(colorTrans.blueMultiplier); // 1

colorTrans.blueMultiplier = .5;
trace(colorTrans.blueMultiplier); // .5

var rect:MovieClip = createRectangle(20, 80, 0x0000FF);
var trans:Transform = new Transform(rect);
trans.colorTransform = colorTrans;

function createRectangle(width:Number, height:Number, color:Number,
  scope:MovieClip):MovieClip {
  scope = (scope == undefined) ? this : scope;
  var depth:Number = scope.getNextHighestDepth();
  var mc:MovieClip = scope.createEmptyMovieClip("mc_" + depth, depth);
  mc.beginFill(color);
  mc.lineTo(0, height);
  mc.lineTo(width, height);
  mc.lineTo(width, 0);
  mc.lineTo(0, 0);
  return mc;
}
```

## blueOffset (propriété ColorTransform.blueOffset)

```
public blueOffset : Number
```

Un nombre, compris entre -255 et 255, qui est ajouté à la valeur du canal de bleu après avoir été multiplié par la valeur `blueMultiplier`.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

### Exemple

L'exemple suivant crée l'objet ColorTransform `colorTrans` et ajuste sa valeur `blueOffset` de 0 à 255.

```
import flash.geom.ColorTransform;
import flash.geom.Transform;

var colorTrans:ColorTransform = new ColorTransform();
trace(colorTrans.blueOffset); // 0
```

```
colorTrans.blueOffset = 255;
trace(colorTrans.blueOffset); // 255

var rect:MovieClip = createRectangle(20, 80, 0x000000);
var trans:Transform = new Transform(rect);
trans.colorTransform = colorTrans;

function createRectangle(width:Number, height:Number, color:Number,
  scope:MovieClip):MovieClip {
  scope = (scope == undefined) ? this : scope;
  var depth:Number = scope.getNextHighestDepth();
  var mc:MovieClip = scope.createEmptyMovieClip("mc_" + depth, depth);
  mc.beginFill(color);
  mc.lineTo(0, height);
  mc.lineTo(width, height);
  mc.lineTo(width, 0);
  mc.lineTo(0, 0);
  return mc;
}
```

# constructeur ColorTransform

```
public ColorTransform([redMultiplier:Number], [greenMultiplier:Number],
  [blueMultiplier:Number], [alphaMultiplier:Number], [redOffset:Number],
  [greenOffset:Number], [blueOffset:Number], [alphaOffset:Number])
```

Crée un objet ColorTransform pour un objet d'affichage avec les paramètres RVB et alpha spécifiés.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

## Paramètres

*redMultiplier*:Number [facultatif] - Valeur du multiplicateur de rouge, comprise entre 0 et 1. La valeur par défaut est 1.

*greenMultiplier*:Number [facultatif] - Valeur du multiplicateur de vert, comprise entre 0 et 1. La valeur par défaut est 1.

*blueMultiplier*:Number [facultatif] - Valeur du multiplicateur de bleu, comprise entre 0 et 1. La valeur par défaut est 1.

*alphaMultiplier*:Number [facultatif] - Valeur du multiplicateur de transparence alpha, comprise entre 0 et 1. La valeur par défaut est 1.

*redOffset*:Number [facultatif] - Décalage de la valeur du canal de couleur rouge (-255 à 255). La valeur par défaut est 0.

*greenOffset*:Number [facultatif] - Décalage de la valeur du canal de couleur vert (-255 à 255). La valeur par défaut est 0.

*blueOffset*:Number [facultatif] - Décalage de la valeur du canal de couleur bleu (-255 à 255). La valeur par défaut est 0.

*alphaOffset*:Number [facultatif] - Décalage de la valeur du canal de transparence alpha (-255 à 255). La valeur par défaut est 0.

### Exemple

L'exemple suivant crée un objet ColorTransform intitulé `greenTransform` :

```
var greenTransform:flash.geom.ColorTransform = new
  flash.geom.ColorTransform(0.5, 1.0, 0.5, 0.5, 10, 10, 10, 0);
```

L'exemple suivant crée l'objet ColorTransform intitulé `colorTrans_1` possédant les valeurs de constructeur par défaut. Le fait que `colorTrans_1` et `colorTrans_2` possèdent les mêmes valeurs est la preuve que les valeurs de constructeur par défaut sont utilisées.

```
import flash.geom.ColorTransform;

var colorTrans_1:ColorTransform = new ColorTransform(1, 1, 1, 1, 0, 0, 0,
  0);
trace(colorTrans_1);
//(redMultiplier=1, greenMultiplier=1, blueMultiplier=1, alphaMultiplier=1,
  redOffset=0, greenOffset=0, blueOffset=0, alphaOffset=0)

var colorTrans_2:ColorTransform = new ColorTransform();
trace(colorTrans_2);
//(redMultiplier=1, greenMultiplier=1, blueMultiplier=1, alphaMultiplier=1,
  redOffset=0, greenOffset=0, blueOffset=0, alphaOffset=0)
```

## concat (méthode ColorTransform.concat)

```
public concat(second:ColorTransform) : Void
```

Applique au clip une deuxième transformation additive de couleur. Le deuxième ensemble de paramètres de transformation est appliqué aux couleurs du clip une fois la première transformation terminée.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

### Paramètres

*second*:flash.geom.ColorTransform - Deuxième objet ColorTransform devant être combiné avec l'objet ColorTransform actuel.

## Exemple

L'exemple suivant concatène l'objet ColorTransform `colorTrans_2` à `colorTrans_1`, ce qui donne un décalage complet de rouge combiné avec un multiplicateur alpha de 0,5.

```
import flash.geom.ColorTransform;
import flash.geom.Transform;

var colorTrans_1:ColorTransform = new ColorTransform(1, 1, 1, 1, 255, 0, 0,
  0);
trace(colorTrans_1);
// (redMultiplier=1, greenMultiplier=1, blueMultiplier=1,
  alphaMultiplier=1, redOffset=255, greenOffset=0, blueOffset=0,
  alphaOffset=0)

var colorTrans_2:ColorTransform = new ColorTransform(1, 1, 1, .5, 0, 0, 0,
  0);
trace(colorTrans_2);
// (redMultiplier=1, greenMultiplier=1, blueMultiplier=1,
  alphaMultiplier=0.5, redOffset=0, greenOffset=0, blueOffset=0,
  alphaOffset=0)

colorTrans_1.concat(colorTrans_2);
trace(colorTrans_1);
// (redMultiplier=1, greenMultiplier=1, blueMultiplier=1,
  alphaMultiplier=0.5, redOffset=255, greenOffset=0, blueOffset=0,
  alphaOffset=0)

var rect:MovieClip = createRectangle(20, 80, 0x000000);
var trans:Transform = new Transform(rect);
trans.colorTransform = colorTrans_1;

function createRectangle(width:Number, height:Number, color:Number,
  scope:MovieClip):MovieClip {
  scope = (scope == undefined) ? this : scope;
  var depth:Number = scope.getNextHighestDepth();
  var mc:MovieClip = scope.createEmptyMovieClip("mc_" + depth, depth);
  mc.beginFill(color);
  mc.lineTo(0, height);
  mc.lineTo(width, height);
  mc.lineTo(width, 0);
  mc.lineTo(0, 0);
  return mc;
}
```

# greenMultiplier (propriété ColorTransform.greenMultiplier)

```
public greenMultiplier : Number
```

Une valeur décimale multipliée par la valeur du canal de vert.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

## Exemple

L'exemple suivant crée l'objet ColorTransform `colorTrans` et ajuste sa valeur `greenMultiplier` de 1 à 0,5.

```
import flash.geom.ColorTransform;
import flash.geom.Transform;

var colorTrans:ColorTransform = new ColorTransform();
trace(colorTrans.greenMultiplier); // 1

colorTrans.greenMultiplier = .5;
trace(colorTrans.greenMultiplier); // .5

var rect:MovieClip = createRectangle(20, 80, 0x00FF00);
var trans:Transform = new Transform(rect);
trans.colorTransform = colorTrans;

function createRectangle(width:Number, height:Number, color:Number,
  scope:MovieClip):MovieClip {
  scope = (scope == undefined) ? this : scope;
  var depth:Number = scope.getNextHighestDepth();
  var mc:MovieClip = scope.createEmptyMovieClip("mc_" + depth, depth);
  mc.beginFill(color);
  mc.lineTo(0, height);
  mc.lineTo(width, height);
  mc.lineTo(width, 0);
  mc.lineTo(0, 0);
  return mc;
}
```

# greenOffset (propriété ColorTransform.greenOffset)

```
public greenOffset : Number
```

Un nombre, compris entre -255 et 255, qui est ajouté à la valeur du canal de vert après avoir été multiplié par la valeur `greenMultiplier`.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

### Exemple

L'exemple suivant crée l'objet ColorTransform `colorTrans` et ajuste sa valeur `greenOffset` de 0 à 255.

```
import flash.geom.ColorTransform;
import flash.geom.Transform;

var colorTrans:ColorTransform = new ColorTransform();
trace(colorTrans.greenOffset); // 0

colorTrans.greenOffset = 255;
trace(colorTrans.greenOffset); // 255

var rect:MovieClip = createRectangle(20, 80, 0x000000);
var trans:Transform = new Transform(rect);
trans.colorTransform = colorTrans;

function createRectangle(width:Number, height:Number, color:Number,
  scope:MovieClip):MovieClip {
  scope = (scope == undefined) ? this : scope;
  var depth:Number = scope.getNextHighestDepth();
  var mc:MovieClip = scope.createEmptyMovieClip("mc_" + depth, depth);
  mc.beginFill(color);
  mc.lineTo(0, height);
  mc.lineTo(width, height);
  mc.lineTo(width, 0);
  mc.lineTo(0, 0);
  return mc;
}
```

## redMultiplier (propriété ColorTransform.redMultiplier)

```
public redMultiplier : Number
```

Une valeur décimale multipliée par la valeur du canal de rouge.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

### Exemple

L'exemple suivant crée l'objet ColorTransform `colorTrans` et ajuste sa valeur `redMultiplier` de 1 à 0,5.

```
import flash.geom.ColorTransform;
import flash.geom.Transform;

var colorTrans:ColorTransform = new ColorTransform();
trace(colorTrans.redMultiplier); // 1

colorTrans.redMultiplier = .5;
```

```
trace(colorTrans.redMultiplier); // .5

var rect:MovieClip = createRectangle(20, 80, 0xFF0000);
var trans:Transform = new Transform(rect);
trans.colorTransform = colorTrans;

function createRectangle(width:Number, height:Number, color:Number,
  scope:MovieClip):MovieClip {
  scope = (scope == undefined) ? this : scope;
  var depth:Number = scope.getNextHighestDepth();
  var mc:MovieClip = scope.createEmptyMovieClip("mc_" + depth, depth);
  mc.beginFill(color);
  mc.lineTo(0, height);
  mc.lineTo(width, height);
  mc.lineTo(width, 0);
  mc.lineTo(0, 0);
  return mc;
}
```

# redOffset (propriété ColorTransform.redOffset)

```
public redOffset : Number
```

Un nombre, compris entre -255 et 255, qui est ajouté à la valeur du canal de rouge après avoir été multiplié par la valeur `redMultiplier`.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

## Exemple

L'exemple suivant crée l'objet ColorTransform `colorTrans` et ajuste sa valeur `redOffset` de 0 à 255.

```
import flash.geom.ColorTransform;
import flash.geom.Transform;

var colorTrans:ColorTransform = new ColorTransform();
trace(colorTrans.redOffset); // 0

colorTrans.redOffset = 255;
trace(colorTrans.redOffset); // 255

var rect:MovieClip = createRectangle(20, 80, 0x000000);
var trans:Transform = new Transform(rect);
trans.colorTransform = colorTrans;

function createRectangle(width:Number, height:Number, color:Number,
  scope:MovieClip):MovieClip {
  scope = (scope == undefined) ? this : scope;
  var depth:Number = scope.getNextHighestDepth();
```

```
  var mc:MovieClip = scope.createEmptyMovieClip("mc_" + depth, depth);
  mc.beginFill(color);
  mc.lineTo(0, height);
  mc.lineTo(width, height);
  mc.lineTo(width, 0);
  mc.lineTo(0, 0);
  return mc;
}
```

# rgb (propriété ColorTransform.rgb)

```
public rgb : Number
```

La valeur de couleur RVB d'un objet ColorTransform.

Quand vous définissez cette propriété, vous obtenez par conséquent un changement des trois valeurs de couleur (`redOffset`, `greenOffset`, et `blueOffset`) et une définition des trois valeurs de multiplicateurs de couleurs (`redMultiplier`, `greenMultiplier`, et `blueMultiplier`) sur zéro. Le multiplicateur de transparence alpha et les valeurs de décalage ne changent pas.

Transmettez une valeur à cette propriété au format : 0x*RRGGBB*. Les valeurs *RR*, *GG* et *BB* se composent chacune de deux chiffres hexadécimaux qui spécifient le décalage de chaque composant de couleur. La valeur 0x indique au compilateur ActionScript que le nombre est une valeur hexadécimale.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

## Exemple

L'exemple suivant crée l'objet ColorTransform `colorTrans` et ajuste sa valeur `rgb` à 0xFF0000.

```
import flash.geom.ColorTransform;
import flash.geom.Transform;

var colorTrans:ColorTransform = new ColorTransform();
trace(colorTrans.rgb); // 0

colorTrans.rgb = 0xFF0000;
trace(colorTrans.rgb); // 16711680
trace("0x" + colorTrans.rgb.toString(16)); // 0xff0000

var rect:MovieClip = createRectangle(20, 80, 0x000000);
var trans:Transform = new Transform(rect);
trans.colorTransform = colorTrans;

function createRectangle(width:Number, height:Number, color:Number,
  scope:MovieClip):MovieClip {
```

```
  scope = (scope == undefined) ? this : scope;
  var depth:Number = scope.getNextHighestDepth();
  var mc:MovieClip = scope.createEmptyMovieClip("mc_" + depth, depth);
  mc.beginFill(color);
  mc.lineTo(0, height);
  mc.lineTo(width, height);
  mc.lineTo(width, 0);
  mc.lineTo(0, 0);
  return mc;
}
```

### Voir également

setRGB (méthode Color.setRGB), getRGB (méthode Color.getRGB)

## toString (méthode ColorTransform.toString)

```
public toString() : String
```

Formate et renvoie une chaîne qui décrit l'ensemble des propriétés de l'objet ColorTransform.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

### Valeur renvoyée

`String` - Chaîne répertoriant toutes les propriétés de l'objet ColorTransform.

### Exemple

L'exemple suivant crée l'objet ColorTransform `colorTrans` et appelle sa méthode `toSting()`. Cette méthode donne une chaîne au format suivant : (redMultiplier=RM, greenMultiplier=GM, blueMultiplier=BM, alphaMultiplier=AM, redOffset=RO, greenOffset=GO, blueOffset=BO, alphaOffset=AO).

```
import flash.geom.ColorTransform;

var colorTrans:ColorTransform = new ColorTransform(1, 2, 3, 4, -255, -128,
  128, 255);
trace(colorTrans.toString());
// (redMultiplier=1, greenMultiplier=2, blueMultiplier=3,
  alphaMultiplier=4, redOffset=-255, greenOffset=-128, blueOffset=128,
  alphaOffset=255)
```

# ContextMenu

```
Object
  |
  +-ContextMenu
```

```
public dynamic class ContextMenu
extends Object
```

La classe ContextMenu permet de contrôler à l'exécution les éléments du menu contextuel de Flash Player qui s'affichent lorsqu'un utilisateur clique avec le bouton droit (Windows) ou en appuyant sur la touche Contrôle (Macintosh) dans Flash Player. Vous pouvez utiliser les méthodes et les propriétés de la classe ContextMenu pour ajouter des éléments de menu personnalisés, contrôler l'affichage des éléments du menu contextuel intégrés (par exemple, Zoom avant et Imprimer) ou créer des copies de menus.

Vous pouvez lier un objet ContextMenu à un bouton, clip ou objet de champ de texte spécifique, ou à un niveau d'animation entier. Pour ce faire, il vous suffit d'utiliser la propriété menu des classes Button, MovieClip ou TextField. Pour plus d'informations sur la propriété `menu`, consultez `Button.menu`, `MovieClip.menu` et `TextField.menu`.

Pour ajouter de nouveaux articles à un objet `ContextMenu`, vous créez un objet `ContextMenuItem` et vous ajoutez ensuite cet objet au tableau `ContextMenu.customItems`. Pour plus d'informations sur la création d'éléments de menu contextuel, consultez l'entrée de la classe ContextMenuItem.

Flash Player est doté de trois types de menus contextuels : le menu standard (qui s'affiche lorsque vous cliquez avec le bouton droit de la souris dans Flash Player), le menu Edition (qui s'affiche lorsque vous cliquez avec le bouton droit de la souris sur un champ de texte sélectionnable ou modifiable), et un menu d'erreur (qui s'affiche lorsque le chargement d'un fichier SWF dans Flash Player a échoué.) Seuls les menus standard et Edition peuvent être modifiés avec la classe ContextMenu.

Les éléments de menu personnalisés s'affichent toujours dans la partie supérieure du menu contextuel de Flash Player, au-dessus des éléments de menu intégrés visibles ; une barre de séparation permet de faire la distinction entre les éléments de menu intégrés et personnalisés. Vous ne pouvez pas ajouter plus de 15 éléments personnalisés à un menu contextuel. Vous ne pouvez pas supprimer l'élément de menu Paramètres dans le menu contextuel. L'élément de menu Paramètres est requis dans Flash de manière à ce que les utilisateurs puissent accéder aux paramètres relatifs à la confidentialité et à l'enregistrement des données sur leurs ordinateurs. De même, vous ne pouvez pas supprimer l'élément de menu A propos de dans le menu contextuel : celui-ci est requis de manière à ce que les utilisateurs puissent connaître la version de Flash Player qu'ils utilisent.

Vous devez utiliser le constructeur new `ContextMenu()` pour créer un objet ContextMenu avant d'appeler ses méthodes.

**Disponibilité :** ActionScript 1.0 ; Flash Player 7

## Voir également

`ContextMenuItem`, `menu (propriété Button.menu)`, `menu (propriété MovieClip.menu)`, `menu (propriété TextField.menu)`

## Résumé des propriétés

| Modificateurs | Propriété | Description |
|---|---|---|
| | `builtInItems:Object` | Un objet ayant les propriétés booléennes suivantes : `zoom`, `quality`, `play`, `loop`, `rewind`, `forward_back`, et `print`. |
| | `customItems:Array` | Un tableau d'objets ContextMenuItem. |

*Propriétés héritées de la classe Object*

`constructeur (propriété Object.constructor)`, `__proto__ (Object.__proto__ property)`, `prototype (Object.prototype, propriété)`, `__resolve (Object.__resolve, propriété)`

## Résumé des événements

| Evénement | Description |
|---|---|
| `onSelect = function(item:Object, item_menu:Object) {}` | Appelé lorsqu'un utilisateur invoque le menu contextuel de Flash Player, mais avant que le menu ne s'affiche. |

## Résumé des constructeurs

| Signature | Description |
|---|---|
| `ContextMenu([callbackFunction:Function])` | Crée un nouvel objet ContextMenu. |

### Résumé de la méthode

| Modificateurs | Signature | Description |
|---|---|---|
| | `copy() : ContextMenu` | Crée une copie de l'objet ContextMenu spécifié. |
| | `hideBuiltInItems() : Void` | Masque tous les éléments de menu intégrés (à l'exception de Paramètres) dans l'objet ContextMenu spécifié. |

*Méthodes héritées de la classe Object*

`addProperty (méthode Object.addProperty), hasOwnProperty (méthode Object.hasOwnProperty), isPropertyEnumerable (méthode Object.isPropertyEnumerable), isPrototypeOf (méthode Object.isPrototypeOf), registerClass (méthode Object.registerClass), toString (méthode Object.toString), unwatch (méthode Object.unwatch), valueOf (méthode Object.valueOf), watch (méthode Object.watch)`

## builtInItems (propriété ContextMenu.builtInItems)

```
public builtInItems : Object
```

Un objet ayant les propriétés booléennes suivantes : `zoom`, `quality`, `play`, `loop`, `rewind`, `forward_back`, et `print`. La définition de ces variables sur `false` supprime les éléments de menu correspondants sur l'objet ContextMenu spécifié. Ces propriétés sont énumérables et définies sur `true` par défaut.

**Disponibilité :** ActionScript 1.0 ; Flash Player 7

### Exemple

Dans cet exemple, les éléments de menu intégrés Qualité et Imprimer sont désactivés pour l'objet ContextMenu intitulé `my_cm`, associé au scénario actuel du fichier SWF.

```
var my_cm:ContextMenu = new ContextMenu ();
my_cm.builtInItems.quality=false;
my_cm.builtInItems.print=false;
this.menu = my_cm;
```

*Remarque :* Vous ne pouvez pas désactiver les éléments de menu Paramètres ou A propos de dans le menu contextuel.

Dans l'exemple suivant, une boucle `for..in` énumère tous les noms et toutes les valeurs des éléments de menu intégrés de l'objet ContextMenu, `my_cm`.

```
var my_cm:ContextMenu = new ContextMenu();
for(eachProp in my_cm.builtInItems) {
  var propName = eachProp;
  var propValue = my_cm.builtInItems[propName];
```

```
  trace(propName + ": " + propValue);
}
```

## constructeur ContextMenu

```
public ContextMenu([callbackFunction:Function])
```

Crée un nouvel objet ContextMenu. Vous pouvez également spécifier un identifiant pour un gestionnaire d'événements lorsque vous créez l'objet. La fonction spécifiée est appelée lorsque l'utilisateur invoque le menu contextuel, mais *avant que* le menu ne s'affiche. Cette fonction s'avère utile pour personnaliser le contenu du menu en fonction de l'état de l'application ou du type d'objet (clip, champ de texte ou bouton) ou du scénario sur lequel l'utilisateur a cliqué avec le bouton droit de la souris ou avec la touche Contrôle. (Voir `ContextMenu.onSelect` pour un exemple de création de gestionnaire d'événements.)

**Disponibilité :** ActionScript 1.0 ; Flash Player 7

### Paramètres

*`callbackFunction`*`:Function` [facultatif] - Référence à une fonction qui est appelée lorsque l'utilisateur clique avec le bouton droit de la souris ou maintient la touche Contrôle enfoncée, avant que le menu s'affiche.

### Exemple

L'exemple suivant masque tous les objets intégrés dans le menu contextuel. (Toutefois, les éléments Paramètres et A propos de s'affichent toujours car ils ne peuvent pas être désactivés.)

```
var newMenu:ContextMenu = new ContextMenu();
newMenu.hideBuiltInItems();
this.menu = newMenu;
```

Dans cet exemple, le gestionnaire d'événements spécifié, `menuHandler`, active ou désactive un élément de menu personnalisé (à l'aide du tableau `ContextMenu.customItems`) selon la valeur d'une variable booléenne intitulée `showItem`. Si la valeur est `false`, l'élément de menu personnalisé est désactivé ; dans le cas contraire, il est activé.

```
var showItem = true; // Change this to false to remove
var my_cm:ContextMenu = new ContextMenu(menuHandler);
my_cm.customItems.push(new ContextMenuItem("Hello", itemHandler));
function menuHandler(obj, menuObj) {
  if (showItem == false) {
  menuObj.customItems[0].enabled = false;
  } else {
  menuObj.customItems[0].enabled = true;
  }
}
function itemHandler(obj, item) {
```

```
  //...put code here...
  trace("selected!");
}
this.menu = my_cm;
```

Lorsque l'utilisateur clique avec le bouton droit de la souris ou maintient la touche Contrôle enfoncée dans la scène, le menu personnalisé s'affiche.

### Voir également

menu (propriété Button.menu), onSelect (gestionnaire ContextMenu.onSelect), customItems (propriété ContextMenu.customItems), hideBuiltInItems (méthode ContextMenu.hideBuiltInItems), menu (propriété MovieClip.menu), menu (propriété TextField.menu)

## copy (méthode ContextMenu.copy)

```
public copy() : ContextMenu
```

Crée une copie de l'objet ContextMenu spécifié. La copie hérite de toutes les propriétés de l'objet de menu original.

**Disponibilité :** ActionScript 1.0 ; Flash Player 7

### Valeur renvoyée

`ContextMenu` - Objet ContextMenu.

### Exemple

Cet exemple crée une copie de l'objet ContextMenu intitulé `my_cm`, dont les éléments de menu intégrés sont masqués, puis ajoute un élément de menu avec le texte « Enregistrer... ». Il crée ensuite une copie de `my_cm` et l'affecte à la variable `clone_cm` qui hérite de toutes les propriétés du menu d'origine.

```
var my_cm:ContextMenu = new ContextMenu();
my_cm.hideBuiltInItems();
var menuItem_cmi:ContextMenuItem = new ContextMenuItem("Save...",
  saveHandler);
my_cm.customItems.push(menuItem_cmi);
function saveHandler(obj, menuItem) {
  // saveDocument();
  // custom function (not shown)
  trace("something");
}
clone_cm = my_cm.copy();
this.menu = my_cm;
for (var i in clone_cm.customItems) {
  trace("clone_cm-> "+clone_cm.customItems[i].caption);
```

```
}
for (var i in my_cm.customItems) {
  trace("my_cm-> "+my_cm.customItems[i].caption);
}
```

# customItems (propriété ContextMenu.customItems)

```
public customItems : Array
```

Un tableau d'objets ContextMenuItem. Chaque objet du tableau représente un élément de menu contextuel que vous avez défini. Utilisez cette propriété pour ajouter, supprimer ou modifier ces éléments de menu personnalisés.

Pour ajouter de nouveaux éléments de menu, commencez par créer un nouvel objet ContextMenuItem, puis ajoutez-le dans le tableau *menu_mc* `.customItems` (par exemple, via `Array.push()`). Pour plus d'informations sur la création de nouveaux éléments de menu, consultez l'entrée de la classe ContextMenuItem.

**Disponibilité :** ActionScript 1.0 ; Flash Player 7

## Exemple

L'exemple suivant crée un nouvel élément de menu personnalisé intitulé `menuItem_cmi` ayant pour titre « Send e-mail » et un gestionnaire de rappel intitulé `emailHandler`. Le nouvel élément de menu est ensuite ajouté à l'objet ContextMenu `my_cm` à l'aide du tableau `customItems`. Enfin, le nouveau menu est associé à un clip intitulé `email_mc`. Pour faire fonctionner cet exemple, créez une occurrence de clip sur votre scène et utilisez l'inspecteur des propriétés pour nommer l'occurrence `email_mc`. En mode Tester l'animation, le nouvel élément de menu contextuel s'affiche si vous ouvrez le menu contextuel lorsque votre pointeur survole le clip `email_mc`.

```
var my_cm:ContextMenu = new ContextMenu();
var menuItem_cmi:ContextMenuItem = new ContextMenuItem("Send e-mail",
  emailHandler);
my_cm.customItems.push(menuItem_cmi);
email_mc.menu = my_cm;
function emailHandler() {
  trace("sending email");
}
```

## Voir également

menu (propriété Button.menu), menu (propriété MovieClip.menu), menu (propriété TextField.menu), push (méthode Array.push)

# hideBuiltInItems (méthode ContextMenu.hideBuiltInItems)

```
public hideBuiltInItems() : Void
```

Masque tous les éléments de menu intégrés (à l'exception de Paramètres) dans l'objet ContextMenu spécifié. Si le débogueur de Flash Player s'exécute, l'élément de menu Débogage apparaît, bien qu'il soit grisé pour les fichiers SWF sur lesquels le débogage à distance n'est pas activé.

Cette méthode masque uniquement les éléments de menu qui s'affichent dans le menu contextuel standard ; elle n'affecte pas les éléments qui s'affichent dans les menus Edition ou d'erreur.

Cette méthode fonctionne en définissant tous les membres booléens de *my_cm* `.builtInItems` sur `false`. Vous pouvez rendre visible un élément intégré de façon sélective en définissant son membre correspondant dans *my_cm* `.builtInItems` sur `true` (comme illustré dans l'exemple suivant).

**Disponibilité :** ActionScript 1.0 ; Flash Player 7

## Exemple

L'exemple suivant crée un nouvel objet ContextMenu, intitulé `my_cm` dont les éléments de menu intégrés sont masqués, à l'exception de Imprimer. L'objet Menu est associé au scénario actuel.

```
var my_cm:ContextMenu = new ContextMenu();
my_cm.hideBuiltInItems();
my_cm.builtInItems.print = true;
this.menu = my_cm;
```

# onSelect (gestionnaire ContextMenu.onSelect)

onSelect = function(item:Object, item_menu:Object) {}

Appelé lorsqu'un utilisateur invoque le menu contextuel de Flash Player, mais avant que le menu ne s'affiche. Ce gestionnaire d'événements permet de personnaliser le contenu du menu contextuel en fonction de l'état de l'application actuelle.

Il est également possible de spécifier le gestionnaire de rappel d'un objet ContextMenu lors de la construction d'un nouvel objet ContextMenu. Pour plus d'informations, consultez l'entrée `onSelect` de ContextMenuItem.

**Disponibilité :** ActionScript 1.0 ; Flash Player 7

### Paramètres

*item*:Object - Référence à l'objet (clip, bouton ou champ de texte sélectionnable) qui était sous le pointeur de la souris quand le menu contextuel Flash Player a été invoqué et dont la propriété menu est définie sur un objet ContextMenu valide.

*item_menu*:Object - Référence à l'objet ContextMenu affecté à la propriété menu de object.

### Exemple

L'exemple suivant détermine le type d'objet à l'origine de l'appel du menu contextuel.

```
my_cm:ContextMenu = new ContextMenu();
function menuHandler(obj:Object, menu:ContextMenu) {
  if(obj instanceof MovieClip) {
    trace("Movie clip: " + obj);
  }
  if(obj instanceof TextField) {
    trace("Text field: " + obj);
  }
  if(obj instanceof Button) {
    trace("Button: " + obj);
  }
}
my_cm.onSelect = menuHandler;
my_mc.menu = my_cm;
my_btn.menu = my_cm;
```

# ContextMenuItem

```
Object
  |
  +-ContextMenuItem
```

public dynamic class *ContextMenuItem*
extends Object

La classe ContextMenuItem vous permet de créer des éléments de menu personnalisés afin qu'ils s'affichent dans le menu contextuel de Flash Player. Chaque objet ContextMenuItem est doté d'une légende (texte) qui s'affiche dans le menu contextuel, et d'un gestionnaire de rappel (une fonction) qui est appelé lorsque l'élément de menu est sélectionné. Pour ajouter un nouvel élément de menu contextuel dans un menu contextuel, il vous suffit de l'ajouter dans le tableau customItems d'un objet ContextMenu.

Vous pouvez activer ou désactiver des éléments de menu spécifiques, rendre des éléments visibles ou invisibles, ou encore modifier la légende ou le gestionnaire de rappel associé(e) à un élément de menu.

Les éléments de menu personnalisés s'affichent dans la partie supérieure du menu contextuel, au-dessus des éléments intégrés. Une barre de séparation sépare toujours les éléments de menu personnalisés des éléments intégrés. Vous ne pouvez pas ajouter plus de 15 éléments personnalisés à un menu contextuel. Chaque élément doit contenir au moins un caractère visible ; les caractères de contrôle, de nouvelle ligne et autres espaces blancs sont ignorés. Aucun élément ne peut contenir plus de 100 caractères. Les éléments identiques à un élément de menu intégré, ou à un autre élément personnalisé, sont ignorés, indépendamment du fait que l'élément correspondant soit visible ou non. Les éléments de menu sont comparés sans respecter la casse, la ponctuation ou les espaces blancs.

Les mots suivants ne peuvent pas apparaître dans un élément personnalisé : *Macromedia*, *Flash Player* et *Paramètres*.

**Disponibilité :** ActionScript 1.0 ; Flash Player 7

## Résumé des propriétés

| Modificateurs | Propriété | Description |
|---|---|---|
| | `caption:String` | Chaîne spécifiant la légende (texte) de l'élément de menu qui s'affiche dans le menu contextuel. |
| | `enabled:Boolean` | Valeur booléenne indiquant si l'élément de menu spécifié est activé ou désactivé. |
| | `separatorBefore:Boolean` | Valeur booléenne indiquant si une barre de séparation doit apparaître au-dessus de l'élément de menu spécifié. |
| | `visible:Boolean` | Valeur booléenne indiquant si l'élément de menu spécifié est visible lorsque le menu contextuel de Flash Player s'affiche. |

*Propriétés héritées de la classe Object*

`constructeur (propriété Object.constructor), __proto__ (Object.__proto__ property), prototype (Object.prototype, propriété), __resolve (Object.__resolve, propriété)`

### Résumé des événements

| Evénement | Description |
|---|---|
| `onSelect = function(obj:Object, menuItem:Object) {}` | Appelé lorsque l'élément de menu spécifié est sélectionné dans le menu contextuel de Flash Player. |

### Résumé des constructeurs

| Signature | Description |
|---|---|
| `ContextMenuItem(caption:String, callbackFunction:Function, [separatorBefore:Boolean], [enabled:Boolean], [visible:Boolean])` | Crée un nouvel objet ContextMenuItem pouvant être ajouté dans le tableau `ContextMenu.customItems`. |

### Résumé de la méthode

| Modificateurs | Signature | Description |
|---|---|---|
| | `copy() : ContextMenuItem` | Crée et renvoie une copie de l'objet ContextMenuItem spécifié. |

*Méthodes héritées de la classe Object*

`addProperty (méthode Object.addProperty), hasOwnProperty (méthode Object.hasOwnProperty), isPropertyEnumerable (méthode Object.isPropertyEnumerable), isPrototypeOf (méthode Object.isPrototypeOf), registerClass (méthode Object.registerClass), toString (méthode Object.toString), unwatch (méthode Object.unwatch), valueOf (méthode Object.valueOf), watch (méthode Object.watch)`

## caption (propriété ContextMenuItem.caption)

```
public caption : String
```

Chaîne spécifiant la légende (texte) de l'élément de menu qui s'affiche dans le menu contextuel.

**Disponibilité :** ActionScript 1.0 ; Flash Player 7

### Exemple

L'exemple suivant affiche la légende de l'élément de menu sélectionné (Pause jeu) dans le panneau de sortie :

```
var my_cm:ContextMenu = new ContextMenu();
var menuItem_cmi:ContextMenuItem = new ContextMenuItem("Pause Game",
  onPause);
my_cm.customItems.push(menuItem_cmi);
function onPause(obj, menuItem) {
  trace("You chose: " + menuItem.caption);
}
this.menu = my_cm;
```

## constructeur ContextMenuItem

```
public ContextMenuItem(caption:String, callbackFunction:Function,
  [separatorBefore:Boolean], [enabled:Boolean], [visible:Boolean])
```

Crée un nouvel objet ContextMenuItem pouvant être ajouté dans le tableau `ContextMenu.customItems`.

**Disponibilité :** ActionScript 1.0 ; Flash Player 7

### Paramètres

*`caption`*`:String` - Chaîne spécifiant le texte associé à l'élément de menu.

*`callbackFunction`*`:Function` - Fonction que vous définissez, appelée lorsque l'élément de menu est sélectionné.

*`separatorBefore`*`:Boolean` [facultatif] - Valeur booléenne indiquant si une barre de séparation doit apparaître au-dessus de l'élément de menu du menu contextuel. La valeur par défaut est `false`.

*`enabled`*`:Boolean` [facultatif] - Valeur booléenne indiquant si l'élément de menu est activé ou désactivé dans le menu contextuel. La valeur par défaut est `true`.

*`visible`*`:Boolean` [facultatif] - Valeur booléenne indiquant si l'élément de menu est visible ou invisible. La valeur par défaut est `true`.

### Exemple

Cet exemple ajoute les éléments de menu Démarrer et Arrêter, séparés par une barre, à l'objet ContextMenu `my_cm`. La fonction `startHandler()` est appelée quand Démarrer est sélectionné dans le menu contextuel ; `stopHandler()` est appelé quand Arrêter est sélectionné. L'objet ContextMenu est appliqué au scénario actuel.

```
var my_cm:ContextMenu = new ContextMenu();
my_cm.customItems.push(new ContextMenuItem("Start", startHandler));
```

```
my_cm.customItems.push(new ContextMenuItem("Stop", stopHandler, true));
function stopHandler(obj, item) {
  trace("Stopping...");
}
function startHandler(obj, item) {
  trace("Starting...");
}
this.menu = my_cm;
```

# copy (méthode ContextMenuItem.copy)

```
public copy() : ContextMenuItem
```

Crée et renvoie une copie de l'objet ContextMenuItem spécifié. La copie inclut toutes les propriétés de l'objet original.

**Disponibilité :** ActionScript 1.0 ; Flash Player 7

## Valeur renvoyée

`ContextMenuItem` - Objet ContextMenuItem.

## Exemple

Cet exemple crée un nouvel objet ContextMenuItem intitulé `original_cmi` incluant la légende Pause et un gestionnaire de rappel défini sur la fonction `onPause`. L'exemple crée ensuite une copie de l'objet ContextMenuItem et l'affecte à la variable `copy_cmi`.

```
var original_cmi:ContextMenuItem = new ContextMenuItem("Pause", onPause);
function onPause(obj:Object, menu:ContextMenu) {
  trace("pause me");
}

var copy_cmi:ContextMenuItem = original_cmi.copy();

var my_cm:ContextMenu = new ContextMenu();
my_cm.customItems.push(original_cmi);
my_cm.customItems.push(copy_cmi);

my_mc.menu = my_cm;
```

# enabled (propriété ContextMenuItem.enabled)

```
public enabled : Boolean
```

Valeur booléenne indiquant si l'élément de menu spécifié est activé ou désactivé. Par défaut, cette propriété est définie sur `true`.

**Disponibilité :** ActionScript 1.0 ; Flash Player 7

### Exemple

L'exemple suivant crée deux nouveaux éléments de menu contextuel : Start et Stop. Lorsque l'utilisateur sélectionne Start, le nombre de millisecondes écoulées depuis l'ouverture du fichier SWF est analysé. L'élément Start est ensuite désactivé dans le menu. Lorsque Stop est sélectionné, le nombre de millisecondes écoulées depuis l'ouverture du fichier SWF est analysé. L'élément de menu Start est de nouveau activé et l'élément de menu Stop est désactivé.

```
var my_cm:ContextMenu = new ContextMenu();
var startMenuItem:ContextMenuItem = new ContextMenuItem("Start",
  startHandler);
startMenuItem.enabled = true;
my_cm.customItems.push(startMenuItem);
var stopMenuItem:ContextMenuItem = new ContextMenuItem("Stop", stopHandler,
  true);
stopMenuItem.enabled = false;
my_cm.customItems.push(stopMenuItem);
function stopHandler(obj, item) {
  trace("Stopping... "+getTimer()+"ms");
  startMenuItem.enabled = true;
  stopMenuItem.enabled = false;
}
function startHandler(obj, item) {
  trace("Starting... "+getTimer()+"ms");
  startMenuItem.enabled = false;
  stopMenuItem.enabled = true;
}
this.menu = my_cm;
```

## onSelect (gestionnaire ContextMenuItem.onSelect)

onSelect = function(obj:Object, menuItem:Object) {}

Appelé lorsque l'élément de menu spécifié est sélectionné dans le menu contextuel de Flash Player. Le gestionnaire de rappel spécifié reçoit deux paramètres : `obj`, une référence à l'objet affiché sous la souris lorsque l'utilisateur a appelé le menu contextuel de Flash Player, et `item`, une référence à l'objet ContextMenuItem qui représente l'élément de menu sélectionné.

**Disponibilité :** ActionScript 1.0 ; Flash Player 7

### Paramètres

*`obj`*`:Object` - Référence à l'objet (clip, scénario, bouton ou champ de texte sélectionnable) sur lequel l'utilisateur fait un clic droit ou Contrôle+clic.

*`menuItem`*`:Object` - Référence à l'objet ContextMenuItem sélectionné.

### Exemple

L'exemple suivant détermine le type d'objet à l'origine de l'appel du menu contextuel.

```
var my_cmi:ContextMenu = new ContextMenu();
var start_cmi:ContextMenuItem = new ContextMenuItem("Start");
start_cmi.onSelect = function(obj, item) {
  trace("You chose: "+item.caption);
};
my_cmi.customItems.push(start_cmi);
my_cmi.customItems.push(new ContextMenuItem("Stop", stopHandler, true));
function stopHandler(obj, item) {
  trace("Stopping...");
}
this.menu = my_cmi;
```

### Voir également

onSelect (gestionnaire ContextMenu.onSelect)

## separatorBefore (propriété ContextMenuItem.separatorBefore)

```
public separatorBefore : Boolean
```

Valeur booléenne indiquant si une barre de séparation doit apparaître au-dessus de l'élément de menu spécifié. Par défaut, cette propriété est définie sur `false`.

*Remarque :* Une barre de séparation apparaît toujours entre les éléments de menu personnalisés et les éléments de menu intégrés.

**Disponibilité :** ActionScript 1.0 ; Flash Player 7

### Exemple

Cet exemple crée trois éléments de menu, intitulés Open, Save et Print. Une barre de séparation sépare les éléments Save et Print. Les éléments de menu sont ensuite ajoutés au tableau `customItems` de l'objet ContextMenu. Enfin, le menu est associé au scénario actuel du fichier SWF.

```
var my_cm:ContextMenu = new ContextMenu();
var open_cmi:ContextMenuItem = new ContextMenuItem("Open", itemHandler);
var save_cmi:ContextMenuItem = new ContextMenuItem("Save", itemHandler);
var print_cmi:ContextMenuItem = new ContextMenuItem("Print", itemHandler);
print_cmi.separatorBefore = true;
my_cm.customItems.push(open_cmi, save_cmi, print_cmi);
function itemHandler(obj, menuItem) {
  trace("You chose: " + menuItem.caption);
};
this.menu = my_cm;
```

### Voir également

onSelect (gestionnaire ContextMenu.onSelect)

## visible (propriété ContextMenuItem.visible)

```
public visible : Boolean
```

Valeur booléenne indiquant si l'élément de menu spécifié est visible lorsque le menu contextuel de Flash Player s'affiche. Par défaut, cette propriété est définie sur `true`.

**Disponibilité :** ActionScript 1.0 ; Flash Player 7

### Exemple

L'exemple suivant crée deux nouveaux éléments de menu contextuel : Start et Stop. Lorsque l'utilisateur sélectionne Start, le nombre de millisecondes écoulées depuis l'ouverture du fichier SWF s'affiche. L'élément Start est ensuite rendu invisible dans le menu. Lorsque Stop est sélectionné, le nombre de millisecondes écoulées depuis l'ouverture du fichier SWF s'affiche. L'élément de menu Start devient visible et l'élément de menu Stop est rendu invisible.

```
var my_cm:ContextMenu = new ContextMenu();
var startMenuItem:ContextMenuItem = new ContextMenuItem("Start",
  startHandler);
startMenuItem.visible = true;
my_cm.customItems.push(startMenuItem);
var stopMenuItem:ContextMenuItem = new ContextMenuItem("Stop", stopHandler,
  true);
stopMenuItem.visible = false;
my_cm.customItems.push(stopMenuItem);
function stopHandler(obj, item) {
  trace("Stopping... "+getTimer()+"ms");
  startMenuItem.visible = true;
  stopMenuItem.visible = false;
}
function startHandler(obj, item) {
  trace("Starting... "+getTimer()+"ms");
  startMenuItem.visible = false;
  stopMenuItem.visible = true;
}
this.menu = my_cm;
```

# ConvolutionFilter (flash.filters.ConvolutionFilter)

```
Object
  |
  +-flash.filters.BitmapFilter
    |
    +-flash.filters.ConvolutionFilter
```

`public class` ***ConvolutionFilter***
`extends BitmapFilter`

La classe ConvolutionFilter applique un effet de filtre de convolution de matrice. Une convolution associe les pixels de l'image d'entrée aux pixels environnants pour produire une image. Les convolutions permettent d'effectuer de nombreuses opérations de traitement de l'image, notamment la définition du flou, la détection de contour, l'accentuation, l'estampage et le biseautage. Vous pouvez appliquer cet effet aux bitmaps et aux occurrences MovieClip.

L'utilisation de filtres dépend de l'objet auquel vous appliquez le filtre.

- Pour appliquer les filtres lors de l'exécution du clip, utilisez la propriété `filters`. Lorsque vous définissez la propriété `filters` d'un objet, celui-ci n'est pas modifié. En outre, vous pouvez l'annuler en supprimant la propriété `filters`.
- Pour appliquer des filtres aux occurrences BitmapData, utilisez la méthode `BitmapData.applyFilter()` . L'appel `applyFilter()` sur un objet BitmapData, utilise l'objet BitmapData d'origine ainsi que l'objet filtre pour générer une image filtrée.

Vous pouvez également appliquer des effets de filtre aux images et aux données vidéo pendant la programmation. Pour plus d'informations, consultez la documentation relative à la programmation.

Si vous appliquez un filtre à un clip ou à un bouton, la propriété `cacheAsBitmap` du clip ou du bouton est définie sur `true`. Si vous supprimez tous les filtres, la valeur d'origine de `cacheAsBitmap` est restaurée.

Une convolution de matrice s'articule autour d'une matrice *n* par *m*, qui décrit la façon dont une valeur de pixels donnée dans l'image d'entrée est associée aux valeurs des pixels environnants pour obtenir une nouvelle valeur de pixels. Chaque pixel obtenu est déterminé par l'application de la matrice au pixel source correspondant et à ses pixels environnants.

Pour une convolution de matrice 3 par 3, la formule suivante est utilisée pour chaque canal de couleur indépendant :

```
dst (x, y) = ((src (x-1, y-1) * a0 + src(x, y-1) * a1....
src(x, y+1) * a7 + src (x+1,y+1) * a8) / divisor) + bias
```

Lorsque le processeur utilisé pour l'exécution des spécifications de filtre est doté d'extensions Streaming SIMD (SSE), certaines d'entre-elles s'exécutent plus rapidement.

- Le filtre doit être un filtre 3 par 3.
- Tous les termes du filtre doivent être des entiers compris entre -127 et +127.
- La valeur absolue correspondant à la somme de tous les termes du filtre ne doit pas être supérieure à 127.
- Si un terme de filtre est négatif, le diviseur doit être compris entre 2,00001 et 256.
- Si tous les termes de filtre sont positifs, le diviseur doit être compris entre 1,1 et 256.
- L'écart doit être un entier.

Un filtre ne peut s'appliquer si l'image résultante dépasse 2 880 pixels en largeur ou en hauteur. Par exemple, si vous faites un zoom avant sur un grand clip auquel le filtre est appliqué, le filtre est désactivé si l'image résultante atteint la limite de 2 880 pixels.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

## Voir également

`applyFilter (méthode BitmapData.applyFilter)`, `filters (propriété MovieClip.filters)`, `cacheAsBitmap (propriété MovieClip.cacheAsBitmap)`

## Résumé des propriétés

| Modificateurs | Propriété | Description |
|---|---|---|
| | `alpha:Number` | La valeur de transparence alpha de la couleur de substitution. |
| | `bias:Number` | Écart à ajouter au résultat de la transformation de matrice. |
| | `clamp:Boolean` | Indique si l'image doit être corrigée. |
| | `color:Number` | La couleur hexadécimale à substituer aux pixels provenant de l'image source. |
| | `divisor:Number` | Le diviseur utilisé pendant la transformation de matrice. |
| | `matrix:Array` | Le tableau de valeurs utilisé pour la transformation de matrice ; renvoie une copie. |
| | `matrixX:Number` | La dimension *x* de la matrice (le nombre de colonnes de la matrice). |
| | `matrixY:Number` | La dimension *y* de la matrice (le nombre de lignes de la matrice). |
| | `preserveAlpha:Boolean` | Indique ce à quoi la convolution s'applique. |

*Propriétés héritées de la classe Object*

`constructeur (propriété Object.constructor), __proto__ (Object.__proto__ property), prototype (Object.prototype, propriété), __resolve (Object.__resolve, propriété)`

## Résumé des constructeurs

| Signature | Description |
|---|---|
| `ConvolutionFilter(matrixX:Number, matrixY:Number, matrix:Array, [divisor:Number], [bias:Number], [preserveAlpha:Boolean], [clamp:Boolean], [color:Number], [alpha:Number])` | Initialise une occurrence ConvolutionFilter avec les paramètres spécifiés. |

### Résumé de la méthode

| Modificateurs | Signature | Description |
|---|---|---|
| | `clone() : ConvolutionFilter` | Renvoie une copie de cet objet filtre. |

*Méthodes héritées de la classe BitmapFilter*

`clone (méthode BitmapFilter.clone )`

*Méthodes héritées de la classe Object*

`addProperty (méthode Object.addProperty), hasOwnProperty (méthode Object.hasOwnProperty), isPropertyEnumerable (méthode Object.isPropertyEnumerable), isPrototypeOf (méthode Object.isPrototypeOf), registerClass (méthode Object.registerClass), toString (méthode Object.toString), unwatch (méthode Object.unwatch), valueOf (méthode Object.valueOf), watch (méthode Object.watch)`

## alpha (propriété ConvolutionFilter.alpha)

`public alpha : Number`

La valeur de transparence alpha de la couleur de substitution. Les valeurs valides sont comprises entre 0 et 1,0. La valeur par défaut est zéro. Par exemple, 0,25 définit une valeur de transparence de 25 %. La valeur par défaut est 1,0.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

### Exemple

L'exemple suivant change la propriété `alpha` de `filter` de sa valeur par défaut (1) à 0,35.

```
import flash.filters.ConvolutionFilter;
import flash.display.BitmapData;
import flash.geom.Rectangle;
import flash.geom.Point;

var alpha:Number = .35;
var filter:ConvolutionFilter = new ConvolutionFilter(3, 3, [1, 1, 1, 1, 1,
  1, 1, 1, 1], 9, 0, true, false, 0x0000FF, alpha);

var myBitmapData:BitmapData = new BitmapData(100, 80, true, 0xCCFF0000);

var mc:MovieClip = this.createEmptyMovieClip("mc",
  this.getNextHighestDepth());
mc.attachBitmap(myBitmapData, this.getNextHighestDepth());
```

```
myBitmapData.noise(128, 0, 255, 1 | 2 | 4 | 8, false);

mc.onPress = function() {
  myBitmapData.applyFilter(myBitmapData, new Rectangle(0, 0, 98, 78), new
  Point(2, 2), filter);
}
```

# bias (propriété ConvolutionFilter.bias)

```
public bias : Number
```

Écart à ajouter au résultat de la transformation de matrice. La valeur par défaut est 0.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

## Exemple

L'exemple suivant change la propriété `bias` de `filter` de sa valeur par défaut (0) à 50.

```
import flash.filters.ConvolutionFilter;
import flash.display.BitmapData;

var bias:Number = 50;
var filter:ConvolutionFilter = new ConvolutionFilter(3, 3, [1, 1, 1, 1, 1,
  1, 1, 1, 1], 9, bias);

var myBitmapData:BitmapData = new BitmapData(100, 80, false, 0x00FF0000);

var mc:MovieClip = this.createEmptyMovieClip("mc",
  this.getNextHighestDepth());
mc.attachBitmap(myBitmapData, this.getNextHighestDepth());
myBitmapData.noise(128);

mc.onPress = function() {
  myBitmapData.applyFilter(myBitmapData, myBitmapData.rectangle, new
  Point(0, 0), filter);
}
```

# clamp (propriété ConvolutionFilter.clamp)

```
public clamp : Boolean
```

Indique si l'image doit être corrigée. Pour les pixels provenant de l'image source, la valeur `true` indique que l'image d'entrée est agrandie autant que nécessaire au niveau de ses bordures en dupliquant les valeurs de couleur sur le bord spécifié de l'image d'entrée. Une valeur `false` indique qu'il faut utiliser une autre couleur, comme spécifié dans les propriétés `color` et `alpha`. La valeur par défaut est `true`.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

### Exemple

L'exemple suivant change la propriété `clamp` de `filter` de sa valeur par défaut `true` à `false`.

```
import flash.filters.ConvolutionFilter;
import flash.display.BitmapData;
import flash.geom.Rectangle;
import flash.geom.Point;

var clamp:Boolean = false;
var filter:ConvolutionFilter = new ConvolutionFilter(3, 3, [1, 1, 1, 1, 1,
  1, 1, 1, 1], 9, 0, true, clamp, 0x00FF00, 1);

var myBitmapData:BitmapData = new BitmapData(100, 80, true, 0xCCFF0000);

var mc:MovieClip = this.createEmptyMovieClip("mc",
  this.getNextHighestDepth());
mc.attachBitmap(myBitmapData, this.getNextHighestDepth());
myBitmapData.noise(128, 0, 255, 1 | 2 | 4 | 8, false);

mc.onPress = function() {
  myBitmapData.applyFilter(myBitmapData, new Rectangle(0, 0, 98, 78), new
  Point(2, -2), filter);
}
```

## clone (méthode ConvolutionFilter.clone)

```
public clone() : ConvolutionFilter
```

Renvoie une copie de cet objet filtre.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

### Valeur renvoyée

`flash.filters.ConvolutionFilter` - Nouvelle occurrence ConvolutionFilter dont les propriétés sont toutes identiques à celles de l'occurrence d'origine.

### Exemple

L'exemple suivant crée trois objets ConvolutionFilter et les compare : `filter_1` est créé en utilisant le constructeur `ConvolutionFilter` ; `filter_2` est créé en le définissant comme égal à `filter_1`; et `clonedFilter` est créé en clonant `filter_1`. Veuillez noter que `filter_2` est considéré comme égal à `filter_1`, `clonedFilter` ne l'est pas, même s'il contient les mêmes valeurs que `filter_1`.

```
import flash.filters.ConvolutionFilter;

var filter_1:ConvolutionFilter = new ConvolutionFilter(3, 3, [1, 1, 1, 1, 1,
  1, 1, 1, 1], 9);
```

```
var filter_2:ConvolutionFilter = filter_1;
var clonedFilter:ConvolutionFilter = filter_1.clone();

trace(filter_1 == filter_2); // true
trace(filter_1 == clonedFilter); // false

for(var i in filter_1) {
  trace(">> " + i + ": " + filter_1[i]);
  // >> clone: [type Function]
  // >> alpha: 0
  // >> color: 0
  // >> clamp: true
  // >> preserveAlpha: true
  // >> bias: 0
  // >> divisor: 9
  // >> matrix: 1,1,1,1,1,1,1,1,1
  // >> matrixY: 3
  // >> matrixX: 3
}

for(var i in clonedFilter) {
  trace(">> " + i + ": " + clonedFilter[i]);
  // >> clone: [type Function]
  // >> alpha: 0
  // >> color: 0
  // >> clamp: true
  // >> preserveAlpha: true
  // >> bias: 0
  // >> divisor: 9
  // >> matrix: 1,1,1,1,1,1,1,1,1
  // >> matrixY: 3
  // >> matrixX: 3
}
```

Pour démontrer davantage les relations entre `filter_1`, `filter_2`, et `clonedFilter`, l'exemple suivant modifie la propriété `bias` de `filter_1`. La modification de `bias` démontre que la méthode `clone()` crée une nouvelle occurrence basée sur les valeurs de `filter_1` au lieu de faire référence à ces valeurs.

```
import flash.filters.ConvolutionFilter;

var filter_1:ConvolutionFilter = new ConvolutionFilter(3, 3, [1, 1, 1, 1, 1,
  1, 1, 1, 1], 9);
var filter_2:ConvolutionFilter = filter_1;
var clonedFilter:ConvolutionFilter = filter_1.clone();
trace(filter_1.bias); // 0
trace(filter_2.bias); // 0
trace(clonedFilter.bias); // 0

filter_1.bias = 20;
```

```
trace(filter_1.bias); // 20
trace(filter_2.bias); // 20
trace(clonedFilter.bias); // 0
```

# color (propriété ConvolutionFilter.color)

```
public color : Number
```

La couleur hexadécimale à substituer aux pixels provenant de l'image source. C'est une valeur RVB sans composant alpha. La valeur par défaut est 0.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

## Exemple

L'exemple suivant change la propriété `color` de `filter` de sa valeur par défaut (0) à 0xFF0000.

```
import flash.filters.ConvolutionFilter;
import flash.display.BitmapData;
import flash.geom.Rectangle;
import flash.geom.Point;

var color:Number = 0x0000FF;
var filter:ConvolutionFilter = new ConvolutionFilter(3, 3, [1, 1, 1, 1, 1,
  1, 1, 1, 1], 9, 0, true, false, color, 1);

var myBitmapData:BitmapData = new BitmapData(100, 80, true, 0xCCFF0000);

var mc:MovieClip = this.createEmptyMovieClip("mc",
  this.getNextHighestDepth());
mc.attachBitmap(myBitmapData, this.getNextHighestDepth());
myBitmapData.noise(128, 0, 255, 1 | 2 | 4 | 8, false);

var height:Number = 100;
var width:Number = 80;
mc.onPress = function() {
  height -= 2;
  width -= 2;
  myBitmapData.applyFilter(myBitmapData, new Rectangle(0, 0, height,
  width), new Point(2, 2), filter);
}
```

# constructeur ConvolutionFilter()

```
public ConvolutionFilter(matrixX:Number, matrixY:Number, matrix:Array,
  [divisor:Number], [bias:Number], [preserveAlpha:Boolean],
  [clamp:Boolean], [color:Number], [alpha:Number])
```

Initialise une occurrence ConvolutionFilter avec les paramètres spécifiés.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

## Paramètres

*matrixX*:Number - Dimension *x* de la matrice (le nombre de colonnes de la matrice). La valeur par défaut est 0.

*matrixY*:Number - Dimension *y* de la matrice (le nombre de lignes de la matrice). La valeur par défaut est 0.

*matrix*:Array - Tableau de valeurs utilisé pour la transformation de matrice ; renvoie une copie. Le nombre d'éléments contenus dans le tableau doit être égal à `matrixX*matrixY`.

*divisor*:Number [facultatif] - Diviseur utilisé pendant la transformation de matrice. La valeur par défaut est 1. Le diviseur correspondant à la somme de toutes les valeurs de matrice égalise l'intensité de couleurs globale du résultat. La valeur 0 est ignorée ; elle est remplacée par la valeur par défaut.

*bias*:Number [facultatif] - Ecart à ajouter au résultat de la transformation de matrice. La valeur par défaut est 0.

*preserveAlpha*:Boolean [facultatif] - La valeur `false` indique que la convolution s'applique à tous les canaux, y compris le canal alpha. La valeur `true` indique que la convolution s'applique uniquement aux canaux de couleur. La valeur par défaut est `true`.

*clamp*:Boolean [facultatif] - Pour les pixels provenant de l'image source, la valeur `true` indique que l'image d'entrée est agrandie autant que nécessaire au niveau de ses bordures en dupliquant les valeurs de couleur sur le bord spécifié de l'image d'entrée. Une valeur `false` indique qu'il faut utiliser une autre couleur, comme spécifié dans les propriétés `color` et `alpha`. La valeur par défaut est `true`.

*color*:Number [facultatif] - Couleur hexadécimale à substituer aux pixels provenant de l'image source.

*alpha*:Number [facultatif] - Valeur Alpha de la couleur de substitution.

### Exemple

Le code suivant crée un filtre de convolution 3 x 3 avec un diviseur de 9. Le filtre produit une image floue :

```
var myArray:Array = [1, 1, 1, 1, 1, 1, 1, 1, 1];
var myFilter:ConvolutionFilter = new flash.filters.ConvolutionFilter (3,
3, myArray, 9);
```

L'exemple suivant crée un objet ConvolutionFilter ayant les quatre paramètres requis `matrixX`, `matrixY`, `matrix`, et `divisor`.

```
import flash.filters.ConvolutionFilter;
import flash.display.BitmapData;

var matrixX:Number = 3;
var matrixY:Number = 3;
var matrix:Array = [1, 1, 1, 1, 1, 1, 1, 1, 1];
var divisor:Number = 9;

var filter:ConvolutionFilter = new ConvolutionFilter(matrixX, matrixY,
  matrix, divisor);

var myBitmapData:BitmapData = new BitmapData(100, 80, false, 0x00FF0000);

var mc:MovieClip = this.createEmptyMovieClip("mc",
  this.getNextHighestDepth());
mc.attachBitmap(myBitmapData, this.getNextHighestDepth());
myBitmapData.noise(128);

mc.onPress = function() {
  myBitmapData.applyFilter(myBitmapData, myBitmapData.rectangle, new
  Point(0, 0), filter);
}
```

## divisor (propriété ConvolutionFilter.divisor)

```
public divisor : Number
```

Le diviseur utilisé pendant la transformation de matrice. La valeur par défaut est 1. Le diviseur correspondant à la somme de toutes les valeurs de matrice égalise l'intensité de couleurs globale du résultat. La valeur 0 est ignorée ; elle est remplacée par la valeur par défaut.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

### Exemple

L'exemple suivant change la propriété `divisor` de `filter` à 6.

```
import flash.filters.ConvolutionFilter;
import flash.display.BitmapData;
```

```
var filter:ConvolutionFilter = new ConvolutionFilter(3, 3, [1, 1, 1, 1, 1,
  1, 1, 1, 1], 9);

var myBitmapData:BitmapData = new BitmapData(100, 80, false, 0x00FF0000);

var mc:MovieClip = this.createEmptyMovieClip("mc",
  this.getNextHighestDepth());
mc.attachBitmap(myBitmapData, this.getNextHighestDepth());
myBitmapData.noise(128);

mc.onPress = function() {
  var newDivisor:Number = 6;
  filter.divisor = newDivisor;
    myBitmapData.applyFilter(myBitmapData, myBitmapData.rectangle, new
  Point(0, 0), filter);
}
```

# matrix (propriété ConvolutionFilter.matrix)

```
public matrix : Array
```

Le tableau de valeurs utilisé pour la transformation de matrice ; renvoie une copie. Le nombre d'éléments contenus dans le tableau doit être égal à `matrixX*matrixY`.

La propriété `matrix` ne peut pas être changée directement en modifiant les valeurs (par exemple, `myFilter.matrix[2] = 1;`). Vous devez plutôt, comme l'illustre l'exemple suivant, donner une référence au tableau, effectuer le changement de la référence, et restaurer la valeur en utilisant `filter.matrix = newMatrix;`.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

## Exemple

L'exemple suivant change la propriété `matrix` de `filter` d'une valeur qui estompe une bitmap en une valeur qui lui donne plus de contraste.

```
import flash.filters.ConvolutionFilter;
import flash.display.BitmapData;

var filter:ConvolutionFilter = new ConvolutionFilter(3, 3, [1, 1, 1, 1, 1,
  1, 1, 1, 1], 9);

var myBitmapData:BitmapData = new BitmapData(100, 80, false, 0x00FF0000);

var mc:MovieClip = this.createEmptyMovieClip("mc",
  this.getNextHighestDepth());
mc.attachBitmap(myBitmapData, this.getNextHighestDepth());
myBitmapData.noise(128);
```

```
mc.onPress = function() {
  var newMatrix:Array = [0, -1, 0, -1, 8, -1, 0, -1, 0];
  filter.matrix = newMatrix;
  myBitmapData.applyFilter(myBitmapData, myBitmapData.rectangle, new
  Point(0, 0), filter);
}
```

# matrixX (propriété ConvolutionFilter.matrixX)

```
public matrixX : Number
```

La dimension *x* de la matrice (le nombre de colonnes de la matrice). La valeur par défaut est 0.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

## Exemple

L'exemple suivant affiche la propriété `matrixX` de `filter`.

```
import flash.filters.ConvolutionFilter;

var filter:ConvolutionFilter = new ConvolutionFilter(2, 3, [1, 0, 0, 1, 0,
  0], 6);
trace(filter.matrixX); // 2
```

# matrixY (propriété ConvolutionFilter.matrixY)

```
public matrixY : Number
```

La dimension *y* de la matrice (le nombre de lignes de la matrice). La valeur par défaut est 0.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

## Exemple

L'exemple suivant affiche la propriété `matrixY` de `filter`.

```
import flash.filters.ConvolutionFilter;

var filter:ConvolutionFilter = new ConvolutionFilter(2, 3, [1, 0, 0, 1, 0,
  0], 6);
trace(filter.matrixY); // 3
```

## preserveAlpha (propriété ConvolutionFilter.preserveAlpha)

```
public preserveAlpha : Boolean
```

Indique ce à quoi la convolution s'applique. La valeur `false` indique que la convolution s'applique à tous les canaux, y compris le canal alpha. La valeur `true` indique que la convolution s'applique uniquement aux canaux de couleur. La valeur par défaut est `true`.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

### Exemple

L'exemple suivant change la propriété `preserveAlpha` de `filter` de sa valeur par défaut `true` à `false`.

```
import flash.filters.ConvolutionFilter;
import flash.display.BitmapData;

var preserveAlpha:Boolean = false;
var filter:ConvolutionFilter = new ConvolutionFilter(3, 3, [1, 1, 1, 1, 1,
  1, 1, 1, 1], 9, 0, preserveAlpha);

var myBitmapData:BitmapData = new BitmapData(100, 80, true, 0xCCFF0000);

var mc:MovieClip = this.createEmptyMovieClip("mc",
  this.getNextHighestDepth());
mc.attachBitmap(myBitmapData, this.getNextHighestDepth());
myBitmapData.noise(128, 0, 255, 1 | 2 | 4 | 8, false);

mc.onPress = function() {
  myBitmapData.applyFilter(myBitmapData, myBitmapData.rectangle, new
  Point(0, 0), filter);
}
```

# CustomActions

```
Object
  |
  +-CustomActions
```

```
public class CustomActions
extends Object
```

Les méthodes de la classe CustomActions permettent de lire un fichier SWF dans l'outil de programmation Flash pour gérer des actions personnalisées enregistrées via l'outil de programmation. Un fichier SWF peut installer et désinstaller des actions personnalisées, récupérer la définition XML d'une action personnalisée, et récupérer la liste des actions personnalisées enregistrées.

Vous pouvez utiliser ces méthodes pour créer des fichiers SWF qui sont des extensions de l'outil de programmation Flash. Cette extension, pourrait par exemple utiliser le protocole d'application Flash pour naviguer dans un référentiel UDDI et télécharger des services Web dans la boîte à outils Actions.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

## Résumé des propriétés

*Propriétés héritées de la classe Object*

```
constructeur (propriété Object.constructor), __proto__ (Object.__proto__
property), prototype (Object.prototype, propriété), __resolve
(Object.__resolve, propriété)
```

## Résumé de la méthode

| Modificateurs | Signature | Description |
|---|---|---|
| `static` | `get(name:String) : String` | Lit le contenu du fichier de définition XML des actions personnalisées nommé `name`. |
| `static` | `install(name:String, data:String) : Boolean` | Installe un nouveau fichier de définition XML des actions personnalisées indiqué par le paramètre `name`. |
| `static` | `list() : Array` | Renvoie un objet Array contenant les noms de toutes les actions personnalisées enregistrées via l'outil de programmation Flash. |
| `static` | `uninstall(name:String) : Boolean` | Supprime le fichier de définition XML des actions personnalisées nommé `name`. |

*Méthodes héritées de la classe Object*

```
addProperty (méthode Object.addProperty), hasOwnProperty (méthode
Object.hasOwnProperty), isPropertyEnumerable (méthode
Object.isPropertyEnumerable), isPrototypeOf (méthode Object.isPrototypeOf),
registerClass (méthode Object.registerClass), toString (méthode
Object.toString), unwatch (méthode Object.unwatch), valueOf (méthode
Object.valueOf), watch (méthode Object.watch)
```

# get (méthode CustomActions.get)

```
public static get(name:String) : String
```

Lit le contenu du fichier de définition XML des actions personnalisées nommé `name`.

Le nom du fichier de définition doit être simple, dépourvu de l'extension de fichier .xml et des caractères de séparation de répertoire (« : », « / » ou « \ »).

Si le fichier de définition spécifié par le paramètre `name` est introuvable, une valeur `undefined` est renvoyée. Si la définition XML des actions personnalisées spécifiée par le paramètre `name` est localisée, elle est lue intégralement et renvoyée sous forme de chaîne.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

## Paramètres

*`name`*`:String` - Nom de la définition des actions personnalisées à récupérer.

## Valeur renvoyée

`String` - Si la définition XML des actions personnalisées est localisée, renvoie une chaîne ; sinon, renvoie une valeur `undefined`.

## Exemple

L'exemple suivant répertorie les actions personnalisées dans une occurrence ComboBox et obtient l'action personnalisée lorsque l'utilisateur clique sur une occurrence Button. Faites glisser une occurrence de ComboBox, Button et TextArea vers la scène. Attribuez à la ComboBox le nom d'occurrence `customActionName_cb`, à la TextArea le nom d'occurrence`customActionXml_ta` et au Button le nom d'occurrence `view_button`. Entrez le code ActionScript suivant sur l'image 1 du scénario :

```
import mx.controls.*;

var customActionName_cb:ComboBox;
var customActionXml_ta:TextArea;
var view_button:Button;

customActionName_cb.dataProvider = CustomActions.list();

customActionXml_ta.editable = false;

var viewListener:Object = new Object();
viewListener.click = function(evt:Object) {
  var caName:String = String(customActionName_cb.selectedItem);
  customActionXml_ta.text = CustomActions.get(caName);
};
view_button.addEventListener("click", viewListener);
```

# install (méthode CustomActions.install)

```
public static install(name:String, data:String) : Boolean
```

Installe un nouveau fichier de définition XML des actions personnalisées indiqué par le paramètre `name`. Le contenu du fichier est spécifié par la chaîne *`customXML`*.

Le nom du fichier de définition doit être simple, dépourvu de l'extension de fichier .xml et des caractères de séparation de répertoire (« : », « / » ou « \ »).

Si un fichier d'actions personnalisées portant le nom `name` existe déjà, il est remplacé.

Si le répertoire Configuration/ActionsPanel/CustomActions n'existe pas lorsque cette méthode est appelée, il est créé.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

## Paramètres

*`name`*`:String` - Nom de la définition des actions personnalisées à installer.

*`data`*`:String` - Texte de la définition XML à installer.

## Valeur renvoyée

`Boolean` - La valeur booléenne `false` est renvoyée si une erreur se produit au cours de l'installation ; sinon, la valeur `true` est renvoyée pour indiquer que l'action personnalisée a été installée avec succès.

## Exemple

L'exemple suivant installe des informations dans le panneau Actions à partir d'un fichier XML. Ouvrez un éditeur de texte et enregistrez un nouveau document intitulé dogclass.xml. Entrez le code suivant :

```
<?xml version="1.0"?>
<customactions>
  <actionspanel>
  <folder version="7" id="DogClass" index="true" name="Dog" tiptext="Dog
  Class">
    <string version="7" id="getFleas" name="getFleas" tiptext="gets number
  of fleas" text=".getFleas(% fleas %)" />
  </folder>
  </actionspanel>
  <colorsyntax>
    <identifier text=".getFleas" />
  </colorsyntax>
  <codehints>
    <typeinfo pattern=" _dog" object="Dog"/>
  </codehints>
</customactions>
```

Ouvrez ensuite un nouveau fichier FLA dans le même répertoire et sélectionnez l'image 1 du scénario. Entrez le code suivant dans le panneau Actions :

```
var my_xml:XML = new XML();
my_xml.ignoreWhite = true;
my_xml.onLoad = function(success:Boolean) {
  trace(success);
  CustomActions.install("dogclass", this.firstChild);
  trace(CustomActions.list());
};
my_xml.load("dogclass.xml");
```

Pointez sur Contrôle > Tester l'animation : si le code XML se charge correctement, la valeur `true` apparaît et un tableau contenant les noms de toutes les actions personnalisées enregistrées via l'outil de programmation Flash s'affiche dans le panneau de sortie. Fermez le fichier SWF et ouvrez le panneau Actions. Un nouvel élément intitulé Dog s'affiche dans la boîte à outils Actions ; ce dossier inclut également getFleas.

# list (méthode CustomActions.list)

```
public static list() : Array
```

Renvoie un objet Array contenant les noms de toutes les actions personnalisées enregistrées via l'outil de programmation Flash. Les éléments du tableau portent des noms simples, dépourvus de l'extension de fichier .xml et des caractères de séparation de répertoire (par exemple, « : », « / » ou « \ »). Si aucune action personnalisée n'est enregistrée, `list()` renvoie un tableau de longueur zéro. Si une erreur se produit, `list()` renvoie la valeur `undefined`.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

## Valeur renvoyée

`Array` - Tableau.

## Exemple

L'exemple suivant répertorie les actions personnalisées dans une occurrence ComboBox et obtient l'action personnalisée lorsque l'utilisateur clique sur une occurrence Button. Faites glisser une occurrence de ComboBox, Button et TextArea vers la scène. Attribuez à la ComboBox le nom d'occurrence `customActionName_cb`, à la TextArea le nom d'occurrence`customActionXml_ta` et au Button le nom d'occurrence `view_button`. Entrez le code ActionScript suivant sur l'image 1 du scénario :

```
import mx.controls.*;

var customActionName_cb:ComboBox;
var customActionXml_ta:TextArea;
```

```
var view_button:Button;

customActionName_cb.dataProvider = CustomActions.list();

customActionXml_ta.editable = false;

var viewListener:Object = new Object();
viewListener.click = function(evt:Object) {
  var caName:String = String(customActionName_cb.selectedItem);
  customActionXml_ta.text = CustomActions.get(caName);
};
view_button.addEventListener("click", viewListener);
```

# uninstall (méthode CustomActions.uninstall)

```
public static uninstall(name:String) : Boolean
```

Supprime le fichier de définition XML des actions personnalisées nommé `name`.

Le nom du fichier de définition doit être simple, dépourvu de l'extension de fichier .xml et des caractères de séparation de répertoire (« : », « / » ou « \ »).

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

## Paramètres

*`name`*`:String` - Nom de la définition des actions personnalisées à désinstaller.

## Valeur renvoyée

`Boolean` - Valeur booléenne `false` si aucune action personnalisée portant le nom `name` n'a été trouvée. Si les actions personnalisées ont été supprimées avec succès, une valeur `true` est renvoyée.

## Exemple

L'exemple suivant installe une nouvelle action personnalisée et affiche un tableau contenant les noms de toutes les actions personnalisées enregistrées via l'outil de programmation Flash dans le panneau de sortie. Lorsque l'utilisateur clique sur le bouton uninstall_btn, l'action personnalisée est désinstallée. Un tableau contenant les noms des actions personnalisées installées s'affiche ; dogclass doit être alors supprimé du tableau. Créez un bouton intitulé `uninstall_btn` et entrez le code ActionScript suivant dans l'image 1 du scénario :

```
var my_xml:XML = new XML();
my_xml.ignoreWhite = true;
my_xml.onLoad = function(success:Boolean) {
  trace(success);
  CustomActions.install("dogclass", this.firstChild);
  trace(CustomActions.list());
```

```
};
my_xml.load("dogclass.xml");

uninstall_btn.onRelease = function() {
  CustomActions.uninstall("dogclass");
  trace(CustomActions.list());
};
```

Pour plus d'informations sur la création de dogclass.xml, consultez `CustomActions.install()`.

### Voir également

`install (méthode CustomActions.install)`

# Date

```
Object
  |
  +-Date
```

`public class` ***Date***
`extends Object`

La classe Date vous permet de récupérer des valeurs de date et d'heure relatives à l'heure universelle (GMT, désormais appelée heure universelle ou UTC) ou au système d'exploitation sur lequel Flash Player s'exécute. Les méthodes de la classe Date ne sont pas statiques mais s'appliquent uniquement au seul objet Date spécifié lorsque la méthode est appelée. La méthode `Date.UTC()` est une exception ; il s'agit d'une méthode statique.

La classe Date gère l'heure d'été différemment, en fonction du système d'exploitation et de la version de Flash Player. Flash Player 6 et les versions ultérieures gèrent l'heure d'été sur les systèmes d'exploitation suivants comme suit :

- Windows : l'objet Date ajuste automatiquement sa sortie pour l'heure d'été. L'objet Date détecte si l'heure d'été est définie selon les paramètres régionaux actuels, et si tel est le cas, détecte la date et les heures de transition de l'heure d'été standard. Toutefois, les dates de transition actuellement en vigueur sont appliquées aux dates passées et à venir : par conséquent, le décalage de l'heure d'été peut être calculé de manière incorrecte pour les dates passées lorsque les paramètres régionaux étaient définis sur différentes dates de transition.
- Mac OS X : l'objet Date ajuste automatiquement sa sortie pour l'heure d'été. La base de données d'informations sur le fuseau horaire dans Mac OS X est utilisée pour déterminer si un décalage d'heure d'été doit être appliqué à une date ou heure actuelle ou passée.

- Mac OS 9 : le système d'exploitation fournit uniquement les informations suffisantes pour déterminer s'il convient d'appliquer un décalage d'heure d'été à la date et l'heure actuelles. En conséquence, l'objet de date suppose que le décalage d'heure d'été actuel s'applique à toutes les dates et heures passées ou à venir.

Flash Player 5 gère l'heure d'été sur les systèmes d'exploitation suivants comme suit :

- Windows : les règles en vigueur aux Etats-Unis concernant l'heure d'été sont toujours appliquées, ce qui entraîne des transitions incorrectes en Europe et dans les autres zones qui adoptent l'heure d'été, mais ayant des heures de transition différentes de celles en vigueur aux Etats-Unis. Flash détecte correctement si l'heure d'été est utilisée dans les paramètres régionaux actuels.

Pour appeler les méthodes de la classe Date, vous devez d'abord créer un objet Date à l'aide du constructeur de la classe Date, décrit plus loin dans cette section.

**Disponibilité :** ActionScript 1.0 ; Flash Player 5

## Résumé des propriétés

*Propriétés héritées de la classe Object*

```
constructeur (propriété Object.constructor), __proto__ (Object.__proto__
property), prototype (Object.prototype, propriété), __resolve
(Object.__resolve, propriété)
```

## Résumé des constructeurs

| Signature | Description |
|---|---|
| `Date([yearOrTimevalue:Number], [month:Number], [date:Number], [hour:Number], [minute:Number], [second:Number], [millisecond:Number])` | Construit un nouvel objet Date qui contient la date et l'heure spécifiées. |

Résumé de la méthode

| Modificateurs | Signature | Description |
|---|---|---|
| | `getDate() : Number` | Renvoie le jour du mois (entier de 1 à 31) de l'objet Date spécifié, conformément à l'heure locale. |
| | `getDay() : Number` | Renvoie le jour de la semaine (0 pour dimanche, 1 pour lundi, etc.) de l'objet Date spécifié conformément à l'heure locale. |
| | `getFullYear() : Number` | Renvoie l'année entière (un nombre à quatre chiffres, tel que 2000) de l'objet Date spécifié, conformément à l'heure locale. |
| | `getHours() : Number` | Renvoie l'heure (un entier de 0 à 23) de l'objet Date spécifié, conformément à l'heure locale. |
| | `getMilliseconds() : Number` | Renvoie les millisecondes (entier de 0 à 999) de l'objet Date spécifié, conformément à l'heure locale. |
| | `getMinutes() : Number` | Renvoie les minutes (entier de 0 à 59) de l'objet Date spécifié, conformément à l'heure locale. |
| | `getMonth() : Number` | Renvoie le mois (0 pour janvier, 1 pour février, etc.) de l'objet Date spécifié conformément à l'heure locale. |
| | `getSeconds() : Number` | Renvoie les secondes (entier de 0 à 59) de l'objet Date spécifié, conformément à l'heure locale. |
| | `getTime() : Number` | Renvoie le nombre de millisecondes écoulées depuis le premier janvier 1970 à minuit, heure universelle, pour l'objet Date spécifié. |
| | `getTimezoneOffset() : Number` | Renvoie la différence, en minutes, entre l'heure locale de l'ordinateur et l'heure universelle. |
| | `getUTCDate() : Number` | Renvoie le jour du mois (entier de 1 à 31) de l'objet Date spécifié, conformément à l'heure universelle. |
| | `getUTCDay() : Number` | Renvoie le jour de la semaine (0 pour dimanche, 1 pour lundi, etc.) de l'objet Date spécifié, conformément à l'heure universelle. |
| | `getUTCFullYear() : Number` | Renvoie l'année à quatre chiffres de l'objet Date spécifié, conformément à l'heure universelle. |
| | `getUTCHours() : Number` | Renvoie l'heure (entier de 0 à 23) de l'objet Date spécifié, conformément à l'heure universelle. |
| | `getUTCMilliseconds() : Number` | Renvoie les millisecondes (entier de 0 à 999) de l'objet Date spécifié, conformément à l'heure universelle. |

| Modificateurs | Signature | Description |
|---|---|---|
| | `getUTCMinutes() : Number` | Renvoie les minutes (entier de 0 à 59) de l'objet Date spécifié, conformément à l'heure universelle. |
| | `getUTCMonth() : Number` | Renvoie le mois (0 [janvier] à 11 [décembre]) de l'objet Date spécifié, conformément à l'heure universelle. |
| | `getUTCSeconds() : Number` | Renvoie les secondes (entier de 0 à 59) de l'objet Date spécifié, conformément à l'heure universelle. |
| | `getUTCYear() : Number` | Renvoie l'année de cette `Date` conformément à l'heure universelle. |
| | `getYear() : Number` | Renvoie l'année de l'objet Date spécifié, conformément à l'heure locale. |
| | `setDate(date:Number) : Number` | Définit le jour du mois de l'objet Date spécifié, conformément à l'heure locale, et renvoie la nouvelle heure en millisecondes. |
| | `setFullYear(year:Number, [month:Number], [date:Number]) : Number` | Définit l'année de l'objet Date spécifié, conformément à l'heure locale, et renvoie la nouvelle heure en millisecondes. |
| | `setHours(hour:Number) : Number` | Définit les heures de l'objet Date spécifié conformément à l'heure locale et renvoie la nouvelle heure en millisecondes. |
| | `setMilliseconds(millisecond:Number) : Number` | Définit les millisecondes de l'objet Date spécifié conformément à l'heure locale et renvoie la nouvelle heure en millisecondes. |
| | `setMinutes(minute:Number) : Number` | Définit les minutes de l'objet Date spécifié conformément à l'heure locale et renvoie la nouvelle heure en millisecondes. |
| | `setMonth(month:Number, [date:Number]) : Number` | Définit le mois de l'objet Date spécifié conformément à l'heure locale et renvoie la nouvelle heure en millisecondes. |
| | `setSeconds(second:Number) : Number` | Définit les secondes de l'objet Date spécifié conformément à l'heure locale et renvoie la nouvelle heure en millisecondes. |
| | `setTime(millisecond:Number) : Number` | Définit la date de l'objet Date spécifié en millisecondes écoulées depuis le premier janvier 1970 à minuit et renvoie la nouvelle heure en millisecondes. |

| Modificateurs | Signature | Description |
|---|---|---|
| | `setUTCDate(date:Number) : Number` | Définit la date de l'objet Date spécifié conformément à l'heure universelle et renvoie la nouvelle heure en millisecondes. |
| | `setUTCFullYear(year:Number, [month:Number], [date:Number]) : Number` | Définit l'année de l'objet Date spécifié (*my_date*) conformément à l'heure universelle et renvoie la nouvelle heure en millisecondes. |
| | `setUTCHours(hour:Number, [minute:Number], [second:Number], [millisecond:Number]) : Number` | Définit l'heure de l'objet Date spécifié conformément à l'heure universelle et renvoie la nouvelle heure en millisecondes. |
| | `setUTCMilliseconds(millisecond:Number) : Number` | Définit les millisecondes de l'objet Date spécifié conformément à l'heure universelle et renvoie la nouvelle heure en millisecondes. |
| | `setUTCMinutes(minute:Number, [second:Number], [millisecond:Number]) : Number` | Définit les minutes de l'objet Date spécifié conformément à l'heure universelle et renvoie la nouvelle heure en millisecondes. |
| | `setUTCMonth(month:Number, [date:Number]) : Number` | Définit le mois, et éventuellement le jour, de l'objet Date spécifié conformément à l'heure universelle et renvoie la nouvelle heure en millisecondes. |
| | `setUTCSeconds(second:Number, [millisecond:Number]) : Number` | Définit les secondes de l'objet Date spécifié conformément à l'heure universelle et renvoie la nouvelle heure en millisecondes. |
| | `setYear(year:Number) : Number` | Définit l'année de l'objet Date spécifié conformément à l'heure locale et renvoie la nouvelle heure en millisecondes. |
| | `toString() : String` | Renvoie une valeur de chaîne pour l'objet de date spécifié dans un format lisible. |

| Modificateurs | Signature | Description |
|---|---|---|
| `static` | `UTC(year:Number, month:Number, [date:Number], [hour:Number], [minute:Number], [second:Number], [millisecond:Number] ) : Number` | Renvoie le nombre de millisecondes écoulées entre le premier janvier 1970 à minuit, heure universelle, et l'heure spécifiée dans les paramètres. |
| | `valueOf() : Number` | Renvoie le nombre de millisecondes écoulées depuis le premier janvier 1970 à minuit, heure universelle, pour cette `Date`. |

*Méthodes héritées de la classe Object*

`addProperty (méthode Object.addProperty), hasOwnProperty (méthode Object.hasOwnProperty), isPropertyEnumerable (méthode Object.isPropertyEnumerable), isPrototypeOf (méthode Object.isPrototypeOf), registerClass (méthode Object.registerClass), toString (méthode Object.toString), unwatch (méthode Object.unwatch), valueOf (méthode Object.valueOf), watch (méthode Object.watch)`

# constructeur Date

```
public Date([yearOrTimevalue:Number], [month:Number], [date:Number],
  [hour:Number], [minute:Number], [second:Number], [millisecond:Number])
```

Construit un nouvel objet Date qui contient la date et l'heure spécifiées.

Le constructeur `Date()` accepte jusqu'à sept paramètres (year, month, ..., millisecond) pour spécifier une date et une heure en millisecondes. Vous pouvez également transmettre une valeur unique au constructeur `Date()`, indiquant la valeur de l'heure en fonction du nombre de millisecondes écoulées depuis le premier janvier 1970 à minuit GMT. Vous pouvez encore omettre les paramètres ; dans ce cas, la date et l'heure actuelles sont affectées à l'objet `Date()`.

Par exemple, ce code illustre différentes manières de créer un objet Date :

```
var d1:Date = new Date();
var d3:Date = new Date(2000, 0, 1);
var d4:Date = new Date(65, 2, 6, 9, 30, 15, 0);
var d5:Date = new Date(-14159025000);
```

Dans la première ligne de code, un objet Date est défini sur l'heure à laquelle l'instruction d'affectation est exécutée.

Dans la deuxième ligne, un objet Date incluant les paramètres year, month et date est créé, soit le premier janvier 2000 à minuit GMT.

Dans la troisième ligne, un objet Date incluant les paramètres year, month et date est créé, soit le six mars 1965 à 09:30:15 GMT (+ 0 milliseconde). Remarque : étant donné que le paramètre year est spécifié en tant que nombre entier à deux chiffres, il est interprété comme 1965.

Dans la quatrième ligne, un seul paramètre est transmis : il s'agit d'une valeur de temps représentant le nombre de millisecondes écoulées avant ou après le premier janvier 1970 à minuit GMT ; étant donné que la valeur est négative, elle représente une heure *avant* le premier janvier 1970 à minuit GMT, soit le 21 juillet 1969 à 02:56:15 GMT.

**Disponibilité :** ActionScript 1.0 ; Flash Player 5

## Paramètres

*yearOrTimevalue*:Number [facultatif] - Si d'autres paramètres sont spécifiés, ce nombre représente une année (telle que 1965) ; sinon, il représente une valeur de temps. Si le nombre représente une année, une valeur comprise entre 0 et 99 renvoie à une année comprise entre 1900 et 1999 ; sinon les quatre chiffres de l'année doivent être spécifiés. Si le nombre représente une valeur de temps (aucun paramètre supplémentaire n'est spécifié), il s'agit du nombre de millisecondes écoulées avant ou après le premier janvier 1970 à minuit GMT ; une valeur négative représente une heure *avant* le premier janvier 1970 à minuit GMT ; une valeur positive représente une heure postérieure à cette date.

*month*:Number [facultatif] - Entier compris entre 0 (janvier) et 11 (décembre).

*date*:Number [facultatif] - Entier compris entre 1 et 31.

*hour*:Number [facultatif] - Entier compris entre 0 (minuit) et 23 (23h00).

*minute*:Number [facultatif] - Entier compris entre 0 et 59.

*second*:Number [facultatif] - Entier compris entre 0 et 59.

*millisecond*:Number - Entier compris entre 0 et 999 millisecondes.

## Exemple

L'exemple suivant récupère la date et l'heure actuelles :

```
var now_date:Date = new Date();
```

L'exemple suivant crée un nouvel objet Date pour le jour de naissance de Marie, le 12 août 1974 (étant donné que le paramètre month est basé sur zéro, cet exemple utilise le chiffre 7 pour le mois, et non le chiffre 8) :

```
var maryBirthday:Date = new Date (74, 7, 12);
```

L'exemple suivant crée un nouvel objet Date et concatène les valeurs renvoyées de `Date.getMonth()`, `Date.getDate()` et `Date.getFullYear()` :

```
var today_date:Date = new Date();
var date_str:String = ((today_date.getMonth()+1)+"/
  "+today_date.getDate()+"/"+today_date.getFullYear());
trace(date_str); // displays current date in United States date format
```

### Voir également

`getMinutes (méthode Date.getMinutes)`, `getDate (méthode Date.getDate)`, `getFullYear (méthode Date.getFullYear)`

## getDate (méthode Date.getDate)

```
public getDate() : Number
```

Renvoie le jour du mois (entier de 1 à 31) de l'objet Date spécifié, conformément à l'heure locale. L'heure locale est déterminée par le système d'exploitation sur lequel Flash Player s'exécute.

**Disponibilité :** ActionScript 1.0 ; Flash Player 5

### Valeur renvoyée

`Number` - Entier.

### Exemple

L'exemple suivant crée un nouvel objet Date et concatène les valeurs renvoyées de `Date.getMonth()`, `Date.getDate()` et `Date.getFullYear()` :

```
var today_date:Date = new Date();
var date_str:String = (today_date.getDate()+"/
  "+(today_date.getMonth()+1)+"/"+today_date.getFullYear());
trace(date_str); // displays current date in United States date format
```

### Voir également

`getMinutes (méthode Date.getMinutes)`, `getFullYear (méthode Date.getFullYear)`

## getDay (méthode Date.getDay)

```
public getDay() : Number
```

Renvoie le jour de la semaine (0 pour dimanche, 1 pour lundi, etc.) de l'objet Date spécifié conformément à l'heure locale. L'heure locale est déterminée par le système d'exploitation sur lequel Flash Player s'exécute.

**Disponibilité :** ActionScript 1.0 ; Flash Player 5

### Valeur renvoyée

`Number` - Entier représentant le jour de la semaine.

### Exemple

L'exemple suivant crée un nouvel objet Date et utilise la méthode `getDay()` afin de déterminer le jour actuel de la semaine :

```
var dayOfWeek_array:Array = new Array("Sunday", "Monday", "Tuesday",
  "Wednesday", "Thursday", "Friday", "Saturday");
var today_date:Date = new Date();
var day_str:String = dayOfWeek_array[today_date.getDay()];
trace("Today is "+day_str);
```

## getFullYear (méthode Date.getFullYear)

```
public getFullYear() : Number
```

Renvoie l'année entière (un nombre à quatre chiffres, tel que 2000) de l'objet Date spécifié, conformément à l'heure locale. L'heure locale est déterminée par le système d'exploitation sur lequel Flash Player s'exécute.

**Disponibilité :** ActionScript 1.0 ; Flash Player 5

### Valeur renvoyée

`Number` - Entier représentant l'année.

### Exemple

L'exemple suivant utilise le constructeur pour créer un objet Date. L'instruction trace affiche la valeur renvoyée par la méthode `getFullYear()`

```
var my_date:Date = new Date();
trace(my_date.getYear()); // displays 104
trace(my_date.getFullYear()); // displays current year
```

## getHours (méthode Date.getHours)

```
public getHours() : Number
```

Renvoie l'heure (un entier de 0 à 23) de l'objet Date spécifié, conformément à l'heure locale. L'heure locale est déterminée par le système d'exploitation sur lequel Flash Player s'exécute.

**Disponibilité :** ActionScript 1.0 ; Flash Player 5

### Valeur renvoyée

`Number` - Entier.

### Exemple

L'exemple suivant utilise le constructeur pour créer un objet Date en fonction de l'heure actuelle et utilise la méthode `getHours()` pour afficher les valeurs d'heure de cet objet :

```
var my_date:Date = new Date();
trace(my_date.getHours());

var my_date:Date = new Date();
var hourObj:Object = getHoursAmPm(my_date.getHours());
trace(hourObj.hours);
trace(hourObj.ampm);

function getHoursAmPm(hour24:Number):Object {
  var returnObj:Object = new Object();
  returnObj.ampm = (hour24<12) ? "AM" : "PM";
  var hour12:Number = hour24%12;
  if (hour12 == 0) {
  hour12 = 12;
  }
  returnObj.hours = hour12;
  return returnObj;
}
```

## getMilliseconds (méthode Date.getMilliseconds)

```
public getMilliseconds() : Number
```

Renvoie les millisecondes (entier de 0 à 999) de l'objet Date spécifié, conformément à l'heure locale. L'heure locale est déterminée par le système d'exploitation sur lequel Flash Player s'exécute.

**Disponibilité :** ActionScript 1.0 ; Flash Player 5

### Valeur renvoyée

`Number` - Entier.

### Exemple

L'exemple suivant utilise le constructeur pour créer un objet Date en fonction de l'heure actuelle et utilise la méthode `getMilliseconds()` pour renvoyer la valeur en millisecondes de cet objet :

```
var my_date:Date = new Date();
trace(my_date.getMilliseconds());
```

# getMinutes (méthode Date.getMinutes)

```
public getMinutes() : Number
```

Renvoie les minutes (entier de 0 à 59) de l'objet Date spécifié, conformément à l'heure locale. L'heure locale est déterminée par le système d'exploitation sur lequel Flash Player s'exécute.

**Disponibilité :** ActionScript 1.0 ; Flash Player 5

## Valeur renvoyée

`Number` - Entier.

## Exemple

L'exemple suivant utilise le constructeur pour créer un objet Date en fonction de l'heure actuelle et utilise la méthode `getMinutes()` pour renvoyer la valeur de minutes de cet objet :

```
var my_date:Date = new Date();
trace(my_date.getMinutes());
```

# getMinutes (méthode Date.getMinutes)

```
public getMonth() : Number
```

Renvoie le mois (0 pour janvier, 1 pour février, etc.) de l'objet Date spécifié conformément à l'heure locale. L'heure locale est déterminée par le système d'exploitation sur lequel Flash Player s'exécute.

**Disponibilité :** ActionScript 1.0 ; Flash Player 5

## Valeur renvoyée

`Number` - Entier.

## Exemple

L'exemple suivant utilise le constructeur pour créer un objet Date en fonction de l'heure actuelle et utilise la méthode `getMonth()` pour renvoyer la valeur en mois de cet objet :

```
var my_date:Date = new Date();
trace(my_date.getMonth());
```

L'exemple suivant utilise le constructeur pour créer un objet Date en fonction de l'heure actuelle et utilise la méthode `getMonth()` pour afficher le mois en cours en tant que valeur numérique, puis le nom du mois.

```
var my_date:Date = new Date();
trace(my_date.getMonth());
trace(getMonthAsString(my_date.getMonth()));
function getMonthAsString(month:Number):String {
```

```
  var monthNames_array:Array = new Array("January", "February", "March",
  "April", "May", "June", "July", "August", "September", "October",
  "November", "December");
  return monthNames_array[month];
}
```

# getSeconds (méthode Date.getSeconds)

```
public getSeconds() : Number
```

Renvoie les secondes (entier de 0 à 59) de l'objet Date spécifié, conformément à l'heure locale. L'heure locale est déterminée par le système d'exploitation sur lequel Flash Player s'exécute.

**Disponibilité :** ActionScript 1.0 ; Flash Player 5

## Valeur renvoyée

`Number` - Entier.

## Exemple

L'exemple suivant utilise le constructeur pour créer un objet Date en fonction de l'heure actuelle et utilise la méthode `getSeconds()` pour renvoyer la valeur en mois de cet objet :

```
var my_date:Date = new Date();
trace(my_date.getSeconds());
```

# getTime (méthode Date.getTime)

```
public getTime() : Number
```

Renvoie le nombre de millisecondes écoulées depuis le premier janvier 1970 à minuit, heure universelle, pour l'objet Date spécifié. Utilisez cette méthode pour représenter un instant spécifique dans le temps lorsque vous comparez deux ou plusieurs objets Date.

**Disponibilité :** ActionScript 1.0 ; Flash Player 5

## Valeur renvoyée

`Number` - Entier.

## Exemple

L'exemple suivant utilise le constructeur pour créer un objet Date en fonction de l'heure actuelle et utilise la méthode `getTime()` pour renvoyer le nombre de millisecondes écoulées depuis le premier janvier 1970 à minuit :

```
var my_date:Date = new Date();
trace(my_date.getTime());
```

# getTimezoneOffset (méthode Date.getTimezoneOffset)

```
public getTimezoneOffset() : Number
```

Renvoie la différence, en minutes, entre l'heure locale de l'ordinateur et l'heure universelle.

**Disponibilité :** ActionScript 1.0 ; Flash Player 5

## Valeur renvoyée

`Number` - Entier.

## Exemple

L'exemple suivant renvoie la différence entre l'heure d'été locale à San Francisco et l'heure universelle. L'heure d'été est factorisée dans le résultat renvoyé uniquement si la date définie dans l'objet Date se trouve dans la plage de l'heure d'été. Le résultat dans cet exemple est 420 minutes, il est affiché dans le panneau de sortie (7 heures * 60 minutes/heure = 420 minutes). Cet exemple est l'heure d'été de la côte Ouest des Etats-Unis (PDT) qui est égale à GMT moins 7 heures. Le résultat varie en fonction du lieu et de l'époque de l'année.

```
var my_date:Date = new Date();
trace(my_date.getTimezoneOffset());
```

# getUTCDate (méthode Date.getUTCDate)

```
public getUTCDate() : Number
```

Renvoie le jour du mois (entier de 1 à 31) de l'objet Date spécifié, conformément à l'heure universelle.

**Disponibilité :** ActionScript 1.0 ; Flash Player 5

## Valeur renvoyée

`Number` - Entier.

## Exemple

L'exemple suivant crée un nouvel objet Date et utilise `Date.getUTCDate()` et `Date.getDate()`. La valeur renvoyée par `Date.getUTCDate()` peut différer de celle renvoyée par `Date.getDate()`,, en fonction de la relation qui existe entre votre fuseau horaire local et l'heure universelle.

```
var my_date:Date = new Date(2004,8,25);
trace(my_date.getUTCDate()); // output: 25
```

### Voir également

`getDate (méthode Date.getDate)`

## getUTCDay (méthode Date.getUTCDay)

```
public getUTCDay() : Number
```

Renvoie le jour de la semaine (0 pour dimanche, 1 pour lundi, etc.) de l'objet Date spécifié, conformément à l'heure universelle.

**Disponibilité :** ActionScript 1.0 ; Flash Player 5

### Valeur renvoyée

`Number` - Entier.

### Exemple

L'exemple suivant crée un nouvel objet Date et utilise `Date.getUTCDay()` et `Date.getDay()`. La valeur renvoyée par `Date.getUTCDay()` peut différer de celle renvoyée par `Date.getDay()`,, en fonction de la relation qui existe entre votre fuseau horaire local et l'heure universelle.

```
var today_date:Date = new Date();
trace(today_date.getDay()); // output will be based on local timezone
trace(today_date.getUTCDay()); // output will equal getDay() plus or minus
  one
```

### Voir également

`getDay (méthode Date.getDay)`

## getUTCFullYear (méthode Date.getUTCFullYear)

```
public getUTCFullYear() : Number
```

Renvoie l'année à quatre chiffres de l'objet Date spécifié, conformément à l'heure universelle.

**Disponibilité :** ActionScript 1.0 ; Flash Player 5

### Valeur renvoyée

`Number` - Entier.

### Exemple

L'exemple suivant crée un nouvel objet Date object et utilise `Date.getUTCFullYear()` et `Date.getFullYear()`. La valeur renvoyée par `Date.getUTCFullYear()` peut différer de celle renvoyée par `Date.getFullYear()` si la date du jour est le 31 décembre ou le 1 janvier, en fonction de la relation qui existe entre votre fuseau horaire local et l'heure universelle.

```
var today_date:Date = new Date();
trace(today_date.getFullYear()); // display based on local timezone
trace(today_date.getUTCFullYear()); // displays getYear() plus or minus 1
```

### Voir également

`getFullYear (méthode Date.getFullYear)`

## getUTCHours (méthode Date.getUTCHours)

```
public getUTCHours() : Number
```

Renvoie l'heure (entier de 0 à 23) de l'objet Date spécifié, conformément à l'heure universelle.

**Disponibilité :** ActionScript 1.0 ; Flash Player 5

### Valeur renvoyée

`Number` - Entier.

### Exemple

L'exemple suivant crée un nouvel objet Date et utilise `Date.getUTCHours()` et `Date.getHours()`. La valeur renvoyée par `Date.getUTCHours()` peut différer de celle renvoyée par `Date.getHours()`,, en fonction de la relation qui existe entre votre fuseau horaire local et l'heure universelle.

```
var today_date:Date = new Date();
trace(today_date.getHours()); // display based on local timezone
trace(today_date.getUTCHours()); // display equals getHours() plus or minus
  12
```

### Voir également

`getHours (méthode Date.getHours)`

# getUTCMilliseconds (méthode Date.getUTCMilliseconds)

```
public getUTCMilliseconds() : Number
```

Renvoie les millisecondes (entier de 0 à 999) de l'objet Date spécifié, conformément à l'heure universelle.

**Disponibilité :** ActionScript 1.0 ; Flash Player 5

## Valeur renvoyée

`Number` - Entier.

## Exemple

L'exemple suivant crée un nouvel objet Date et utilise la méthode `getUTCMilliseconds()` pour renvoyer la valeur de millisecondes de l'objet Date.

```
var today_date:Date = new Date();
trace(today_date.getUTCMilliseconds());
```

# getUTCMinutes (méthode Date.getUTCMinutes)

```
public getUTCMinutes() : Number
```

Renvoie les minutes (entier de 0 à 59) de l'objet Date spécifié, conformément à l'heure universelle.

**Disponibilité :** ActionScript 1.0 ; Flash Player 5

## Valeur renvoyée

`Number` - Entier.

## Exemple

L'exemple suivant crée un nouvel objet Date et utilise la méthode `getUTCMinutes()` pour renvoyer la valeur en minutes de l'objet Date.

```
var today_date:Date = new Date();
trace(today_date.getUTCMinutes());
```

# getUTCMonth (méthode Date.getUTCMonth)

```
public getUTCMonth() : Number
```

Renvoie le mois (0 [janvier] à 11 [décembre]) de l'objet Date spécifié, conformément à l'heure universelle.

**Disponibilité :** ActionScript 1.0 ; Flash Player 5

### Valeur renvoyée

`Number` - Entier.

### Exemple

L'exemple suivant crée un nouvel objet Date et utilise `Date.getUTCMonth()` et `Date.getMonth()`. La valeur renvoyée par `Date.getUTCMonth()` peut différer de celle renvoyée par `Date.getMonth()` si la date du jour est le premier ou le dernier jour du mois, en fonction de la relation qui existe entre votre fuseau horaire local et l'heure universelle.

```
var today_date:Date = new Date();
trace(today_date.getMonth()); // output based on local timezone
trace(today_date.getUTCMonth()); // output equals getMonth() plus or minus
  1
```

### Voir également

`getMinutes (méthode Date.getMinutes)`

## getUTCSeconds (méthode Date.getUTCSeconds)

```
public getUTCSeconds() : Number
```

Renvoie les secondes (entier de 0 à 59) de l'objet Date spécifié, conformément à l'heure universelle.

**Disponibilité :** ActionScript 1.0 ; Flash Player 5

### Valeur renvoyée

`Number` - Entier.

### Exemple

L'exemple suivant crée un nouvel objet Date et utilise la méthode `getUTCSeconds()` pour renvoyer la valeur en secondes de l'objet Date.

```
var today_date:Date = new Date();
trace(today_date.getUTCSeconds());
```

## getUTCYear (méthode Date.getUTCYear)

```
public getUTCYear() : Number
```

Renvoie l'année de cette `Date` conformément à l'heure universelle. L'année est l'année entière moins 1900. Par exemple, l'année 2000 est représentée comme 100.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

### Valeur renvoyée

`Number` - Entier.

### Exemple

L'exemple suivant crée un nouvel objet Date et utilise `Date.getUTCFullYear()` et `Date.getFullYear()`. La valeur renvoyée par `Date.getUTCFullYear()` peut différer de celle renvoyée par `Date.getFullYear()` si la date du jour est le 31 décembre ou le 1 janvier, en fonction de la relation qui existe entre votre fuseau horaire local et l'heure universelle.

```
var today_date:Date = new Date();

trace(today_date.getFullYear()); // display based on local timezone

trace(today_date.getUTCFullYear()); // displays getYear() plus or minus 1
```

## getYear (méthode Date.getYear)

```
public getYear() : Number
```

Renvoie l'année de l'objet Date spécifié, conformément à l'heure locale. L'heure locale est déterminée par le système d'exploitation sur lequel Flash Player s'exécute. L'année est l'année entière moins 1900. Par exemple, l'année 2000 est représentée comme 100.

**Disponibilité :** ActionScript 1.0 ; Flash Player 5

### Valeur renvoyée

`Number` - Entier.

### Exemple

L'exemple suivant crée un objet Date dont le mois et l'année sont définis sur Mai 2004. La méthode `Date.getYear()` renvoie 104 et `Date.getFullYear()` renvoie 2004 :

```
var today_date:Date = new Date(2004,4);
trace(today_date.getYear()); // output: 104
trace(today_date.getFullYear()); // output: 2004
```

### Voir également

`getFullYear (méthode Date.getFullYear)`

# setDate (méthode Date.setDate)

```
public setDate(date:Number) : Number
```

Définit le jour du mois de l'objet Date spécifié, conformément à l'heure locale, et renvoie la nouvelle heure en millisecondes. L'heure locale est déterminée par le système d'exploitation sur lequel Flash Player s'exécute.

**Disponibilité :** ActionScript 1.0 ; Flash Player 5

## Paramètres

*date*:Number - Entier compris entre 1 et 31.

## Valeur renvoyée

Number - Entier.

## Exemple

L'exemple suivant crée initialement un nouvel objet Date dont la date est définie sur 15 mai 2004, puis utilise la méthode Date.setDate() pour modifier la date et la définir sur 25 mai 2004 :

```
var today_date:Date = new Date(2004,4,15);
trace(today_date.getDate()); //displays 15
today_date.setDate(25);
trace(today_date.getDate()); //displays 25
```

# setFullYear (méthode Date.setFullYear)

```
public setFullYear(year:Number, [month:Number], [date:Number]) : Number
```

Définit l'année de l'objet Date spécifié, conformément à l'heure locale, et renvoie la nouvelle heure en millisecondes. Si les paramètres month et date sont spécifiés, ils sont définis sur l'heure locale. L'heure locale est déterminée par le système d'exploitation sur lequel Flash Player s'exécute.

L'appel de cette méthode ne modifie pas les autres champs de l'objet Date spécifié mais Date.getUTCDay() et Date.getDay() peuvent signaler une nouvelle valeur si le jour de la semaine change suite à l'appel de cette méthode.

**Disponibilité :** ActionScript 1.0 ; Flash Player 5

## Paramètres

*year*:Number - Nombre à quatre chiffres spécifiant une année. Les nombres à deux chiffres ne représentent pas les années à quatre chiffres ; par exemple, 99 ne représente pas l'année 1999 mais l'an 99.

*month*:Number [facultatif] - Entier compris entre 0 (janvier) et 11 (décembre). Si vous omettez ce paramètre, le champ Mois de l'objet Date spécifié ne sera pas modifié.

*date*:Number [facultatif] - Nombre compris entre 1 et 31. Si vous omettez ce paramètre, le champ Date de l'objet Date spécifié ne sera pas modifié.

### Valeur renvoyée

Number - Entier.

### Exemple

L'exemple suivant crée initialement un nouvel objet Date dont la date est définie sur 15 mai 2004, puis utilise la méthode Date.setFullYear() pour modifier la date et la définir sur 15 mai 2002 :

```
var my_date:Date = new Date(2004,4,15);
trace(my_date.getFullYear()); //output: 2004
my_date.setFullYear(2002);
trace(my_date.getFullYear()); //output: 2002
```

### Voir également

getUTCDay (méthode Date.getUTCDay), getDay (méthode Date.getDay)

## setHours (méthode Date.setHours)

```
public setHours(hour:Number) : Number
```

Définit les heures de l'objet Date spécifié conformément à l'heure locale et renvoie la nouvelle heure en millisecondes. L'heure locale est déterminée par le système d'exploitation sur lequel Flash Player s'exécute.

**Disponibilité :** ActionScript 1.0 ; Flash Player 5

### Paramètres

*hour*:Number - Entier compris entre 0 (minuit) et 23 (23h00).

### Valeur renvoyée

Number - Entier.

### Exemple

L'exemple suivant crée initialement un nouvel objet Date dont l'heure et la date sont définies sur 15 mai 2004 à 8:00, puis utilise la méthode Date.setHours() pour modifier l'heure et la définir sur 16:00 :

```
var my_date:Date = new Date(2004,4,15,8);
trace(my_date.getHours()); // output: 8
my_date.setHours(16);
trace(my_date.getHours()); // output: 16
```

# setMilliseconds (méthode Date.setMilliseconds)

```
public setMilliseconds(millisecond:Number) : Number
```

Définit les millisecondes de l'objet Date spécifié conformément à l'heure locale et renvoie la nouvelle heure en millisecondes. L'heure locale est déterminée par le système d'exploitation sur lequel Flash Player s'exécute.

**Disponibilité :** ActionScript 1.0 ; Flash Player 5

## Paramètres

*millisecond*:Number - Entier compris entre 0 et 999.

## Valeur renvoyée

Number - Entier.

## Exemple

L'exemple suivant crée initialement un nouvel objet Date dont la date est définie sur 15 mai 2004 à 8:30 (valeur de millisecondes définie sur 250), puis utilise la méthode Date.setMilliseconds() pour modifier la valeur de millisecondes et la définir sur 575 :

```
var my_date:Date = new Date(2004,4,15,8,30,0,250);
trace(my_date.getMilliseconds()); // output: 250
my_date.setMilliseconds(575);
trace(my_date.getMilliseconds()); // output: 575
```

# setMinutes (méthode Date.setMinutes)

```
public setMinutes(minute:Number) : Number
```

Définit les minutes de l'objet Date spécifié conformément à l'heure locale et renvoie la nouvelle heure en millisecondes. L'heure locale est déterminée par le système d'exploitation sur lequel Flash Player s'exécute.

**Disponibilité :** ActionScript 1.0 ; Flash Player 5

## Paramètres

*minute*:Number - Entier compris entre 0 et 59.

### Valeur renvoyée

`Number` - Entier.

### Exemple

L'exemple suivant crée initialement un nouvel objet Date dont l'heure et la date sont définies sur 15 mai 2004 à 8:00, puis utilise la méthode `Date.setMinutes()` pour modifier l'heure et la définir sur 8:30 :

```
var my_date:Date = new Date(2004,4,15,8,0);
trace(my_date.getMinutes()); // output: 0
my_date.setMinutes(30);
trace(my_date.getMinutes()); // output: 30
```

## setMonth (méthode Date.setMonth)

```
public setMonth(month:Number, [date:Number]) : Number
```

Définit le mois de l'objet Date spécifié conformément à l'heure locale et renvoie la nouvelle heure en millisecondes. L'heure locale est déterminée par le système d'exploitation sur lequel Flash Player s'exécute.

**Disponibilité :** ActionScript 1.0 ; Flash Player 5

### Paramètres

*`month`*`:Number` - Entier compris entre 0 (janvier) et 11 (décembre).

*`date`*`:Number` [facultatif] - Entier compris entre 1 et 31. Si vous omettez ce paramètre, le champ Date de l'objet Date spécifié ne sera pas modifié.

### Valeur renvoyée

`Number` - Entier.

### Exemple

L'exemple suivant crée initialement un nouvel objet Date dont la date est définie sur 15 mai 2004, puis utilise la méthode `Date.setMonth()` pour modifier la date et la définir sur 15.06.04 :

```
var my_date:Date = new Date(2004,4,15);
trace(my_date.getMonth()); //output: 4
my_date.setMonth(5);
trace(my_date.getMonth()); //output: 5
```

# setSeconds (méthode Date.setSeconds)

```
public setSeconds(second:Number) : Number
```

Définit les secondes de l'objet Date spécifié conformément à l'heure locale et renvoie la nouvelle heure en millisecondes. L'heure locale est déterminée par le système d'exploitation sur lequel Flash Player s'exécute.

**Disponibilité :** ActionScript 1.0 ; Flash Player 5

## Paramètres

*second*:Number - Entier compris entre 0 et 59.

## Valeur renvoyée

Number - Entier.

## Exemple

L'exemple suivant crée initialement un nouvel objet Date dont l'heure et la date sont définies sur 15 mai 2004 à 8:00:00, puis utilise la méthode `Date.setSeconds()` pour modifier l'heure et la définir sur 8:00:45 :

```
var my_date:Date = new Date(2004,4,15,8,0,0);
trace(my_date.getSeconds()); // output: 0
my_date.setSeconds(45);
trace(my_date.getSeconds()); // output: 45
```

# setTime (méthode Date.setTime)

```
public setMilliseconds(millisecond:Number) : Number
```

Définit la date de l'objet Date spécifié en millisecondes écoulées depuis le premier janvier 1970 à minuit et renvoie la nouvelle heure en millisecondes.

**Disponibilité :** ActionScript 1.0 ; Flash Player 5

## Paramètres

*millisecond*:Number - Nombre ; valeur entière où 0 représente minuit le premier janvier, heure universelle.

## Valeur renvoyée

Number - Entier.

### Exemple

L'exemple suivant crée initialement un nouvel objet Date dont l'heure et la date sont définies sur 15 mai 2004 à 08:00:00, puis utilise la méthode `Date.setTime()` pour modifier l'heure et la définir sur 08:30:00 :

```
var my_date:Date = new Date(2004,4,15,8,0,0);
var myDate_num:Number = my_date.getTime(); // convert my_date to
  milliseconds
myDate_num += 30 * 60 * 1000; // add 30 minutes in milliseconds
my_date.setTime(myDate_num); // set my_date Date object 30 minutes forward
trace(my_date.getFullYear()); // output: 2004
trace(my_date.getMonth()); // output: 4
trace(my_date.getDate()); // output: 15
trace(my_date.getHours()); // output: 8
trace(my_date.getMinutes()); // output: 30
```

## setUTCDate (méthode Date.setUTCDate)

```
public setUTCDate(date:Number) : Number
```

Définit la date de l'objet Date spécifié conformément à l'heure universelle et renvoie la nouvelle heure en millisecondes. L'appel de cette méthode ne modifie pas les autres champs de l'objet Date spécifié mais `Date.getUTCDay()` et `Date.getDay()` peuvent signaler une nouvelle valeur si le jour de la semaine change suite à l'appel de cette méthode.

**Disponibilité :** ActionScript 1.0 ; Flash Player 5

### Paramètres

*`date`*`:Number` - Nombre ; entier compris entre 1 et 31.

### Valeur renvoyée

`Number` - Entier.

### Exemple

L'exemple suivant crée initialement un nouvel objet Date à la date du jour, utilise la méthode `Date.setUTCDate()` pour modifier la valeur de date et la définir sur 10, puis la définir de nouveau sur 25 :

```
var my_date:Date = new Date();
my_date.setUTCDate(10);
trace(my_date.getUTCDate()); // output: 10
my_date.setUTCDate(25);
trace(my_date.getUTCDate()); // output: 25
```

### Voir également

getUTCDay (méthode Date.getUTCDay), getDay (méthode Date.getDay)

## setUTCFullYear (méthode Date.setUTCFullYear)

```
public setUTCFullYear(year:Number, [month:Number], [date:Number]) : Number
```

Définit l'année de l'objet Date spécifié (*my_date*) conformément à l'heure universelle et renvoie la nouvelle heure en millisecondes.

Cette méthode peut également définir le mois et la date représentés par l'objet Date spécifié. L'appel de cette méthode ne modifie pas les autres champs de l'objet Date spécifié mais `Date.getUTCDay()` et `Date.getDay()` peuvent signaler une nouvelle valeur si le jour de la semaine change suite à l'appel de cette méthode.

**Disponibilité :** ActionScript 1.0 ; Flash Player 5

### Paramètres

*year*:`Number` - Entier représentant l'année spécifiée en tant qu'année entière à quatre chiffres, telle que 2000.

*month*:`Number` [facultatif] - Entier compris entre 0 (janvier) et 11 (décembre). Si vous omettez ce paramètre, le champ Mois de l'objet Date spécifié ne sera pas modifié.

*date*:`Number` [facultatif] - Entier compris entre 1 et 31. Si vous omettez ce paramètre, le champ Date de l'objet Date spécifié ne sera pas modifié.

### Valeur renvoyée

`Number` - Entier.

### Exemple

L'exemple suivant crée initialement un nouvel objet Date à la date du jour, utilise la méthode `Date.setUTCFullYear()` pour modifier la valeur de l'année et la définir sur 2001, puis définir la date sur 25 mai 1995 :

```
var my_date:Date = new Date();
my_date.setUTCFullYear(2001);
trace(my_date.getUTCFullYear()); // output: 2001
my_date.setUTCFullYear(1995, 4, 25);
trace(my_date.getUTCFullYear()); // output: 1995
trace(my_date.getUTCMonth()); // output: 4
trace(my_date.getUTCDate()); // output: 25
```

### Voir également

getUTCDay (méthode Date.getUTCDay), getDay (méthode Date.getDay)

# setUTCHours (méthode Date.setUTCHours)

```
public setUTCHours(hour:Number, [minute:Number], [second:Number],
  [millisecond:Number]) : Number
```

Définit l'heure de l'objet Date spécifié conformément à l'heure universelle et renvoie la nouvelle heure en millisecondes.

**Disponibilité :** ActionScript 1.0 ; Flash Player 5

## Paramètres

*hour*:Number - Nombre ; entier compris entre 0 (minuit) et 23 (23h00).

*minute*:Number [facultatif] - Nombre ; entier compris entre 0 et 59. Si vous omettez ce paramètre, le champ Minutes de l'objet Date spécifié ne sera pas modifié.

*second*:Number [facultatif] - Nombre ; entier compris entre 0 et 59. Si vous omettez ce paramètre, le champ Secondes de l'objet Date spécifié ne sera pas modifié.

*millisecond*:Number [facultatif] - Nombre ; entier compris entre 0 et 999. Si vous omettez ce paramètre, le champ Millisecondes de l'objet Date spécifié ne sera pas modifié.

## Valeur renvoyée

Number - Entier.

## Exemple

L'exemple suivant crée initialement un nouvel objet Date à la date du jour, utilise la méthode Date.setUTCHours() pour modifier l'heure et la définir sur 8:30, puis la définir de nouveau sur 17:30:47 :

```
var my_date:Date = new Date();
my_date.setUTCHours(8,30);
trace(my_date.getUTCHours()); // output: 8
trace(my_date.getUTCMinutes()); // output: 30
my_date.setUTCHours(17,30,47);
trace(my_date.getUTCHours()); // output: 17
trace(my_date.getUTCMinutes()); // output: 30
trace(my_date.getUTCSeconds()); // output: 47
```

# setUTCMilliseconds (méthode Date.setUTCMilliseconds)

```
public setUTCMilliseconds(millisecond:Number) : Number
```

Définit les millisecondes de l'objet Date spécifié conformément à l'heure universelle et renvoie la nouvelle heure en millisecondes.

**Disponibilité :** ActionScript 1.0 ; Flash Player 5

### Paramètres

*millisecond*:Number - Entier compris entre 0 et 999.

### Valeur renvoyée

Number - Entier.

### Exemple

L'exemple suivant crée initialement un nouvel objet Date dont la date est définie sur 15 mai 2004 à 8:30 (valeur de millisecondes définie sur 250), puis utilise la méthode Date.setUTCMilliseconds() pour modifier la valeur de millisecondes et la définir sur 575 :

```
var my_date:Date = new Date(2004,4,15,8,30,0,250);
trace(my_date.getUTCMilliseconds()); // output: 250
my_date.setUTCMilliseconds(575);
trace(my_date.getUTCMilliseconds()); // output: 575
```

## setUTCMinutes (méthode Date.setUTCMinutes)

```
public setUTCMinutes(minute:Number, [second:Number],
  [millisecond:Number]) : Number
```

Définit les minutes de l'objet Date spécifié conformément à l'heure universelle et renvoie la nouvelle heure en millisecondes.

**Disponibilité :** ActionScript 1.0 ; Flash Player 5

### Paramètres

*minute*:Number - Entier compris entre 0 et 59.

*second*:Number [facultatif] - Entier compris entre 0 et 59. Si vous omettez ce paramètre, le champ Secondes de l'objet Date spécifié ne sera pas modifié.

*millisecond*:Number [facultatif] - Entier compris entre 0 et 999. Si vous omettez ce paramètre, le champ Millisecondes de l'objet Date spécifié ne sera pas modifié.

### Valeur renvoyée

Number - Entier.

### Exemple

L'exemple suivant crée initialement un nouvel objet Date dont l'heure et la date sont définies sur 15 mai 2004 à 08:00:00, puis utilise la méthode `Date.setUTCMinutes()` pour modifier l'heure et la définir sur 08:30:00 :

```
var my_date:Date = new Date(2004,4,15,8,0);
trace(my_date.getUTCMinutes()); // output: 0
my_date.setUTCMinutes(30);
trace(my_date.getUTCMinutes()); // output: 30
```

## setUTCMonth (méthode Date.setUTCMonth)

```
public setUTCMonth(month:Number, [date:Number]) : Number
```

Définit le mois, et éventuellement le jour, de l'objet Date spécifié conformément à l'heure universelle et renvoie la nouvelle heure en millisecondes. L'appel de cette méthode ne modifie pas les autres champs de l'objet Date spécifié, mais `Date.getUTCDay()` et `Date.getDay()` peuvent signaler une nouvelle valeur si le jour de la semaine change suite à la spécification d'une valeur pour le paramètre `date`.

**Disponibilité :** ActionScript 1.0 ; Flash Player 5

### Paramètres

*`month`*`:Number` - Entier compris entre 0 (janvier) et 11 (décembre).

*`date`*`:Number` [facultatif] - Entier compris entre 1 et 31. Si vous omettez ce paramètre, le champ Date de l'objet Date spécifié ne sera pas modifié.

### Valeur renvoyée

`Number` - Entier.

### Exemple

L'exemple suivant crée initialement un nouvel objet Date dont la date est définie sur 15 mai 2004, puis utilise la méthode `Date.setMonth()` pour modifier la date et la définir sur 15 juin 2004 :

```
var today_date:Date = new Date(2004,4,15);
trace(today_date.getUTCMonth()); // output: 4
today_date.setUTCMonth(5);
trace(today_date.getUTCMonth()); // output: 5
```

### Voir également

`getUTCDay (méthode Date.getUTCDay)`, `getDay (méthode Date.getDay)`

# setUTCSeconds (méthode Date.setUTCSeconds)

```
public setUTCSeconds(second:Number, [millisecond:Number]) : Number
```

Définit les secondes de l'objet Date spécifié conformément à l'heure universelle et renvoie la nouvelle heure en millisecondes.

**Disponibilité :** ActionScript 1.0 ; Flash Player 5

## Paramètres

*second*:Number - Entier compris entre 0 et 59.

*millisecond*:Number [facultatif] - Entier compris entre 0 et 999. Si vous omettez ce paramètre, le champ Millisecondes de l'objet Date spécifié ne sera pas modifié.

## Valeur renvoyée

Number - Entier.

## Exemple

L'exemple suivant crée initialement un nouvel objet Date dont l'heure et la date sont définies sur 15 mai 2004 à 8:00:00, puis utilise la méthode `Date.setSeconds()` pour modifier l'heure et la définir sur 08:30:45 :

```
var my_date:Date = new Date(2004,4,15,8,0,0);
trace(my_date.getUTCSeconds()); // output: 0
my_date.setUTCSeconds(45);
trace(my_date.getUTCSeconds()); // output: 45
```

# setYear (méthode Date.setYear)

```
public setYear(year:Number) : Number
```

Définit l'année de l'objet Date spécifié conformément à l'heure locale et renvoie la nouvelle heure en millisecondes. L'heure locale est déterminée par le système d'exploitation sur lequel Flash Player s'exécute.

**Disponibilité :** ActionScript 1.0 ; Flash Player 5

## Paramètres

*year*:Number - Nombre représentant l'année. Si `year` est un entier compris entre 0 et 99, `setYear` définit l'année sur 1900 + `year` ; sinon, l'année correspond à la valeur du paramètre `year`.

## Valeur renvoyée

Number - Entier.

### Exemple

L'exemple suivant crée un nouvel objet Date dont la date est définie sur 25 mai 2004, utilise la méthode `setYear()` pour modifier la valeur de l'année et la définir sur 1999, puis définir l'année sur 2003 :

```
var my_date:Date = new Date(2004,4,25);
trace(my_date.getYear()); // output: 104
trace(my_date.getFullYear()); // output: 2004
my_date.setYear(99);
trace(my_date.getYear()); // output: 99
trace(my_date.getFullYear()); // output: 1999
my_date.setYear(2003);
trace(my_date.getYear()); // output: 103
trace(my_date.getFullYear()); // output: 2003
```

## toString (méthode Date.toString)

```
public toString() : String
```

Renvoie une valeur de chaîne pour l'objet de date spécifié dans un format lisible.

**Disponibilité :** ActionScript 1.0 ; Flash Player 5

### Valeur renvoyée

`String` - Chaîne.

### Exemple

L'exemple suivant renvoie les informations dans l'objet Date `dateOfBirth_date` sous forme de chaîne : La sortie des instructions trace est en heure locale et varie en fonction de : Pour l'heure d'été de la côte Ouest des Etats-Unis (PDT), la sortie est l'heure universelle moins 7 heures : lundi 12 août 1974 à 18:15:00, GMT - 7h00.

```
var dateOfBirth_date:Date = new Date(74, 7, 12, 18, 15);
trace (dateOfBirth_date);
trace (dateOfBirth_date.toString());
```

## UTC (méthode Date.UTC)

```
public static UTC(year:Number, month:Number, [date:Number], [hour:Number],
  [minute:Number], [second:Number], [millisecond:Number]) : Number
```

Renvoie le nombre de millisecondes écoulées entre le premier janvier 1970 à minuit, heure universelle, et l'heure spécifiée dans les paramètres. Il s'agit d'une méthode statique appelée via le constructeur de l'objet Date, et non pas via un objet Date spécifique. Cette méthode vous permet de créer un objet Date qui adopte l'heure universelle, tandis que le constructeur Date adopte l'heure locale.

**Disponibilité :** ActionScript 1.0 ; Flash Player 5

### Paramètres

*year*:Number - Entier à quatre chiffres qui représente l'année (par exemple, 2000).

*month*:Number - Entier compris entre 0 (janvier) et 11 (décembre).

*date*:Number [facultatif] - Entier compris entre 1 et 31.

*hour*:Number [facultatif] - Entier compris entre 0 (minuit) et 23 (23h00).

*minute*:Number [facultatif] - Entier compris entre 0 et 59.

*second*:Number [facultatif] - Entier compris entre 0 et 59.

*millisecond*:Number [facultatif] - Entier compris entre 0 et 999.

### Valeur renvoyée

Number - Entier.

### Exemple

L'exemple suivant crée un nouvel objet Date maryBirthday_date défini conformément à l'heure universelle. Cet exemple reprend l'exemple utilisé pour la nouvelle méthode du constructeur new Date, en se basant sur l'heure universelle. La sortie est en heure locale et varie en fonction de : Pour l'heure d'été de la côte Ouest des Etats-Unis (PDT), la sortie est l'heure universelle moins 7 heures : dimanche 11 août 1974 à 17:00:00, GMT - 7h00.

```
var maryBirthday_date:Date = new Date(Date.UTC(1974, 7, 12));
trace(maryBirthday_date);
```

## valueOf (méthode Date.valueOf)

```
public valueOf() : Number
```

Renvoie le nombre de millisecondes écoulées depuis le premier janvier 1970 à minuit, heure universelle, pour cette Date.

**Disponibilité :** ActionScript 1.0 ; Flash Player 5

### Valeur renvoyée

Number - Nombre de millisecondes.

# DisplacementMapFilter (flash.filters.DisplacementMapFilter)

```
Object
  |
  +-flash.filters.BitmapFilter
    |
    +-flash.filters.DisplacementMapFilter
```

`public class` ***DisplacementMapFilter***
`extends BitmapFilter`

La classe DisplacementMapFilter utilise les valeurs de pixels de l'objet BitmapData spécifié (intitulé *image de mappage du déplacement*) pour déplacer un objet situé sur la scène, telle qu'une occurrence MovieClip. Vous pouvez utiliser ce filtre pour obtenir un effet voilé ou de tacheté sur une occurrence BitmapData ou MovieClip.

L'utilisation de filtres dépend de l'objet auquel vous appliquez le filtre.

Pour appliquer des filtres aux clips lors de l'exécution, utilisez la propriété `filters`. Lorsque vous définissez la propriété `filters` d'un objet, celui-ci n'est pas modifié. En outre, vous pouvez l'annuler en supprimant la propriété `filters`.

Pour appliquer des filtres aux occurrences BitmapData, utilisez la méthode `BitmapData.applyFilter()`. L'appel `applyFilter()` sur un objet BitmapData modifie ce dernier et ne peut pas être annulé.

Vous pouvez également appliquer des effets de filtre aux images et aux données vidéo pendant la programmation. Pour plus d'informations, consultez la documentation relative à la programmation.

Si vous appliquez un filtre à un clip ou à un bouton, la propriété `cacheAsBitmap` du clip ou du bouton est définie sur `true`. Si vous supprimez tous les filtres, la valeur d'origine de `cacheAsBitmap` est restaurée.

Le filtre utilise la formule suivante :

```
dstPixel[x, y] = srcPixel[x + ((componentX(x, y) - 128) * scaleX) / 256, y +
  ((componentY(x, y) - 128) * scaleY) / 256]
```

où `componentX(x, y)` attribue la valeur de couleur `componentX` de la propriété `mapBitmap` à `(x - mapPoint.x ,y - mapPoint.y)`.

L'image de mappage utilisée par le filtre est redimensionnée afin de correspondre au redimensionnement de la scène. Elle n'est en aucun cas redimensionnée lorsque l'objet l'est.

Ce filtre prend en charge le redimensionnement de la scène, mais pas le redimensionnement général, la rotation ni l'inclinaison. Si l'objet lui-même est redimensionné (si l'échelle *x* et l'échelle *y*ne sont pas à 100 %), l'effet du filtre n'est pas redimensionné. Le redimensionnement est effectué uniquement en cas de zoom avant sur la scène.

Voici comment fonctionne la classe DisplacementMapFilter. Pour chaque pixel (*x*,*y*) dans la bitmap de *destination*, la classe DisplacementMapFilter effectue ce qui suit :

- Elle acquiert la couleur de (*x*,*y*) dans le *mappage* de la bitmap
- Elle calcule un décalage sur la base de cette couleur
- Elle recherche cet emplacement de décalage (*x*+*dx*,*y*+*dy*) dans la bitmap *source*
- Elle inscrit ce pixel à la destination (*x*,*y*), si les conditions de limites le permettent.

Un filtre ne peut s'appliquer si l'image résultante dépasse 2 880 pixels en largeur ou en hauteur. Par exemple, si vous faites un zoom avant sur un grand clip auquel un filtre est appliqué, le filtre est désactivé si l'image résultante dépasse la limite de 2 880 pixels.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

## Voir également

`applyFilter (méthode BitmapData.applyFilter)`, `filters (propriété MovieClip.filters)`, `cacheAsBitmap (propriété MovieClip.cacheAsBitmap)`

## Résumé des propriétés

| Modificateurs | Propriété | Description |
|---|---|---|
| | `alpha:Number` | Spécifie la transparence alpha à utiliser pour les déplacements en dehors des limites. |
| | `color:Number` | Spécifie la couleur à utiliser pour les déplacements en dehors des limites. |
| | `componentX:Number` | Décrit le canal de couleur à utiliser dans l'image de mappage pour déplacer le résultat *x*. |
| | `componentY:Number` | Décrit le canal de couleur à utiliser dans l'image de mappage pour déplacer le résultat *y*. |
| | `mapBitmap:BitmapData` | Un objet BitmapData contenant les données de mappage du déplacement. |
| | `mapPoint:Point` | Une valeur `flash.geom.Point` représentant le décalage du coin supérieur gauche du clip cible par rapport au coin supérieur gauche de l'image de mappage. |
| | `mode:String` | Le mode du filtre. |
| | `scaleX:Number` | Le multiplicateur à utiliser pour mettre à l'échelle le résultat du déplacement *x* à partir du calcul de mappage. |
| | `scaleY:Number` | Le multiplicateur à utiliser pour mettre à l'échelle le résultat du déplacement *y* à partir du calcul de mappage. |

*Propriétés héritées de la classe Object*

```
constructeur (propriété Object.constructor), __proto__ (Object.__proto__
property), prototype (Object.prototype, propriété), __resolve
(Object.__resolve, propriété)
```

### Récapitulatif des constructeurs

| Signature | Description |
|---|---|
| `DisplacementMapFilter(mapBitmap:BitmapData, mapPoint:Point, componentX:Number, componentY:Number, scaleX:Number, scaleY:Number, [mode:String], [color:Number], [alpha:Number])` | Initialise une occurrence DisplacementMapFilter avec les paramètres spécifiés. |

### Résumé de la méthode

| Modificateurs | Signature | Description |
|---|---|---|
| | `clone() : DisplacementMapFilter` | Renvoie une copie de cet objet filtre. |

*Méthodes héritées de la classe BitmapFilter*

`clone (méthode BitmapFilter.clone )`

*Méthodes héritées de la classe Object*

`addProperty (méthode Object.addProperty), hasOwnProperty (méthode Object.hasOwnProperty), isPropertyEnumerable (méthode Object.isPropertyEnumerable), isPrototypeOf (méthode Object.isPrototypeOf), registerClass (méthode Object.registerClass), toString (méthode Object.toString), unwatch (méthode Object.unwatch), valueOf (méthode Object.valueOf), watch (méthode Object.watch)`

## alpha (propriété DisplacementMapFilter.alpha)

`public alpha : Number`

Spécifie la transparence alpha à utiliser pour les déplacements en dehors des limites. Elle est spécifiée en tant que valeur normalisée comprise entre 0,0 et 1,0. Par exemple, 0,25 définit une valeur de transparence de 25 %. La valeur par défaut est 0. Utilisez cette propriété si la propriété `mode` est définie sur `3, COLOR`.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

## Exemple

L'exemple suivant modifie la propriété hors limites `alpha` pour la mettre à `0x00FF00` sur le clip existant `filteredMc` quand un utilisateur clique sur celui-ci.

```
import flash.filters.DisplacementMapFilter;
import flash.display.BitmapData;
import flash.geom.Point;
import flash.geom.Matrix;
import flash.geom.ColorTransform;

var filteredMc:MovieClip = createDisplacementMapRectangle();

filteredMc.onPress = function() {
  var filter:DisplacementMapFilter = this.filters[0];
  filter.scaleY = 25;
  filter.mode = "color";
  filter.alpha = .25;
  this.filters = new Array(filter);
}

function createDisplacementMapRectangle():MovieClip {
  var mapBitmap:BitmapData = createGradientBitmap(300, 80, 0xFF000000,
  "radial");
  var filter:DisplacementMapFilter = new DisplacementMapFilter(mapBitmap,
  new Point(-30, -30), 1, 1, 10, 10, "wrap", 0x000000, 0x000000);

  var txtBlock:MovieClip = createTextBlock();
  txtBlock._x = 30;
  txtBlock._y = 30;

  txtBlock.filters = new Array(filter);

  return txtBlock;
}

function createGradientBitmap(w:Number, h:Number, bgColor:Number,
  type:String, hide:Boolean):BitmapData {
  var mc:MovieClip = this.createEmptyMovieClip("mc", 1);
  var matrix:Matrix = new Matrix();
  matrix.createGradientBox(w, h, 0, 0, 0);

  mc.beginGradientFill(type, [0xFF0000, 0x0000FF], [100, 100], [0x55,
  0x99], matrix, "pad");
  mc.lineTo(w, 0);
  mc.lineTo(w, h);
  mc.lineTo(0, h);
  mc.lineTo(0, 0);
  mc.endFill();
  (hide == true) ? mc._alpha = 0 : mc._alpha = 100;
```

```
  var bmp:BitmapData = new BitmapData(w, h, true, bgColor);
    bmp.draw(mc, new Matrix(), new ColorTransform(), "normal",
  bmp.rectangle, true);
  mc.attachBitmap(bmp, this.getNextHighestDepth());

  return bmp;
}

function createTextBlock():MovieClip {
  var txtBlock:MovieClip = this.createEmptyMovieClip("txtBlock",
  this.getNextHighestDepth());
  txtBlock.createTextField("txt", this.getNextHighestDepth(), 0, 0, 300,
  80);
  txtBlock.txt.text = "watch the text bend with the displacement map";
  return txtBlock;
}
```

# clone (méthode DisplacementMapFilter.clone)

```
public clone() : DisplacementMapFilter
```

Renvoie une copie de cet objet filtre.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

## Valeur renvoyée

`flash.filters.DisplacementMapFilter` - Nouvelle occurrence DisplacementMapFilter dont les propriétés sont toutes identiques à celles de l'occurrence d'origine.

## Exemple

L'exemple suivant crée trois objets DisplacementMapFilter et les compare : `filter_1` est créé en utilisant le constructeur `DisplacementMapFilter` ; `filter_2` est créé en le définissant comme égal à `filter_1`; et `clonedFilter` est créé en clonant `filter_1`. Veuillez noter que `filter_2` est considéré comme égal à `filter_1`, `clonedFilter` ne l'est pas, même s'il contient les mêmes valeurs que `filter_1`.

```
import flash.filters.DisplacementMapFilter;
import flash.display.BitmapData;
import flash.geom.Point;
import flash.geom.Matrix;
import flash.geom.ColorTransform;

var mapBitmap:BitmapData = createGradientBitmap(300, 80, 0xFF000000,
  "radial", true);

var filter_1:DisplacementMapFilter = new DisplacementMapFilter(mapBitmap,
  new Point(-30, -30), 1, 1, 10, 10, "wrap", 0x000000, 0x000000);
```

```
var filter_2:DisplacementMapFilter = filter_1;
var clonedFilter:DisplacementMapFilter = filter_1.clone();

trace(filter_1 == filter_2); // true
trace(filter_1 == clonedFilter); // false

for(var i in filter_1) {
  trace(">> " + i + ": " + filter_1[i]);
  // >> clone: [type Function]
  // >> alpha: 0
  // >> color: 0
  // >> mode: wrap
  // >> scaleY: 10
  // >> scaleX: 10
  // >> componentY: 1
  // >> componentX: 1
  // >> mapPoint: (-30, -30)
  // >> mapBitmap: [object Object]
}

for(var i in clonedFilter) {
  trace(">> " + i + ": " + clonedFilter[i]);
  // >> clone: [type Function]
  // >> alpha: 0
  // >> color: 0
  // >> mode: wrap
  // >> scaleY: 10
  // >> scaleX: 10
  // >> componentY: 1
  // >> componentX: 1
  // >> mapPoint: (-30, -30)
  // >> mapBitmap: [object Object]
}

function createGradientBitmap(w:Number, h:Number, bgColor:Number,
  type:String, hide:Boolean):BitmapData {
  var mc:MovieClip = this.createEmptyMovieClip("mc", 1);
  var matrix:Matrix = new Matrix();
  matrix.createGradientBox(w, h, 0, 0, 0);

  mc.beginGradientFill(type, [0xFF0000, 0x0000FF], [100, 100], [0x55,
  0x99], matrix, "pad");
  mc.lineTo(w, 0);
  mc.lineTo(w, h);
  mc.lineTo(0, h);
  mc.lineTo(0, 0);
  mc.endFill();
  (hide == true) ? mc._alpha = 0 : mc._alpha = 100;

  var bmp:BitmapData = new BitmapData(w, h, true, bgColor);
```

```
    bmp.draw(mc, new Matrix(), new ColorTransform(), "normal",
  bmp.rectangle, true);
  mc.attachBitmap(bmp, this.getNextHighestDepth());

  return bmp;
}
```

Pour démontrer davantage les relations entre `filter_1`, `filter_2`, et `clonedFilter`, l'exemple suivant modifie la propriété `mode` de `filter_1`. La modification de `mode` démontre que la méthode `clone()` crée une nouvelle occurrence basée sur les valeurs de `filter_1` au lieu de faire référence à ces valeurs.

```
import flash.filters.DisplacementMapFilter;
import flash.display.BitmapData;
import flash.geom.Point;
import flash.geom.Matrix;
import flash.geom.ColorTransform;

var mapBitmap:BitmapData = createGradientBitmap(300, 80, 0xFF000000,
  "radial", true);

var filter_1:DisplacementMapFilter = new DisplacementMapFilter(mapBitmap,
  new Point(-30, -30), 1, 1, 10, 10, "wrap", 0x000000, 0x000000);
var filter_2:DisplacementMapFilter = filter_1;
var clonedFilter:DisplacementMapFilter = filter_1.clone();

trace(filter_1.mode); // wrap
trace(filter_2.mode); // wrap
trace(clonedFilter.mode); // wrap

filter_1.mode = "ignore";

trace(filter_1.mode); // ignore
trace(filter_2.mode); // ignore
trace(clonedFilter.mode); // wrap

function createGradientBitmap(w:Number, h:Number, bgColor:Number,
  type:String, hide:Boolean):BitmapData {
  var mc:MovieClip = this.createEmptyMovieClip("mc", 1);
  var matrix:Matrix = new Matrix();
  matrix.createGradientBox(w, h, 0, 0, 0);

  mc.beginGradientFill(type, [0xFF0000, 0x0000FF], [100, 100], [0x55,
  0x99], matrix, "pad");
  mc.lineTo(w, 0);
  mc.lineTo(w, h);
  mc.lineTo(0, h);
  mc.lineTo(0, 0);
  mc.endFill();
  (hide == true) ? mc._alpha = 0 : mc._alpha = 100;
```

```
  var bmp:BitmapData = new BitmapData(w, h, true, bgColor);
    bmp.draw(mc, new Matrix(), new ColorTransform(), "normal",
  bmp.rectangle, true);
  mc.attachBitmap(bmp, this.getNextHighestDepth());

  return bmp;
}
```

# color (propriété DisplacementMapFilter.color)

```
public color : Number
```

Spécifie la couleur à utiliser pour les déplacements en dehors des limites. La plage valide de déplacements est comprise entre 0,0 et 1,0. Les valeurs sont au format hexadécimal. La valeur par défaut de `color` est 0. Utilisez cette propriété si la propriété `mode` est définie sur `3`, `COLOR`.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

## Exemple

L'exemple suivant modifie la propriété hors limites `color` pour la mettre à `0x00FF00` sur le clip existant `filteredMc` quand un utilisateur clique sur celui-ci.

```
import flash.filters.DisplacementMapFilter;
import flash.display.BitmapData;
import flash.geom.Point;
import flash.geom.Matrix;
import flash.geom.ColorTransform;

var filteredMc:MovieClip = createDisplacementMapRectangle();

filteredMc.onPress = function() {
  var filter:DisplacementMapFilter = this.filters[0];
  filter.scaleY = 25;
  filter.mode = "color";
  filter.alpha = .25;
  filter.color = 0x00FF00;
  this.filters = new Array(filter);
}

function createDisplacementMapRectangle():MovieClip {
  var mapBitmap:BitmapData = createGradientBitmap(300, 80, 0xFF000000,
  "radial");
  var filter:DisplacementMapFilter = new DisplacementMapFilter(mapBitmap,
  new Point(-30, -30), 1, 1, 10, 10, "wrap", 0x000000, 0x000000);

  var txtBlock:MovieClip = createTextBlock();
  txtBlock._x = 30;
```

```
  txtBlock._y = 30;

  txtBlock.filters = new Array(filter);

  return txtBlock;
}

function createGradientBitmap(w:Number, h:Number, bgColor:Number,
  type:String, hide:Boolean):BitmapData {
  var mc:MovieClip = this.createEmptyMovieClip("mc", 1);
  var matrix:Matrix = new Matrix();
  matrix.createGradientBox(w, h, 0, 0, 0);

  mc.beginGradientFill(type, [0xFF0000, 0x0000FF], [100, 100], [0x55,
  0x99], matrix, "pad");
  mc.lineTo(w, 0);
  mc.lineTo(w, h);
  mc.lineTo(0, h);
  mc.lineTo(0, 0);
  mc.endFill();
  (hide == true) ? mc._alpha = 0 : mc._alpha = 100;

  var bmp:BitmapData = new BitmapData(w, h, true, bgColor);
    bmp.draw(mc, new Matrix(), new ColorTransform(), "normal",
  bmp.rectangle, true);
  mc.attachBitmap(bmp, this.getNextHighestDepth());

  return bmp;
}

function createTextBlock():MovieClip {
  var txtBlock:MovieClip = this.createEmptyMovieClip("txtBlock",
  this.getNextHighestDepth());
  txtBlock.createTextField("txt", this.getNextHighestDepth(), 0, 0, 300,
  80);
  txtBlock.txt.text = "watch the text bend with the displacement map";
  return txtBlock;
}
```

## componentX (propriété DisplacementMapFilter.componentX)

```
public componentX : Number
```

Décrit le canal de couleur à utiliser dans l'image de mappage pour déplacer le résultat *x*. Les valeurs possibles sont 1 (rouge), 2 (vert), 4 (bleu) et 8 (alpha).

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

## Exemple

L'exemple suivant change la propriété `componentX` sur le clip existant `filteredMc` quand un utilisateur clique dessus. La valeur passe de 1 à 4, ce qui change le canal de couleur de rouge à bleu.

```
import flash.filters.DisplacementMapFilter;
import flash.display.BitmapData;
import flash.geom.Point;
import flash.geom.Matrix;
import flash.geom.ColorTransform;

var filteredMc:MovieClip = createDisplacementMapRectangle();

filteredMc.onPress = function() {
  var filter:DisplacementMapFilter = this.filters[0];
  filter.componentX = 4;
  this.filters = new Array(filter);
}


function createDisplacementMapRectangle():MovieClip {
  var mapBitmap:BitmapData = createGradientBitmap(300, 80, 0xFF000000,
  "radial");
  var filter:DisplacementMapFilter = new DisplacementMapFilter(mapBitmap,
  new Point(-30, -30), 1, 1, 10, 10, "wrap", 0x000000, 0x000000);

  var txtBlock:MovieClip = createTextBlock();
  txtBlock._x = 30;
  txtBlock._y = 30;

  txtBlock.filters = new Array(filter);

  return txtBlock;
}

function createGradientBitmap(w:Number, h:Number, bgColor:Number,
  type:String, hide:Boolean):BitmapData {
  var mc:MovieClip = this.createEmptyMovieClip("mc", 1);
  var matrix:Matrix = new Matrix();
  matrix.createGradientBox(w, h, 0, 0, 0);

  mc.beginGradientFill(type, [0xFF0000, 0x0000FF], [100, 100], [0x55,
  0x99], matrix, "pad");
  mc.lineTo(w, 0);
  mc.lineTo(w, h);
  mc.lineTo(0, h);
  mc.lineTo(0, 0);
  mc.endFill();
  (hide == true) ? mc._alpha = 0 : mc._alpha = 100;
```

```
  var bmp:BitmapData = new BitmapData(w, h, true, bgColor);
    bmp.draw(mc, new Matrix(), new ColorTransform(), "normal",
  bmp.rectangle, true);
  mc.attachBitmap(bmp, this.getNextHighestDepth());

  return bmp;
}

function createTextBlock():MovieClip {
  var txtBlock:MovieClip = this.createEmptyMovieClip("txtBlock",
  this.getNextHighestDepth());
  txtBlock.createTextField("txt", this.getNextHighestDepth(), 0, 0, 300,
  80);
  txtBlock.txt.text = "watch the text bend with the displacement map";
  return txtBlock;
}
```

### Voir également

`BitmapData (flash.display.BitmapData)`

## componentY (propriété DisplacementMapFilter.componentY)

```
public componentY : Number
```

Décrit le canal de couleur à utiliser dans l'image de mappage pour déplacer le résultat *y*. Les valeurs possibles sont 1 (rouge), 2 (vert), 4 (bleu) et 8 (alpha).

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

### Exemple

L'exemple suivant change la propriété `componentY` sur le clip existant `filteredMc` quand un utilisateur clique dessus. La valeur passe de 1 à 4, ce qui change le canal de couleur de rouge à bleu.

```
import flash.filters.DisplacementMapFilter;
import flash.display.BitmapData;
import flash.geom.Point;
import flash.geom.Matrix;
import flash.geom.ColorTransform;

var filteredMc:MovieClip = createDisplacementMapRectangle();

filteredMc.onPress = function() {
  var filter:DisplacementMapFilter = this.filters[0];
  filter.componentY = 4;
```

```
  this.filters = new Array(filter);
}

function createDisplacementMapRectangle():MovieClip {
  var mapBitmap:BitmapData = createGradientBitmap(300, 80, 0xFF000000,
  "radial");
  var filter:DisplacementMapFilter = new DisplacementMapFilter(mapBitmap,
  new Point(-30, -30), 1, 1, 10, 10, "wrap", 0x000000, 0x000000);

  var txtBlock:MovieClip = createTextBlock();
  txtBlock._x = 30;
  txtBlock._y = 30;

  txtBlock.filters = new Array(filter);

  return txtBlock;
}

function createGradientBitmap(w:Number, h:Number, bgColor:Number,
  type:String, hide:Boolean):BitmapData {
  var mc:MovieClip = this.createEmptyMovieClip("mc", 1);
  var matrix:Matrix = new Matrix();
  matrix.createGradientBox(w, h, 0, 0, 0);

  mc.beginGradientFill(type, [0xFF0000, 0x0000FF], [100, 100], [0x55,
  0x99], matrix, "pad");
  mc.lineTo(w, 0);
  mc.lineTo(w, h);
  mc.lineTo(0, h);
  mc.lineTo(0, 0);
  mc.endFill();
  (hide == true) ? mc._alpha = 0 : mc._alpha = 100;

  var bmp:BitmapData = new BitmapData(w, h, true, bgColor);
    bmp.draw(mc, new Matrix(), new ColorTransform(), "normal",
  bmp.rectangle, true);
  mc.attachBitmap(bmp, this.getNextHighestDepth());

  return bmp;
}

function createTextBlock():MovieClip {
  var txtBlock:MovieClip = this.createEmptyMovieClip("txtBlock",
  this.getNextHighestDepth());
  txtBlock.createTextField("txt", this.getNextHighestDepth(), 0, 0, 300,
  80);
  txtBlock.txt.text = "watch the text bend with the displacement map";
  return txtBlock;
}
```

### Voir également

`BitmapData (flash.display.BitmapData)`

## constructeur DisplacementMapFilter

```
public DisplacementMapFilter(mapBitmap:BitmapData, mapPoint:Point,
  componentX:Number, componentY:Number, scaleX:Number, scaleY:Number,
  [mode:String], [color:Number], [alpha:Number])
```

Initialise une occurrence DisplacementMapFilter avec les paramètres spécifiés.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

### Paramètres

*mapBitmap*`:flash.display.BitmapData` - Objet BitmapData contenant les données de mappage du déplacement.

*mapPoint*`:flash.geom.Point` - Valeur `flash.geom.Point` représentant le décalage du coin supérieur gauche du clip cible par rapport au coin supérieur gauche de l'image de mappage.

*componentX*`:Number` - Décrit le canal de couleur à utiliser dans l'image de mappage pour déplacer le résultat *x*. Les valeurs possibles sont les suivantes :

- `1` (rouge)
- `2` (vert)
- `4`(bleu)
- `8` (alpha)

*componentY*`:Number` - Décrit le canal de couleur à utiliser dans l'image de mappage pour déplacer le résultat *y*. Les valeurs possibles sont les suivantes :

- `1` (rouge)
- `2` (vert)
- `4`(bleu)
- `8` (alpha)

*scaleX*`:Number` - Multiplicateur à utiliser pour mettre à l'échelle le résultat du déplacement *x* à partir du calcul de mappage.

*scaleY*`:Number` - Multiplicateur à utiliser pour mettre à l'échelle le résultat du déplacement *y* à partir du calcul de mappage.

*mode*`:String` [facultatif] - Mode du filtre. Les valeurs possibles sont les suivantes :

- `"wrap"` - Ramène la valeur de déplacement à l'autre côté de l'image source.
- `"clamp"` - Corrige la valeur de déplacement en fonction du bord de l'image source.

- `"ignore"` - Si la valeur de déplacement est hors limites, ignore le déplacement et utilise le pixel source.
- `"color"` - Si la valeur de déplacement est en dehors de l'image, remplace une valeur de pixel composée des propriétés `alpha` et `color` du filtre.

*`color`*`:Number` [facultatif] - Spécifie la couleur à utiliser pour les déplacements en dehors des limites. La plage valide de déplacements est comprise entre 0,0 et 1,0. Utilisez ce paramètre si le `mode` est défini sur `"color"`.

*`alpha`*`:Number` [facultatif] - Spécifie la valeur alpha à utiliser pour les déplacements en dehors des limites. Elle est spécifiée en tant que valeur normalisée comprise entre 0,0 et 1,0. Par exemple, 0,25 définit une valeur de transparence de 25 %. La valeur par défaut est 1,0. Utilisez ce paramètre si le `mode` est défini sur `"color"`.

## Exemple

La fonction constructeur suivante crée une nouvelle occurrence du filtre :

```
myFilter = new flash.filters.DisplacementMapFilter (mapBitmap, mapPoint,
  componentX, componentY, scale, [mode], [color], [alpha])
```

L'exemple suivant instancie un nouveau `DisplacementMapFilter` ayant une bitmap avec dégradé radial et l'applique au texte contenant l'objet MovieClip, `txtBlock`.

```
import flash.filters.DisplacementMapFilter;
import flash.display.BitmapData;
import flash.geom.Point;
import flash.geom.Matrix;
import flash.geom.ColorTransform;

var mapBitmap:BitmapData = createGradientBitmap(300, 80, 0xFF000000,
  "radial");

var mapPoint:Point = new Point(-30, -30);
var componentX:Number = 1;
var componentY:Number = 1;
var scaleX:Number = 10;
var scaleY:Number = 10;
var mode:String = "wrap";
var color:Number = 0x000000;
var alpha:Number = 0x000000;

var filter:DisplacementMapFilter = new DisplacementMapFilter(mapBitmap,
  mapPoint, componentX, componentY, scaleX, scaleY, mode, color, alpha);

var txtBlock:MovieClip = createTextBlock();
txtBlock._x = 30;
txtBlock._y = 30;

txtBlock.filters = new Array(filter);
```

```
function createGradientBitmap(w:Number, h:Number, bgColor:Number,
  type:String, hide:Boolean):BitmapData {
  var mc:MovieClip = this.createEmptyMovieClip("mc", 1);
  var matrix:Matrix = new Matrix();
  matrix.createGradientBox(w, h, 0, 0, 0);

  mc.beginGradientFill(type, [0xFF0000, 0x0000FF], [100, 100], [0x55,
  0x99], matrix, "pad");
  mc.lineTo(w, 0);
  mc.lineTo(w, h);
  mc.lineTo(0, h);
  mc.lineTo(0, 0);
  mc.endFill();
  (hide == true) ? mc._alpha = 0 : mc._alpha = 100;

  var bmp:BitmapData = new BitmapData(w, h, true, bgColor);
    bmp.draw(mc, new Matrix(), new ColorTransform(), "normal",
  bmp.rectangle, true);
  mc.attachBitmap(bmp, this.getNextHighestDepth());

  return bmp;
}

function createTextBlock():MovieClip {
  var txtBlock:MovieClip = this.createEmptyMovieClip("txtBlock",
  this.getNextHighestDepth());
    txtBlock.createTextField("txt", this.getNextHighestDepth(), 0, 0, 300,
  80);
  txtBlock.txt.text = "watch the text bend with the displacement map";
  return txtBlock;
}
```

## mapBitmap (propriété DisplacementMapFilter.mapBitmap)

```
public mapBitmap : BitmapData
```

Un objet BitmapData contenant les données de mappage du déplacement.

La propriété `mapBitmap` ne peut pas être changée en modifiant directement sa valeur. Vous devez plutôt acquérir une référence à `mapBitmap`, effectuer le changement de référence et ensuite définir `mapBitmap` par rapport à la référence.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

## Exemple

L'exemple suivant change la propriété `mapBitmap` sur le clip existant `filteredMc` quand un utilisateur clique dessus.

```
import flash.filters.DisplacementMapFilter;
import flash.display.BitmapData;
import flash.geom.Point;
import flash.geom.Matrix;
import flash.geom.ColorTransform;

var filteredMc:MovieClip = createDisplacementMapRectangle();
var scope:Object = this;

filteredMc.onPress = function() {
  var filter:DisplacementMapFilter = this.filters[0];
  filter.mapBitmap = scope.createGradientBitmap(300, 80, 0xFF000000,
  "linear");
  this.filters = new Array(filter);
}

function createDisplacementMapRectangle():MovieClip {
  var mapBitmap:BitmapData = createGradientBitmap(300, 80, 0xFF000000,
  "radial");
  var filter:DisplacementMapFilter = new DisplacementMapFilter(mapBitmap,
  new Point(-30, -30), 1, 1, 10, 10, "wrap", 0x000000, 0x000000);

  var txtBlock:MovieClip = createTextBlock();
  txtBlock._x = 30;
  txtBlock._y = 30;

  txtBlock.filters = new Array(filter);

  return txtBlock;
}

function createGradientBitmap(w:Number, h:Number, bgColor:Number,
  type:String, hide:Boolean):BitmapData {
  var mc:MovieClip = this.createEmptyMovieClip("mc", 1);
  var matrix:Matrix = new Matrix();
  matrix.createGradientBox(w, h, 0, 0, 0);

  mc.beginGradientFill(type, [0xFF0000, 0x0000FF], [100, 100], [0x55,
  0x99], matrix, "pad");
  mc.lineTo(w, 0);
  mc.lineTo(w, h);
  mc.lineTo(0, h);
  mc.lineTo(0, 0);
  mc.endFill();
  (hide == true) ? mc._alpha = 0 : mc._alpha = 100;
```

```
  var bmp:BitmapData = new BitmapData(w, h, true, bgColor);
    bmp.draw(mc, new Matrix(), new ColorTransform(), "normal",
  bmp.rectangle, true);
  mc.attachBitmap(bmp, this.getNextHighestDepth());

  return bmp;
}

function createTextBlock():MovieClip {
  var txtBlock:MovieClip = this.createEmptyMovieClip("txtBlock",
  this.getNextHighestDepth());
  txtBlock.createTextField("txt", this.getNextHighestDepth(), 0, 0, 300,
  80);
  txtBlock.txt.text = "watch the text bend with the displacement map";
  return txtBlock;
}
```

### Voir également

`BitmapData (flash.display.BitmapData)`

## mapPoint (propriété DisplacementMapFilter.mapPoint)

```
public mapPoint : Point
```

Une valeur `flash.geom.Point` représentant le décalage du coin supérieur gauche du clip cible par rapport au coin supérieur gauche de l'image de mappage.

La propriété `mapPoint` ne peut pas être changée en modifiant directement sa valeur. Vous devez plutôt acquérir une référence à `mapPoint`, effectuer le changement de référence et ensuite définir `mapPoint` par rapport à la référence.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

### Exemple

L'exemple suivant change la propriété `mapPoint` sur le clip existant `filteredMc` quand un utilisateur clique dessus.

```
import flash.filters.DisplacementMapFilter;
import flash.display.BitmapData;
import flash.geom.Point;
import flash.geom.Matrix;
import flash.geom.ColorTransform;

var filteredMc:MovieClip = createDisplacementMapRectangle();

filteredMc.onPress = function() {
```

```
  var filter:DisplacementMapFilter = this.filters[0];
  filter.mapPoint = new Point(-30, -40);
  this.filters = new Array(filter);
  this._x = 30;
  this._y = 40;
}

function createDisplacementMapRectangle():MovieClip {
  var mapBitmap:BitmapData = createGradientBitmap(300, 80, 0xFF000000,
  "radial");
  var filter:DisplacementMapFilter = new DisplacementMapFilter(mapBitmap,
  new Point(-30, -30), 1, 1, 10, 10, "wrap", 0x000000, 0x000000);

  var txtBlock:MovieClip = createTextBlock();
  txtBlock._x = 30;
  txtBlock._y = 30;

  txtBlock.filters = new Array(filter);

  return txtBlock;
}

function createGradientBitmap(w:Number, h:Number, bgColor:Number,
  type:String, hide:Boolean):BitmapData {
  var mc:MovieClip = this.createEmptyMovieClip("mc", 1);
  var matrix:Matrix = new Matrix();
  matrix.createGradientBox(w, h, 0, 0, 0);

  mc.beginGradientFill(type, [0xFF0000, 0x0000FF], [100, 100], [0x55,
  0x99], matrix, "pad");
  mc.lineTo(w, 0);
  mc.lineTo(w, h);
  mc.lineTo(0, h);
  mc.lineTo(0, 0);
  mc.endFill();
  (hide == true) ? mc._alpha = 0 : mc._alpha = 100;

  var bmp:BitmapData = new BitmapData(w, h, true, bgColor);
    bmp.draw(mc, new Matrix(), new ColorTransform(), "normal",
  bmp.rectangle, true);
  mc.attachBitmap(bmp, this.getNextHighestDepth());

  return bmp;
}

function createTextBlock():MovieClip {
  var txtBlock:MovieClip = this.createEmptyMovieClip("txtBlock",
  this.getNextHighestDepth());
  txtBlock.createTextField("txt", this.getNextHighestDepth(), 0, 0, 300,
  80);
```

```
  txtBlock.txt.text = "watch the text bend with the displacement map";
  return txtBlock;
}
```

### Voir également

`Point (flash.geom.Point)`

## mode (propriété DisplacementMapFilter.mode)

```
public mode : String
```

Le mode du filtre. Les valeurs possibles sont les suivantes :

- `"wrap"` - Ramène la valeur de déplacement à l'autre côté de l'image source. Il s'agit de la valeur par défaut.
- `"clamp"` - Corrige la valeur de déplacement en fonction du bord de l'image source.
- `"ignore"` - Si la valeur de déplacement est hors limites, ignore le déplacement et utilise le pixel source.
- `"color"` - Si la valeur de déplacement est en dehors de l'image, remplace une valeur de pixel composée des propriétés `alpha` et `color` du filtre.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

### Exemple

L'exemple suivant modifie `scaleY` pour créer une valeur de déplacement hors limites et ensuite change la propriété `mode` sur le clip existant `filteredMc` pour la définir à `ignore` quand un utilisateur clique sur celui-ci.

```
import flash.filters.DisplacementMapFilter;
import flash.display.BitmapData;
import flash.geom.Point;
import flash.geom.Matrix;
import flash.geom.ColorTransform;

var filteredMc:MovieClip = createDisplacementMapRectangle();

filteredMc.onPress = function() {
  var filter:DisplacementMapFilter = this.filters[0];
  filter.scaleY = 25;
  filter.mode = "ignore";
  this.filters = new Array(filter);
}

function createDisplacementMapRectangle():MovieClip {
  var mapBitmap:BitmapData = createGradientBitmap(300, 80, 0xFF000000,
  "radial");
```

```
  var filter:DisplacementMapFilter = new DisplacementMapFilter(mapBitmap,
  new Point(-30, -30), 1, 1, 10, 10, "wrap", 0x000000, 0x000000);

  var txtBlock:MovieClip = createTextBlock();
  txtBlock._x = 30;
  txtBlock._y = 30;

  txtBlock.filters = new Array(filter);

  return txtBlock;
}

function createGradientBitmap(w:Number, h:Number, bgColor:Number,
  type:String, hide:Boolean):BitmapData {
  var mc:MovieClip = this.createEmptyMovieClip("mc", 1);
  var matrix:Matrix = new Matrix();
  matrix.createGradientBox(w, h, 0, 0, 0);

  mc.beginGradientFill(type, [0xFF0000, 0x0000FF], [100, 100], [0x55,
  0x99], matrix, "pad");
  mc.lineTo(w, 0);
  mc.lineTo(w, h);
  mc.lineTo(0, h);
  mc.lineTo(0, 0);
  mc.endFill();
  (hide == true) ? mc._alpha = 0 : mc._alpha = 100;

  var bmp:BitmapData = new BitmapData(w, h, true, bgColor);
    bmp.draw(mc, new Matrix(), new ColorTransform(), "normal",
  bmp.rectangle, true);
  mc.attachBitmap(bmp, this.getNextHighestDepth());

  return bmp;
}

function createTextBlock():MovieClip {
  var txtBlock:MovieClip = this.createEmptyMovieClip("txtBlock",
  this.getNextHighestDepth());
  txtBlock.createTextField("txt", this.getNextHighestDepth(), 0, 0, 300,
  80);
  txtBlock.txt.text = "watch the text bend with the displacement map";
  return txtBlock;
}
```

# scaleX (propriété DisplacementMapFilter.scaleX)

```
public scaleX : Number
```

Le multiplicateur à utiliser pour mettre à l'échelle le résultat du déplacement *x* à partir du calcul de mappage.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

## Exemple

L'exemple suivant change la propriété `scaleX` sur le clip existant `filteredMc` quand un utilisateur clique dessus.

```
import flash.filters.DisplacementMapFilter;
import flash.display.BitmapData;
import flash.geom.Point;
import flash.geom.Matrix;
import flash.geom.ColorTransform;

var filteredMc:MovieClip = createDisplacementMapRectangle();

filteredMc.onPress = function() {
  var filter:DisplacementMapFilter = this.filters[0];
  filter.scaleX = 5;
  this.filters = new Array(filter);
}

function createDisplacementMapRectangle():MovieClip {
  var mapBitmap:BitmapData = createGradientBitmap(300, 80, 0xFF000000,
  "radial");
  var filter:DisplacementMapFilter = new DisplacementMapFilter(mapBitmap,
  new Point(-30, -30), 1, 1, 10, 10, "wrap", 0x000000, 0x000000);

  var txtBlock:MovieClip = createTextBlock();
  txtBlock._x = 30;
  txtBlock._y = 30;

  txtBlock.filters = new Array(filter);

  return txtBlock;
}

function createGradientBitmap(w:Number, h:Number, bgColor:Number,
  type:String, hide:Boolean):BitmapData {
  var mc:MovieClip = this.createEmptyMovieClip("mc", 1);
  var matrix:Matrix = new Matrix();
  matrix.createGradientBox(w, h, 0, 0, 0);

  mc.beginGradientFill(type, [0xFF0000, 0x0000FF], [100, 100], [0x55,
  0x99], matrix, "pad");
  mc.lineTo(w, 0);
  mc.lineTo(w, h);
  mc.lineTo(0, h);
  mc.lineTo(0, 0);
  mc.endFill();
  (hide == true) ? mc._alpha = 0 : mc._alpha = 100;
```

```
  var bmp:BitmapData = new BitmapData(w, h, true, bgColor);
    bmp.draw(mc, new Matrix(), new ColorTransform(), "normal",
  bmp.rectangle, true);
  mc.attachBitmap(bmp, this.getNextHighestDepth());

  return bmp;
}

function createTextBlock():MovieClip {
  var txtBlock:MovieClip = this.createEmptyMovieClip("txtBlock",
  this.getNextHighestDepth());
  txtBlock.createTextField("txt", this.getNextHighestDepth(), 0, 0, 300,
  80);
  txtBlock.txt.text = "watch the text bend with the displacement map";
  return txtBlock;
}
```

# scaleY (propriété DisplacementMapFilter.scaleY)

```
public scaleY : Number
```

Le multiplicateur à utiliser pour mettre à l'échelle le résultat du déplacement *y* à partir du calcul de mappage.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

## Exemple

L'exemple suivant change la propriété `scaleY` sur le clip existant `filteredMc` quand un utilisateur clique dessus.

```
import flash.filters.DisplacementMapFilter;
import flash.display.BitmapData;
import flash.geom.Point;
import flash.geom.Matrix;
import flash.geom.ColorTransform;

var filteredMc:MovieClip = createDisplacementMapRectangle();

filteredMc.onPress = function() {
  var filter:DisplacementMapFilter = this.filters[0];
  filter.scaleY = 5;
  this.filters = new Array(filter);
}

function createDisplacementMapRectangle():MovieClip {
  var mapBitmap:BitmapData = createGradientBitmap(300, 80, 0xFF000000,
  "radial");
```

```
  var filter:DisplacementMapFilter = new DisplacementMapFilter(mapBitmap,
  new Point(-30, -30), 1, 1, 10, 10, "wrap", 0x000000, 0x000000);

  var txtBlock:MovieClip = createTextBlock();
  txtBlock._x = 30;
  txtBlock._y = 30;

  txtBlock.filters = new Array(filter);

  return txtBlock;
}

function createGradientBitmap(w:Number, h:Number, bgColor:Number,
  type:String, hide:Boolean):BitmapData {
  var mc:MovieClip = this.createEmptyMovieClip("mc", 1);
  var matrix:Matrix = new Matrix();
  matrix.createGradientBox(w, h, 0, 0, 0);

  mc.beginGradientFill(type, [0xFF0000, 0x0000FF], [100, 100], [0x55,
  0x99], matrix, "pad");
  mc.lineTo(w, 0);
  mc.lineTo(w, h);
  mc.lineTo(0, h);
  mc.lineTo(0, 0);
  mc.endFill();
  (hide == true) ? mc._alpha = 0 : mc._alpha = 100;

  var bmp:BitmapData = new BitmapData(w, h, true, bgColor);
    bmp.draw(mc, new Matrix(), new ColorTransform(), "normal",
  bmp.rectangle, true);
  mc.attachBitmap(bmp, this.getNextHighestDepth());

  return bmp;
}

function createTextBlock():MovieClip {
  var txtBlock:MovieClip = this.createEmptyMovieClip("txtBlock",
  this.getNextHighestDepth());
  txtBlock.createTextField("txt", this.getNextHighestDepth(), 0, 0, 300,
  80);
  txtBlock.txt.text = "watch the text bend with the displacement map";
  return txtBlock;
}
```

# DropShadowFilter (flash.filters.DropShadowFilter)

```
Object
  |
  +-flash.filters.BitmapFilter
    |
    +-flash.filters.DropShadowFilter
```

```
public class DropShadowFilter
extends BitmapFilter
```

La classe DropShadowFilter vous permet d'ajouter une ombre portée à divers objets dans Flash. Vous disposez de plusieurs options pour définir le style de l'ombre portée, notamment l'ombre intérieure ou extérieure et le mode de masquage.

L'utilisation de filtres dépend de l'objet auquel vous appliquez le filtre.

- Pour appliquer des filtres aux clips, champs de texte et boutons lors de l'exécution, utilisez la propriété `filters`. Lorsque vous définissez la propriété `filters` d'un objet, celui-ci n'est pas modifié. En outre, vous pouvez l'annuler en supprimant la propriété `filters`.
- Pour appliquer des filtres aux occurrences BitmapData, utilisez la méthode `BitmapData.applyFilter()`. L'appel d'`applyFilter()` sur un objet BitmapData utilise l'objet BitmapData source ainsi que l'objet filtre pour générer une image filtrée.

Vous pouvez également appliquer des effets de filtre aux images et aux données vidéo pendant la programmation. Pour plus d'informations, consultez la documentation relative à la programmation.

Si vous appliquez un filtre à un clip ou à un bouton, la propriété `cacheAsBitmap` du clip ou du bouton est définie sur `true`. Si vous supprimez tous les filtres, la valeur d'origine de `cacheAsBitmap` est restaurée.

Ce filtre prend en charge le redimensionnement de la scène. Mais le redimensionnement général, la rotation et l'inclinaison ne sont pas gérés. Si l'objet lui-même est redimensionné (si `_xscale` et `_yscale` ne sont pas à 100%), l'effet de filtre n'est pas redimensionné. Le redimensionnement est effectué uniquement en cas de zoom avant sur la scène.

Un filtre ne peut s'appliquer si l'image résultante dépasse 2 880 pixels en largeur ou en hauteur. Par exemple, si vous faites un zoom avant sur un grand clip auquel un filtre est appliqué, le filtre est désactivé si l'image résultante dépasse la limite de 2 880 pixels.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

## Voir également

filters (propriété MovieClip.filters), cacheAsBitmap (propriété MovieClip.cacheAsBitmap), filters (propriété Button.filters), cacheAsBitmap (propriété Button.cacheAsBitmap), filters (propriété TextField.filters), applyFilter (méthode BitmapData.applyFilter)

## Résumé des propriétés

| Modificateurs | Propriété | Description |
|---|---|---|
| | alpha:Number | La valeur de transparence alpha de la couleur d'ombre. |
| | angle:Number | L'angle de l'ombre. |
| | blurX:Number | Le montant de flou horizontal. |
| | blurY:Number | Le montant de flou vertical. |
| | color:Number | La couleur de l'ombre. |
| | distance:Number | La distance de décalage de l'ombre en pixels. |
| | hideObject:Boolean | Indique si l'objet est masqué ou non. |
| | inner:Boolean | Indique si l'ombre est intérieure ou non. |
| | knockout:Boolean | Applique un effet de poinçonnage (true) qui rend le remplissage de l'objet effectivement transparent et révèle la couleur d'arrière-plan du document. |
| | quality:Number | Le nombre de fois où le filtre doit s'appliquer. |
| | strength:Number | L'intensité de l'impression ou du recouvrement. |

*Propriétés héritées de la classe Object*

constructeur (propriété Object.constructor), __proto__ (Object.__proto__ property), prototype (Object.prototype, propriété), __resolve (Object.__resolve, propriété)

### Récapitulatif des constructeurs

| Signature | Description |
|---|---|
| `DropShadowFilter([distance:Number], [angle:Number], [color:Number], [alpha:Number], [blurX:Number], [blurY:Number], [strength:Number], [quality:Number], [inner:Boolean], [knockout:Boolean], [hideObject:Boolean])` | Crée une nouvelle occurrence DropShadowFilter avec les paramètres spécifiés. |

### Résumé de la méthode

| Modificateurs | Signature | Description |
|---|---|---|
| | `clone() : DropShadowFilter` | Renvoie une copie de cet objet filtre. |

*Méthodes héritées de la classe BitmapFilter*

`clone (méthode BitmapFilter.clone )`

*Méthodes héritées de la classe Object*

`addProperty (méthode Object.addProperty), hasOwnProperty (méthode Object.hasOwnProperty), isPropertyEnumerable (méthode Object.isPropertyEnumerable), isPrototypeOf (méthode Object.isPrototypeOf), registerClass (méthode Object.registerClass), toString (méthode Object.toString), unwatch (méthode Object.unwatch), valueOf (méthode Object.valueOf), watch (méthode Object.watch)`

## alpha (propriété DropShadowFilter.alpha)

`public alpha : Number`

La valeur de transparence alpha de la couleur d'ombre. Les valeurs valides vont de 0 à 1. Par exemple, 0,25 définit la valeur de transparence à 25 %. La valeur par défaut est 1.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

### Exemple

L'exemple suivant change la propriété `alpha` sur un clip quand un utilisateur clique dessus.

```
import flash.filters.DropShadowFilter;
var mc:MovieClip = createDropShadowRectangle("DropShadowAlpha");
mc.onRelease = function() {
  var filter:DropShadowFilter = this.filters[0];
  filter.alpha = .4;
  this.filters = new Array(filter);
}

function createDropShadowRectangle(name:String):MovieClip {
  var art:MovieClip = this.createEmptyMovieClip(name,
  this.getNextHighestDepth());
  var w:Number = 100;
  var h:Number = 100;
  art.beginFill(0x003366);
  art.lineTo(w, 0);
  art.lineTo(w, h);
  art.lineTo(0, h);
  art.lineTo(0, 0);
  art._x = 20;
  art._y = 20;

  var filter:DropShadowFilter = new DropShadowFilter(15, 45, 0x000000, .8,
  16, 16, 1, 3, false, false, false);
  var filterArray:Array = new Array();
  filterArray.push(filter);
  art.filters = filterArray;
  return art;
}
```

## angle (propriété DropShadowFilter.angle)

```
public angle : Number
```

L'angle de l'ombre. Les valeurs valides sont comprises entre 0 et 360° (virgule flottante). La valeur par défaut est 45.

La valeur de l'angle représente l'angle de la source de lumière théorique projetée sur l'objet et détermine le placement de l'effet relatif à l'objet. Si la distance est définie à 0, l'effet n'est pas décalé de l'objet, et donc la propriété angle n'a aucun effet.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

### Exemple

L'exemple suivant change la propriété `angle` sur une occurrence de clip existant quand un utilisateur clique dessus.

```
import flash.filters.DropShadowFilter;
var mc:MovieClip = createDropShadowRectangle("DropShadowAngle");
mc.onRelease = function() {
  var filter:DropShadowFilter = this.filters[0];
  filter.angle = 135;
  this.filters = new Array(filter);
}

function createDropShadowRectangle(name:String):MovieClip {
  var art:MovieClip = this.createEmptyMovieClip(name,
  this.getNextHighestDepth());
  var w:Number = 100;
  var h:Number = 100;
  art.beginFill(0x003366);
  art.lineTo(w, 0);
  art.lineTo(w, h);
  art.lineTo(0, h);
  art.lineTo(0, 0);
  art._x = 20;
  art._y = 20;

  var filter:DropShadowFilter = new DropShadowFilter(15, 45, 0x000000, .8,
  16, 16, 1, 3, false, false, false);
  var filterArray:Array = new Array();
  filterArray.push(filter);
  art.filters = filterArray;
  return art;
}
```

# blurX (propriété DropShadowFilter.blurX)

```
public blurX : Number
```

Le montant de flou horizontal. Les valeurs valides sont comprises entre 0 et 255 (virgule flottante). La valeur par défaut est 4. Les valeurs multiples de 2 (telles que 2, 4, 8, 16 et 32) sont optimisées pour donner un rendu plus rapide que les autres valeurs.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

## Exemple

L'exemple suivant change la propriété `blurX` sur une occurrence de clip existant quand un utilisateur clique dessus.

```
import flash.filters.DropShadowFilter;
var mc:MovieClip = createDropShadowRectangle("DropShadowBlurX");
mc.onRelease = function() {
  var filter:DropShadowFilter = this.filters[0];
  filter.blurX = 40;
  this.filters = new Array(filter);
```

```
}

function createDropShadowRectangle(name:String):MovieClip {
  var art:MovieClip = this.createEmptyMovieClip(name,
  this.getNextHighestDepth());
  var w:Number = 100;
  var h:Number = 100;
  art.beginFill(0x003366);
  art.lineTo(w, 0);
  art.lineTo(w, h);
  art.lineTo(0, h);
  art.lineTo(0, 0);
  art._x = 20;
  art._y = 20;

  var filter:DropShadowFilter = new DropShadowFilter(15, 45, 0x000000, .8,
  16, 16, 1, 3, false, false, false);
  var filterArray:Array = new Array();
  filterArray.push(filter);
  art.filters = filterArray;
  return art;
}
```

# blurY (propriété DropShadowFilter.blurY)

```
public blurY : Number
```

Le montant de flou vertical. Les valeurs valides sont comprises entre 0 et 255 (virgule flottante). La valeur par défaut est 4. Les valeurs multiples de 2 (telles que 2, 4, 8, 16 et 32) sont optimisées pour donner un rendu plus rapide que les autres valeurs.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

## Exemple

L'exemple suivant change la propriété `blurY` sur une occurrence de clip existant quand un utilisateur clique dessus.

```
import flash.filters.DropShadowFilter;
var mc:MovieClip = createDropShadowRectangle("DropShadowBlurY");
mc.onRelease = function() {
  var filter:DropShadowFilter = this.filters[0];
  filter.blurY = 40;
  this.filters = new Array(filter);
}

function createDropShadowRectangle(name:String):MovieClip {
  var art:MovieClip = this.createEmptyMovieClip(name,
  this.getNextHighestDepth());
  var w:Number = 100;
```

```
  var h:Number = 100;
  art.beginFill(0x003366);
  art.lineTo(w, 0);
  art.lineTo(w, h);
  art.lineTo(0, h);
  art.lineTo(0, 0);
  art._x = 20;
  art._y = 20;

  var filter:DropShadowFilter = new DropShadowFilter(15, 45, 0x000000, .8,
  16, 16, 1, 3, false, false, false);
  var filterArray:Array = new Array();
  filterArray.push(filter);
  art.filters = filterArray;
  return art;
}
```

## clone (méthode DropShadowFilter.clone)

```
public clone() : DropShadowFilter
```

Renvoie une copie de cet objet filtre.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

### Valeur renvoyée

`flash.filters.DropShadowFilter` - Nouvelle occurrence DropShadowFilter dont les propriétés sont toutes identiques à celles de l'occurrence d'origine.

### Exemple

L'exemple suivant crée trois objets DropShadowFilter et les compare; `filter_1` est créé par le constructeur DropShadowFilter ; `filter_2` est créé en le définissant comme égal à `filter_1` ; et `clonedFilter` est créé en clonant `filter_1`. Veuillez noter que `filter_2` est considéré comme égal à `filter_1`, `clonedFilter` ne l'est pas, même s'il contient les mêmes valeurs que `filter_1`.

```
import flash.filters.DropShadowFilter;

var filter_1:DropShadowFilter = new DropShadowFilter(15, 45, 0x000000, .8,
  16, 16, 1, 3, false, false, false);
var filter_2:DropShadowFilter = filter_1;
var clonedFilter:DropShadowFilter = filter_1.clone();

trace(filter_1 == filter_2); // true
trace(filter_1 == clonedFilter); // false

for(var i in filter_1) {
```

```
  trace(">> " + i + ": " + filter_1[i]);
  // >> clone: [type Function]
  // >> hideObject: false
  // >> strength: 1
  // >> blurY: 16
  // >> blurX: 16
  // >> knockout: false
  // >> inner: false
  // >> quality: 3
  // >> alpha: 0.8
  // >> color: 0
  // >> angle: 45
  // >> distance: 15
}

for(var i in clonedFilter) {
  trace(">> " + i + ": " + clonedFilter[i]);
  // >> clone: [type Function]
  // >> hideObject: false
  // >> strength: 1
  // >> blurY: 16
  // >> blurX: 16
  // >> knockout: false
  // >> inner: false
  // >> quality: 3
  // >> alpha: 0.8
  // >> color: 0
  // >> angle: 45
  // >> distance: 15
}
```

Pour démontrer davantage les relations entre `filter_1`, `filter_2`, et `clonedFilter`, l'exemple suivant modifie la propriété `knockout` de `filter_1`. La modification de `knockout` démontre que la méthode `clone()` crée une nouvelle occurrence basée sur les valeurs de `filter_1` au lieu de faire référence à ces valeurs.

```
import flash.filters.DropShadowFilter;

var filter_1:DropShadowFilter = new DropShadowFilter(15, 45, 0x000000, .8,
  16, 16, 1, 3, false, false, false);
var filter_2:DropShadowFilter = filter_1;
var clonedFilter:DropShadowFilter = filter_1.clone();

trace(filter_1.knockout); // false
trace(filter_2.knockout); // false
trace(clonedFilter.knockout); // false

filter_1.knockout = true;

trace(filter_1.knockout); // true
```

```
trace(filter_2.knockout); // true
trace(clonedFilter.knockout); // false
```

# color (propriété DropShadowFilter.color)

```
public color : Number
```

La couleur de l'ombre. Les valeurs valides sont au format hexadécimal *0xRRGGBB*. La valeur par défaut est 0x000000.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

## Exemple

L'exemple suivant change la propriété `color` sur une occurrence de clip existant quand un utilisateur clique dessus.

```
import flash.filters.DropShadowFilter;
var mc:MovieClip = createDropShadowRectangle("DropShadowColor");
mc.onRelease = function() {
  var filter:DropShadowFilter = this.filters[0];
  filter.color = 0xFF0000;
  this.filters = new Array(filter);
}

function createDropShadowRectangle(name:String):MovieClip {
  var art:MovieClip = this.createEmptyMovieClip(name,
  this.getNextHighestDepth());
  var w:Number = 100;
  var h:Number = 100;
  art.beginFill(0x003366);
  art.lineTo(w, 0);
  art.lineTo(w, h);
  art.lineTo(0, h);
  art.lineTo(0, 0);
  art._x = 20;
  art._y = 20;

  var filter:DropShadowFilter = new DropShadowFilter(15, 45, 0x000000, .8,
  16, 16, 1, 3, false, false, false);
  var filterArray:Array = new Array();
  filterArray.push(filter);
  art.filters = filterArray;
  return art;
}
```

# distance (propriété DropShadowFilter.distance)

```
public distance : Number
```

La distance de décalage de l'ombre en pixels. La valeur par défaut est 4 (virgule flottante).

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

## Exemple

L'exemple suivant change la propriété `distance` sur une occurrence de clip existant quand un utilisateur clique dessus.

```
import flash.filters.DropShadowFilter;
var mc:MovieClip = createDropShadowRectangle("DropShadowDistance");
mc.onRelease = function() {
  var filter:DropShadowFilter = this.filters[0];
  filter.distance = 40;
  this.filters = new Array(filter);
}

function createDropShadowRectangle(name:String):MovieClip {
  var art:MovieClip = this.createEmptyMovieClip(name,
  this.getNextHighestDepth());
  var w:Number = 100;
  var h:Number = 100;
  art.beginFill(0x003366);
  art.lineTo(w, 0);
  art.lineTo(w, h);
  art.lineTo(0, h);
  art.lineTo(0, 0);
  art._x = 20;
  art._y = 20;

  var filter:DropShadowFilter = new DropShadowFilter(15, 45, 0x000000, .8,
  16, 16, 1, 3, false, false, false);
  var filterArray:Array = new Array();
  filterArray.push(filter);
  art.filters = filterArray;
  return art;
}
```

# constructeur DropShadowFilter

```
public DropShadowFilter([distance:Number], [angle:Number], [color:Number],
  [alpha:Number], [blurX:Number], [blurY:Number], [strength:Number],
  [quality:Number], [inner:Boolean], [knockout:Boolean],
  [hideObject:Boolean])
```

Crée une nouvelle occurrence DropShadowFilter avec les paramètres spécifiés.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

## Paramètres

*distance*:Number [facultatif] - Distance de décalage de l'ombre en pixels. La valeur par défaut est 4 (virgule flottante).

*angle*:Number [facultatif] - Angle de l'ombre, de 0 à 360° (virgule flottante). La valeur par défaut est 45.

*color*:Number [facultatif] - Couleur de l'ombre, au format hexadécimal *0xRRGGBB*. La valeur par défaut est 0x000000.

*alpha*:Number [facultatif] - Valeur de transparence alpha de la couleur d'ombre. Les valeurs valides vont de 0 à 1. Par exemple, 0,25 définit la valeur de transparence à 25 %. La valeur par défaut est 1.

*blurX*:Number [facultatif] - Quantité de flou à appliquer horizontalement. Les valeurs valides sont comprises entre 0 et 255 (virgule flottante). La valeur par défaut est 4. Les valeurs multiples de 2 (telles que 2, 4, 8, 16 et 32) sont optimisées pour donner un rendu plus rapide que les autres valeurs.

*blurY*:Number [facultatif] - Quantité de flou à appliquer verticalement. Les valeurs valides sont comprises entre 0 et 255 (virgule flottante). La valeur par défaut est 4. Les valeurs multiples de 2 (telles que 2, 4, 8, 16 et 32) sont optimisées pour donner un rendu plus rapide que les autres valeurs.

*strength*:Number [facultatif] - Intensité de l'impression ou du recouvrement. Plus la valeur est élevée, plus l'intensité des couleurs apparaît à l'impression et plus le contraste est important entre l'ombre et l'arrière-plan. Les valeurs valides sont comprises entre 0 et 255. La valeur par défaut est 1.

*quality*:Number [facultatif] - Nombre de fois que le filtre doit s'appliquer. Les valeurs valides sont 0 à 15. La valeur par défaut est 1, ce qui équivaut à une qualité inférieure. Une valeur de 2 est une qualité moyenne et une valeur de 3 est la qualité supérieure.

*inner*:Boolean [facultatif] - Indique si l'ombre est intérieure ou non. Une valeur `true` spécifie une ombre intérieure. La valeur par défaut est `false`, renvoyant une ombre extérieure au niveau des bords extérieurs de l'objet.

*knockout*:Boolean [facultatif] - Applique un effet de poinçonnage (`true`) qui rend le remplissage de l'objet effectivement transparent et révèle la couleur d'arrière-plan du document. La valeur par défaut est `false` soit pas de poinçonnage.

*hideObject*:Boolean [facultatif] - Indique si l'objet est masqué ou non. Une valeur `true` indique que l'objet n'est pas tracé et que seule l'ombre est visible. La valeur par défaut est `false`, soit montrer l'objet.

## Exemple

L'exemple suivant instancie une nouvelle occurrence DropShadowFilter et l'applique à un rectangle plat.

```
import flash.filters.DropShadowFilter;
var art:MovieClip = createRectangle(100, 100, 0x003366,
  "gradientGlowFilterExample");
var distance:Number = 20;
var angleInDegrees:Number = 45;
var color:Number = 0x000000;
var alpha:Number = .8;
var blurX:Number = 16;
var blurY:Number = 16;
var strength:Number = 1;
var quality:Number = 3;
var inner:Boolean = false;
var knockout:Boolean = false;
var hideObject:Boolean = false;

var filter:DropShadowFilter = new DropShadowFilter(distance,
                              angleInDegrees,
                              color,
                              alpha,
                              blurX,
                              blurY,
                              strength,
                              quality,
                              inner,
                              knockout,
                              hideObject);
var filterArray:Array = new Array();
filterArray.push(filter);
art.filters = filterArray;

function createRectangle(w:Number, h:Number, bgColor:Number,
  name:String):MovieClip {
  var mc:MovieClip = this.createEmptyMovieClip(name,
  this.getNextHighestDepth());
  mc.beginFill(bgColor);
  mc.lineTo(w, 0);
  mc.lineTo(w, h);
  mc.lineTo(0, h);
  mc.lineTo(0, 0);
  mc._x = 20;
  mc._y = 20;
  return mc;
}
```

# hideObject (propriété DropShadowFilter.hideObject)

```
public hideObject : Boolean
```

Indique si l'objet est masqué ou non. Une valeur `true` indique que l'objet n'est pas tracé et que seule l'ombre est visible. La valeur par défaut est `false`, soit montrer l'objet.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

## Exemple

L'exemple suivant change la propriété `hideObject` sur un clip existant quand un utilisateur clique dessus.

```
import flash.filters.DropShadowFilter;
var mc:MovieClip = createDropShadowRectangle("DropShadowHideObject");
mc.onRelease = function() {
  var filter:DropShadowFilter = this.filters[0];
  filter.hideObject = true;
  this.filters = new Array(filter);
}

function createDropShadowRectangle(name:String):MovieClip {
  var art:MovieClip = this.createEmptyMovieClip(name,
  this.getNextHighestDepth());
  var w:Number = 100;
  var h:Number = 100;
  art.beginFill(0x003366);
  art.lineTo(w, 0);
  art.lineTo(w, h);
  art.lineTo(0, h);
  art.lineTo(0, 0);
  art._x = 20;
  art._y = 20;

  var filter:DropShadowFilter = new DropShadowFilter(15, 45, 0x000000, .8,
  16, 16, 1, 3, false, false, false);
  var filterArray:Array = new Array();
  filterArray.push(filter);
  art.filters = filterArray;
  return art;
}
```

# inner (propriété DropShadowFilter.inner)

```
public inner : Boolean
```

Indique si l'ombre est intérieure ou non. La valeur `true` indique une ombre intérieure. La valeur par défaut est `false`, renvoyant une ombre extérieure au niveau des bords extérieurs de l'objet.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

### Exemple

L'exemple suivant change la propriété `inner` sur un clip existant quand un utilisateur clique dessus.

```
import flash.filters.DropShadowFilter;
var mc:MovieClip = createDropShadowRectangle("DropShadowInner");
mc.onRelease = function() {
  var filter:DropShadowFilter = this.filters[0];
  filter.inner = true;
  this.filters = new Array(filter);
}

function createDropShadowRectangle(name:String):MovieClip {
  var art:MovieClip = this.createEmptyMovieClip(name,
  this.getNextHighestDepth());
  var w:Number = 100;
  var h:Number = 100;
  art.beginFill(0x003366);
  art.lineTo(w, 0);
  art.lineTo(w, h);
  art.lineTo(0, h);
  art.lineTo(0, 0);
  art._x = 20;
  art._y = 20;

  var filter:DropShadowFilter = new DropShadowFilter(15, 45, 0x000000, .8,
  16, 16, 1, 3, false, false, false);
  var filterArray:Array = new Array();
  filterArray.push(filter);
  art.filters = filterArray;
  return art;
}
```

## knockout (propriété DropShadowFilter.knockout)

```
public knockout : Boolean
```

Applique un effet de poinçonnage (`true`) qui rend le remplissage de l'objet effectivement transparent et révèle la couleur d'arrière-plan du document. La valeur par défaut est `false` soit pas de poinçonnage.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

### Exemple

L'exemple suivant change la propriété `knockout` sur un clip existant quand un utilisateur clique dessus.

```
import flash.filters.DropShadowFilter;
var mc:MovieClip = createDropShadowRectangle("DropShadowKnockout");
mc.onRelease = function() {
  var filter:DropShadowFilter = this.filters[0];
  filter.knockout = true;
  this.filters = new Array(filter);
}

function createDropShadowRectangle(name:String):MovieClip {
  var art:MovieClip = this.createEmptyMovieClip(name,
  this.getNextHighestDepth());
  var w:Number = 100;
  var h:Number = 100;
  art.beginFill(0x003366);
  art.lineTo(w, 0);
  art.lineTo(w, h);
  art.lineTo(0, h);
  art.lineTo(0, 0);
  art._x = 20;
  art._y = 20;

  var filter:DropShadowFilter = new DropShadowFilter(15, 45, 0x000000, .8,
  16, 16, 1, 3, false, false, false);
  var filterArray:Array = new Array();
  filterArray.push(filter);
  art.filters = filterArray;
  return art;
}
```

## quality (propriété DropShadowFilter.quality)

```
public quality : Number
```

Le nombre de fois où le filtre doit s'appliquer. Les valeurs valides sont 0 à 15. La valeur par défaut est 1, ce qui équivaut à une qualité inférieure. Une valeur de 2 est une qualité moyenne et une valeur de 3 est la qualité supérieure. Les rendus des filtres de valeurs inférieures sont obtenus plus rapidement.

Pour la plupart des applications, une valeur de `quality` de 1, 2, ou 3 est suffisante. Bien que vous puissiez utiliser les valeurs numériques supplémentaires jusqu'à 15 pour appliquer les différents effets, les valeurs les plus hautes donnent un rendu plus lent. Plutôt que d'augmenter la valeur de `quality`, vous pouvez souvent obtenir un effet similaire avec un rendu plus rapide en augmentant simplement les valeurs de `blurX` et `blurY`.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

### Exemple

L'exemple suivant change la propriété `quality` sur un clip existant quand un utilisateur clique dessus.

```
import flash.filters.DropShadowFilter;
var mc:MovieClip = createDropShadowRectangle("DropShadowQuality");
mc.onRelease = function() {
  var filter:DropShadowFilter = this.filters[0];
  filter.quality = 0;
  this.filters = new Array(filter);
}

function createDropShadowRectangle(name:String):MovieClip {
  var art:MovieClip = this.createEmptyMovieClip(name,
  this.getNextHighestDepth());
  var w:Number = 100;
  var h:Number = 100;
  art.beginFill(0x003366);
  art.lineTo(w, 0);
  art.lineTo(w, h);
  art.lineTo(0, h);
  art.lineTo(0, 0);
  art._x = 20;
  art._y = 20;

  var filter:DropShadowFilter = new DropShadowFilter(15, 45, 0x000000, .8,
  16, 16, 1, 3, false, false, false);
  var filterArray:Array = new Array();
  filterArray.push(filter);
  art.filters = filterArray;
  return art;
}
```

## strength (propriété DropShadowFilter.strength)

```
public strength : Number
```

L'intensité de l'impression ou du recouvrement. Plus la valeur est élevée, plus l'intensité des couleurs apparaît à l'impression et plus le contraste est important entre l'ombre et l'arrière-plan. Les valeurs valides sont comprises entre 0 et 255. La valeur par défaut est 1.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

### Exemple

L'exemple suivant change la propriété `strength` sur un clip existant quand un utilisateur clique dessus.

```
import flash.filters.DropShadowFilter;
var mc:MovieClip = createDropShadowRectangle("DropShadowStrength");
mc.onRelease = function() {
  var filter:DropShadowFilter = this.filters[0];
  filter.strength = .6;
  this.filters = new Array(filter);
}

function createDropShadowRectangle(name:String):MovieClip {
  var art:MovieClip = this.createEmptyMovieClip(name,
  this.getNextHighestDepth());
  var w:Number = 100;
  var h:Number = 100;
  art.beginFill(0x003366);
  art.lineTo(w, 0);
  art.lineTo(w, h);
  art.lineTo(0, h);
  art.lineTo(0, 0);
  art._x = 20;
  art._y = 20;

  var filter:DropShadowFilter = new DropShadowFilter(15, 45, 0x000000, .8,
  16, 16, 1, 3, false, false, false);
  var filterArray:Array = new Array();
  filterArray.push(filter);
  art.filters = filterArray;
  return art;
}
```

# Erreur

```
Object
  |
  +-Error
```

`public class` ***Error***
`extends Object`

Contient des informations sur une erreur qui s'est produite dans un script. Vous pouvez créer un objet Error à l'aide de la fonction constructeur `Error`. En général, vous générez (`throw`) un nouvel objet Error à partir d'un bloc de code `try`, qui est ensuite détecté par un bloc de code `catch` ou `finally`.

Vous pouvez également créer une sous-classe de la classe Error et générer des occurrences de cette sous-classe.

**Disponibilité :** ActionScript 1.0 ; Flash Player 7

## Résumé des propriétés

| Modificateurs | Propriété | Description |
|---|---|---|
| | `message:String` | Contient le message associé à l'objet Error. |
| | `name:String` | Contient le nom de l'objet Error. |

*Propriétés héritées de la classe Object*

`constructeur (propriété Object.constructor), __proto__ (Object.__proto__ property), prototype (Object.prototype, propriété), __resolve (Object.__resolve, propriété)`

## Récapitulatif des constructeurs

| Signature | Description |
|---|---|
| `Error([message:String])` | Crée un nouvel objet Error. |

## Résumé de la méthode

| Modificateurs | Signature | Description |
|---|---|---|
| | `toString() : String` | Renvoie la chaîne "Error" par défaut ou la valeur contenue dans Error.message, s'il est défini. |

*Méthodes héritées de la classe Object*

`addProperty (méthode Object.addProperty), hasOwnProperty (méthode Object.hasOwnProperty), isPropertyEnumerable (méthode Object.isPropertyEnumerable), isPrototypeOf (méthode Object.isPrototypeOf), registerClass (méthode Object.registerClass), toString (méthode Object.toString), unwatch (méthode Object.unwatch), valueOf (méthode Object.valueOf), watch (méthode Object.watch)`

# constructeur Error

`public Error([message:String])`

Crée un nouvel objet Error. Si *`message`* est spécifié, sa valeur est affectée à la propriété `Error.message` de l'objet.

**Disponibilité :** ActionScript 1.0 ; Flash Player 7

### Paramètres

*message*:String [facultatif] - Chaîne associée à l'objet Error.

### Exemple

Dans l'exemple suivant, une fonction renvoie une erreur (avec un message spécifié) si les deux chaînes qui lui sont transmises ne sont pas identiques :

```
function compareStrings(str1_str:String, str2_str:String):Void {
  if (str1_str != str2_str) {
  throw new Error("Strings do not match.");
  }
}
try {
  compareStrings("Dog", "dog");
  // output: Strings do not match.
} catch (e_err:Error) {
  trace(e_err.toString());
}
```

### Voir également

Instruction throw, Instruction try..catch..finally

## message (propriété Error.message)

```
public Message : String
```

Contient le message associé à l'objet Error. Par défaut, la valeur de cette propriété est "Error". Vous pouvez spécifier une propriété message lorsque vous créez un objet Error en transmettant la chaîne d'erreur à la fonction constructeur Error.

**Disponibilité :** ActionScript 1.0 ; Flash Player 7

### Exemple

Dans l'exemple suivant, une fonction renvoie un message spécifié en fonction des paramètres entrés dans theNum. Si les deux nombres peuvent être divisés, la valeur SUCCESS et le nombre s'affichent. Des erreurs spécifiques s'affichent si vous essayez de diviser par 0 ou si vous entrez un seul paramètre :

```
function divideNum(num1:Number, num2:Number):Number {
  if (isNaN(num1) || isNaN(num2)) {
  throw new Error("divideNum function requires two numeric parameters.");
  } else if (num2 == 0) {
  throw new Error("cannot divide by zero.");
```

```
  }
  return num1/num2;
}
try {
  var theNum:Number = divideNum(1, 0);
  trace("SUCCESS! "+theNum);
} catch (e_err:Error) {
  trace("ERROR! "+e_err.message);
  trace("\t"+e_err.name);
}
```

Si vous testez ce code ActionScript sans modifier les nombres que vous divisez, une erreur s'affiche dans le panneau de sortie car vous essayez de diviser par 0.

### Voir également

`Instruction throw`, `Instruction try..catch..finally`

## name (propriété Error.name)

```
public name : String
```

Contient le nom de l'objet Error. Par défaut, la valeur de cette propriété est "`Error`".

**Disponibilité :** ActionScript 1.0 ; Flash Player 7

### Exemple

Dans l'exemple suivant, une fonction renvoie une erreur spécifiée en fonction des deux nombres que vous essayez de diviser. Ajoutez le code ActionScript suivant à l'image 1 du scénario :

```
function divideNumber(numerator:Number, denominator:Number):Number {
  if (isNaN(numerator) || isNaN(denominator)) {
  throw new Error("divideNum function requires two numeric parameters.");
  } else if (denominator == 0) {
  throw new DivideByZeroError();
  }
  return numerator/denominator;
}
try {
  var theNum:Number = divideNumber(1, 0);
  trace("SUCCESS! "+theNum);
  // output: DivideByZeroError -> Unable to divide by zero.
} catch (e_err:DivideByZeroError) {
  // divide by zero error occurred
  trace(e_err.name+" -> "+e_err.toString());
} catch (e_err:Error) {
  // generic error occurred
  trace(e_err.name+" -> "+e_err.toString());
```

```
}
```

Pour ajouter une erreur personnalisée, insérez le code suivant dans un fichier .AS intitulé DivideByZeroError.as et enregistrez le fichier de classe dans le même répertoire que votre document FLA.

```
class DivideByZeroError extends Error {
  var name:String = "DivideByZeroError";
  var message:String = "Unable to divide by zero.";
}
```

### Voir également

`Instruction throw`, `Instruction try..catch..finally`

## toString (méthode Error.toString)

```
public toString() : String
```

Renvoie la chaîne "Error" par défaut ou la valeur contenue dans Error.message, s'il est défini.

**Disponibilité :** ActionScript 1.0 ; Flash Player 7

### Valeur renvoyée

`String` - Chaîne

### Exemple

Dans l'exemple suivant, une fonction renvoie une erreur (avec un message spécifié) si les deux chaînes qui lui sont transmises ne sont pas identiques :

```
function compareStrings(str1_str:String, str2_str:String):Void {
if (str1_str != str2_str) {
  throw new Error("Strings do not match.");
  }
}
try {
  compareStrings("Dog", "dog");
  // output: Strings do not match.
} catch (e_err:Error) {
  trace(e_err.toString());
}
```

### Voir également

`message (propriété Error.message)`, `Instruction throw`, `Instruction try..catch..finally`

# ExternalInterface (flash.external.ExternalInterface)

```
Object
  |
  +-flash.external.ExternalInterface
```

```
public class ExternalInterface
extends Object
```

La classe ExternalInterface est l'API External, une interface de programmation d'application qui autorise les communications simples entre ActionScript et le conteneur de Flash Player ; par exemple, une page HTML utilisant JavaScript ou une application de bureau intégrant Flash Player.

ExternalInterface a les mêmes fonctionnalités que les méthodes `fscommand()`, `CallFrame()` et `CallLabel()`, mais possède plus de flexibilité et peut s'appliquer plus généralement. L'utilisation de ExternalInterface est recommandée pour les communications entre JavaScript et ActionScript.

A partir d'ActionScript, vous pouvez appeler toutes les fonctions JavaScript sur une page HTML, passer tous les arguments de tout type de données et recevoir une valeur de retour pour un appel.

Depuis JavaScript sur la page HTML, vous pouvez appeler une fonction ActionScript dans Flash Player. La fonction ActionScript peut renvoyer une valeur que JavaScript reçoit immédiatement comme valeur de retour de l'appel.

ExternalInterface est pris en charge par les combinaisons suivantes de navigateurs et de systèmes d'exploitation :

| **Navigateur** | **Système d'exploitation** | |
|---|---|---|
| Internet Explorer 5.0 et versions ultérieures | Windows | |
| Netscape 8.0 et versions ultérieures | Windows | Macintosh |
| Mozilla 1.7.5 et versions ultérieures | Windows | Macintosh |
| Firefox 1.0 et versions ultérieures | Windows | Macintosh |
| Safari 1.3 et versions ultérieures | | Macintosh |

ExternalInterface requiert que le navigateur Web de l'utilisateur prenne en charge soit ActiveX soit l'API NPRuntime qui est proposée par certains navigateurs pour les scripts de plug-ins. Voir http://www.mozilla.org/projects/plugins/npruntime.html.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

## Résumé des propriétés

| Modificateurs | Propriété | Description |
|---|---|---|
| `static` | `available:Boolean` [lecture seule] | Indique si ce lecteur se trouve dans un conteneur doté d'une interface externe. |

*Propriétés héritées de la classe Object*

`constructeur (propriété Object.constructor), __proto__ (Object.__proto__ property), prototype (Object.prototype, propriété), __resolve (Object.__resolve, propriété)`

## Résumé de la méthode

| Modificateurs | Signature | Description |
|---|---|---|
| `static` | `addCallback(methodName:String, instance:Object, method:Function) : Boolean` | Enregistre une méthode ActionScript comme pouvant être appelée à partir du conteneur. |
| `static` | `call(methodName:String, [parameter1:Object]) : Object` | Appelle une fonction présentée par le conteneur Flash Player, en transmettant la valeur zéro ou d'autres arguments. |

*Méthodes héritées de la classe Object*

`addProperty (méthode Object.addProperty), hasOwnProperty (méthode Object.hasOwnProperty), isPropertyEnumerable (méthode Object.isPropertyEnumerable), isPrototypeOf (méthode Object.isPrototypeOf), registerClass (méthode Object.registerClass), toString (méthode Object.toString), unwatch (méthode Object.unwatch), valueOf (méthode Object.valueOf), watch (méthode Object.watch)`

# addCallback (méthode ExternalInterface.addCallback)

```
public static addCallback(methodName:String, instance:Object,
  method:Function) : Boolean
```

Enregistre une méthode ActionScript comme pouvant être appelée à partir du conteneur. Lorsque l'invocation de `addCallBack()` a réussi, la fonction enregistrée dans Flash Player peut être appelée par le code JavaScript ou ActiveX dans le conteneur.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

## Paramètres

*`methodName`*`:String` - Nom utilisé pour appeler la fonction ActionScript à partir de JavaScript. Il n'est pas nécessaire que ce nom corresponde au nom actuel de la méthode ActionScript.

*`instance`*`:Object` - Objet que `this` convertit dans la méthode. Ce n'est pas nécessairement l'objet sur lequel la méthode se trouve : vous pouvez spécifier n'importe quel objet (ou `null`).

*`method`*`:Function` - Méthode ActionScript à appeler à partir de JavaScript.

## Valeur renvoyée

`Boolean` - Renvoie `true` si l'appel a réussi. `false`est renvoyé s'il a échoué parce que l'occurrence n'était pas disponible, en raison d'une restriction de sécurité, parce qu'aucun objet fonction de ce type n'existait, en raison d'une erreur de récursivité, ou s'y apparentant.

Une valeur `false` renvoyée peut également signifier que l'environnement conteneur appartient à un Sandbox de sécurité auquel le code effectuant l'appel n'a pas accès. Vous pouvez contourner ce problème en définissant une valeur appropriée pour la balise `allowScriptAccess` `OBJECT` ou `EMBED` du HTML de l'environnement conteneur.

## Exemple

L'exemple suivant enregistre la fonction `goToMacromedia()` comme pouvant être appelée à partir du conteneur du nom de `goHome`.

```
import flash.external.*;

var methodName:String = "goHome";
var instance:Object = null;
var method:Function = goToMacromedia;
var wasSuccessful:Boolean = ExternalInterface.addCallback(methodName,
  instance, method);

var txtField:TextField = this.createTextField("txtField",
  this.getNextHighestDepth(), 0, 0, 200, 50);
```

```
txtField.border = true;
txtField.text = wasSuccessful.toString();

function goToMacromedia() {
  txtField.text = "http://www.macromedia.com";
  getURL("http://www.macromedia.com", "_self");
}
```

Pour que l'exemple ci-dessus fonctionne correctement, vous devez copier et coller le code suivant dans la page HTML conteneur. Ce code est basé sur le fait que l'attribut `id` de la balise `OBJECT` et l'attribut `name` de la balise `EMBED` doivent avoir la valeur `externalInterfaceExample`. La fonction `thisMovie` renvoie la syntaxe appropriée selon le navigateur, vu que Internet Explorer et Netscape font référence à l'objet movie différemment. A moins que la page HTML ne soit hébergée sur un serveur, votre navigateur peut vous alerter par un message de sécurité.

*Remarque* : Evitez d'utiliser d'autres méthodes d'accès à l'objet plug-in, telles que `document.getElementById("pluginName")` ou `document.all.pluginName`, parce que ces autres méthodes ne fonctionnent pas régulièrement sur tous les navigateurs.

```
<form>
  <input type="button" onclick="callExternalInterface()" value="Call
  ExternalInterface" />
</form>
<script>
function callExternalInterface() {
  thisMovie("externalInterfaceExample").goHome();
}

function thisMovie(movieName) {
  if (navigator.appName.indexOf("Microsoft") != -1) {
    return window[movieName]
  }
  else {
    return document[movieName]
  }
}
</script>
```

# available (propriété ExternalInterface.available)

```
public static available : Boolean [lecture seule]
```

Indique si ce lecteur se trouve dans un conteneur doté d'une interface externe. Si l'interface externe est disponible, cette propriété est `true` ; sinon, elle est `false`.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

### Exemple

L'exemple suivant utilise `ExternalInterface.available` pour déterminer si le lecteur est dans un conteneur doté d'une interface externe.

```
import flash.external.*;

var isAvailable:Boolean = ExternalInterface.available;
trace(isAvailable);
```

## call (méthode ExternalInterface.call)

```
public static call(methodName:String, [parameter1:Object]) : Object
```

Appelle une fonction présentée par le conteneur Flash Player, en transmettant la valeur zéro ou d'autres arguments. Si la fonction voulue n'est pas disponible, l'appel renvoie `null` ; sinon, elle renvoie la valeur fournie par la fonction. La récursivité n'est pas autorisée; un appel récursif entraîne une réponse `null`.

Si le conteneur correspond à une page HTML, cette méthode appelle une fonction JavaScript dans un élément `<script>`.

Si le conteneur est un autre conteneur de type ActiveX, cette méthode émet un événement ayant le nom spécifié ; le conteneur traite alors l'événement.

Si le conteneur renferme le plug-in Netscape, vous pouvez soit écrire la prise en charge personnalisé de la nouvelle interface NPRuntime, soit intégrer un contrôle HTML et intégrer Flash Player dans le contrôle HTML. Si vous intégrez un contrôle HTML, vous pouvez communiquer avec Flash Player via une interface JavaScript qui dialogue avec l'application conteneur native.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

### Paramètres

*`methodName`*`:String` - Nom de la fonction à appeler dans le conteneur. Si la fonction accepte les paramètres, ils doivent apparaître après le paramètre `methodName`.

*`parameter1`*`:Object` [facultatif] - Tout paramètre à passer à la fonction. Vous pouvez ne spécifier aucun paramètre ou en spécifier plusieurs en les séparant par des virgules. Ces paramètres peuvent être de tout type de donnée ActionScript. Si vous faites appel à une fonction JavaScript, les types ActionScript sont automatiquement rentrés en ordre dans les types JavaScript. Si vous faites appel à un autre conteneur ActiveX, les paramètres sont encodés dans le message de requête.

## Valeur renvoyée

`Object` - Réponse émanant du conteneur. Si l'appel a échoué parce que cette fonction ne se trouvait pas dans le conteneur, l'interface n'était pas disponible, une erreur de récursivité s'est produite ou en raison d'un problème de sécurité, la valeur `null` est renvoyée.

## Exemple

L'exemple suivant appelle la fonction JavaScript `sayHello()` dans la page HTML contenant le fichier SWF. L'appel est effectué en utilisant la méthode `ExternalInterface.call()`.

```
import flash.external.*;

var greeting:String;
var btn:MovieClip = createButton(100, 30, 0xCCCCCC);
btn.onPress = function() {
  greeting = String(ExternalInterface.call("sayHello", "browser"));
  this.mcTxt.text = greeting; // >> Hi Flash.
}

function createButton(width:Number, height:Number, color:Number):MovieClip
  {
  var depth:Number = this.getNextHighestDepth();
  var mc:MovieClip = this.createEmptyMovieClip("mc_" + depth, depth);
  var mcFmt:TextFormat;

  mc.beginFill(color);
  mc.lineTo(0, height);
  mc.lineTo(width, height);
  mc.lineTo(width, 0);
  mc.lineTo(0, 0);

  mcFmt = new TextFormat();
  mcFmt.align = "center";
  mcFmt.bold = true;

  mc.createTextField("mcTxt", depth, 0, 0, width, height);
  mc.mcTxt.text = "Call JS Function";
  mc.mcTxt.setTextFormat(mcFmt);

  return mc;
}
```

Pour que l'exemple ci-dessus fonctionne correctement, vous devez copier et coller le code suivant dans la page HTML conteneur. A moins que la page HTML ne soit hébergée sur un serveur, votre navigateur peut vous alerter par un message de sécurité.

```
<script>
  function sayHello(name) {
    alert(">> Hello " + name + ".");
    return ">> Hi Flash.";
```

```
  }
</script>
```

# FileReference (flash.net.FileReference)

```
Object
  |
  +-flash.net.FileReference
```

`public class` ***FileReference***
`extends Object`

La classe FileReference permet de charger et télécharger des fichiers entre l'ordinateur d'un utilisateur et le serveur. Une boîte de dialogue d'un système d'exploitation invite l'utilisateur à sélectionner un fichier pour le charger ou à choisir un emplacement pour le télécharger. Chaque objet FileReference fait référence à un fichier unique sur le disque dur de l'utilisateur et inclut des propriétés contenant des informations sur la taille, le type, le nom, la date de création, la date de modification et le type de créateur du fichier (Macintosh uniquement).

Les occurrences FileReference sont créées de deux façons :

- En utilisant l'opérateur `new` avec le constructeur FileReference : `var myFileReference = new FileReference();`
- En appelant `FileReferenceList.browse()`, ce qui crée un tableau d'objets FileReference

Pendant une opération de chargement, toutes les propriétés d'un objet FileReference sont complétées d'appels à `FileReference.browse()` ou `FileReferenceList.browse()`. Pendant une opération de téléchargement, la propriété `name` est complétée quand `onSelect` a été invoqué ; toutes les autres propriétés sont complétées quand `onComplete` a été invoqué.

La méthode `browse()` ouvre une boîte de dialogue d'un système d'exploitation qui invite l'utilisateur à sélectionner un fichier local pour le télécharger. La méthode `FileReference.browse()` permet à l'utilisateur de choisir un seul fichier ; la méthode `FileReferenceList.browse()` lui permet d'en sélectionner plusieurs. Lorsque l'appel de la méthode `browse()` a réussi, appelez la méthode `FileReference.upload()` pour charger un fichier à la fois. La méthode `FileReference.download()` invite l'utilisateur à sélectionner un emplacement pour enregistrer le fichier et initie le téléchargement à partir d'une URL distante.

les classes FileReference et FileReferenceList ne vous permettent pas de définir un emplacement de fichier par défaut pour la boîte de dialogue générée par les appels`browse()` et `download()`. L'emplacement par défaut sélectionné dans les boîtes de dialogue est le dernier dossier parcouru, dans la mesure où il est possible de déterminer cet emplacement, ou le bureau. Les classes ne vous permettent la lecture ou l'écriture sur les fichiers transférés. Elles ne permettent pas au fichier SWF qui a initié le téléchargement d'accéder au fichier téléchargé ou à l'emplacement du fichier sur le disque de l'utilisateur.

Les classes FileReference et FileReferenceList ne fournissent pas de méthode d'authentification. Pour les serveurs ayant besoin d'une authentification, vous pouvez télécharger des fichiers avec le plug-in navigateur Flash Player , mais le chargement (sur tous les lecteurs) et le téléchargement (sur les lecteurs autonomes ou externes) échouent. Utilisez les écouteurs d'événements FileReference afin de déterminer si les opérations ont réussi, ou non, et pour traiter les erreurs.

Pour les opérations de téléchargement (montantes ou descendantes), un fichier SWF peut accéder aux fichiers uniquement à l'intérieur de son propre domaine, ce qui comprend tous les domaines spécifiés par un fichier de régulation inter-domaines. Si le SWF qui initialise le téléchargement n'appartient pas au même domaine que le serveur de fichiers, vous devez placer un fichier de régulation sur le serveur de fichiers.

Pendant que les appels aux méthodes `FileReference.browse()`, `FileReferenceList.browse()`, ou `FileReference.download()` s'exécutent, la lecture du fichier SWF s'arrête sur les plate-formes suivantes : le plug-in Flash Player pour Mac OS X, le Flash Player externe pour Macintosh et le lecteur autonome pour Mac OS X 10.1 et versions ultérieures. Le fichier SWF continue à s'exécuter sur tous les lecteurs pour Windows et sur le lecteur autonome pour Macintosh sur Mac OS X 10.2 et versions ultérieures

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

## Exemple

L'exemple suivant crée un objet FileReference qui invite l'utilisateur à sélectionner une image ou un fichier texte à télécharger. Il écoute également les événements possibles.

```
import flash.net.FileReference;

var allTypes:Array = new Array();
var imageTypes:Object = new Object();
imageTypes.description = "Images (*.jpg, *.jpeg, *.gif, *.png)";
imageTypes.extension = "*.jpg; *.jpeg; *.gif; *.png";
allTypes.push(imageTypes);

var textTypes:Object = new Object();
textTypes.description = "Text Files (*.txt, *.rtf)";
textTypes.extension = "*.txt;*.rtf";
```

```
allTypes.push(textTypes);

var listener:Object = new Object();

listener.onSelect = function(file:FileReference):Void {
  trace("onSelect: " + file.name);
  if(!file.upload("http://www.yourdomain.com/
  yourUploadHandlerScript.cfm")) {
    trace("Upload dialog failed to open.");
  }
}

listener.onCancel = function(file:FileReference):Void {
  trace("onCancel");
}

listener.onOpen = function(file:FileReference):Void {
  trace("onOpen: " + file.name);
}

listener.onProgress = function(file:FileReference, bytesLoaded:Number,
  bytesTotal:Number):Void {
  trace("onProgress with bytesLoaded: " + bytesLoaded + " bytesTotal: " +
  bytesTotal);
}

listener.onComplete = function(file:FileReference):Void {
  trace("onComplete: " + file.name);
}

listener.onHTTPError = function(file:FileReference):Void {
  trace("onHTTPError: " + file.name);
}

listener.onIOError = function(file:FileReference):Void {
  trace("onIOError: " + file.name);
}

listener.onSecurityError = function(file:FileReference,
  errorString:String):Void {
  trace("onSecurityError: " + file.name + " errorString: " + errorString);
}

var fileRef:FileReference = new FileReference();
fileRef.addListener(listener);
fileRef.browse(allTypes);
```

## Voir également

FileReferenceList (flash.net.FileReferenceList)

## Résumé des propriétés

| Modificateurs | Propriété | Description |
|---|---|---|
| | `creationDate:Date` [lecture seule] | Date de création du fichier sur le disque local. |
| | `creator:String` [lecture seule] | Type de créateur Macintosh du fichier. |
| | `modificationDate:Date` [lecture seule] | Date de la dernière modification du fichier sur le disque local. |
| | `name:String` [lecture seule] | Nom du fichier sur le disque local. |
| | `size:Number` [lecture seule] | Taille du fichier sur le disque local en octets. |
| | `type:String` [lecture seule] | Type de fichier. |

*Propriétés héritées de la classe Object*

`constructeur (propriété Object.constructor), __proto__ (Object.__proto__ property), prototype (Object.prototype, propriété), __resolve (Object.__resolve, propriété)`

## Résumé des événements

| Evénement | Description |
|---|---|
| `onCancel = function(fileRef:FileReference) {}` | Appelé lorsque l'utilisateur ferme la boîte de dialogue de recherche de fichier. |
| `onComplete = function(fileRef:FileReference) {}` | Appelé en cas de réussite de l'opération d'envoi ou de téléchargement. |
| `onHTTPError = function(fileRef:FileReference, httpError:Number) {}` | Appelé lorsqu'un envoi échoue en raison d'une erreur HTTP. |

| Evénement | Description |
|---|---|
| `onIOError = function(fileRef:FileReference) {}` | Appelé lorsqu'une erreur d'entrée/sortie se produit. |
| `onOpen = function(fileRef:FileReference) {}` | Appelé au début d'une opération d'envoi ou de téléchargement. |
| `onProgress = function(fileRef:FileReference, bytesLoaded:Number, bytesTotal:Number) {}` | Appelé régulièrement pendant l'opération d'envoi ou de téléchargement. |
| `onSecurityError = function(fileRef:FileReference, errorString:String) {}` | Appelé lorsqu'un envoi ou un téléchargement échoue en raison d'une erreur de sécurité. |
| `onSelect = function(fileRef:FileReference) {}` | Appelé lorsque l'utilisateur sélectionne un fichier à envoyer ou télécharger dans la boîte de dialogue de recherche de fichiers. |

## Récapitulatif des constructeurs

| Signature | Description |
|---|---|
| `FileReference()` | Crée un nouvel objet FileReference. |

## Résumé de la méthode

| Modificateurs | Signature | Description |
|---|---|---|
| | `addListener(listener:Object) : Void` | Enregistre un objet pour recevoir une notification lorsqu'un écouteur d'événements FileReference est invoqué. |
| | `browse([typelist:Array]) : Boolean` | Affiche une boîte de dialogue de recherche de fichier dans laquelle l'utilisateur peut sélectionner un fichier local à envoyer. |
| | `cancel() : Void` | Annule une opération de chargement ou de téléchargement en cours sur cet objet FileReference. |
| | `download(url:String, [defaultFileName:String]) : Boolean` | Affiche une boîte de dialogue permettant à l'utilisateur de télécharger un fichier à partir d'un serveur distant. |
| | `removeListener(listener:Object) : Boolean` | Supprime un objet de la liste d'objets recevant des messages de notification d'événement. |
| | `upload(url:String) : Boolean` | Procède au chargement d'un fichier sélectionné par un utilisateur sur un serveur distant. |

*Méthodes héritées de la classe Object*

`addProperty (méthode Object.addProperty)`, `hasOwnProperty (méthode Object.hasOwnProperty)`, `isPropertyEnumerable (méthode Object.isPropertyEnumerable)`, `isPrototypeOf (méthode Object.isPrototypeOf)`, `registerClass (méthode Object.registerClass)`, `toString (méthode Object.toString)`, `unwatch (méthode Object.unwatch)`, `valueOf (méthode Object.valueOf)`, `watch (méthode Object.watch)`

# addListener (méthode FileReference.addListener)

```
public addListener(listener:Object) : Void
```

Enregistre un objet pour recevoir une notification lorsqu'un écouteur d'événements FileReference est invoqué.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

## Paramètres

*`listener`*`:Object` - Objet qui écoute une notification de rappel venant des écouteurs d'événements FileReference.

### Exemple

L'exemple suivant ajoute un écouteur à une occurrence de FileReference.

```
import flash.net.FileReference;

var listener:Object = new Object();

listener.onProgress = function(file:FileReference, bytesLoaded:Number,
  bytesTotal:Number):Void {
  trace("onProgress with bytesLoaded: " + bytesLoaded + " bytesTotal: " +
  bytesTotal);
}

listener.onComplete = function(file:FileReference):Void {
  trace("onComplete: " + file.name);
}

var fileRef:FileReference = new FileReference();
fileRef.addListener(listener);
var url:String = "http://www.macromedia.com/platform/whitepapers/
  platform_overview.pdf";
fileRef.download(url, "FlashPlatform.pdf");
```

## browse (méthode FileReference.browse)

```
public browse([typelist:Array]) : Boolean
```

Affiche une boîte de dialogue de recherche de fichier dans laquelle l'utilisateur peut sélectionner un fichier local à envoyer. La boîte de dialogue est spécifique au système d'exploitation de l'utilisateur. Lorsque vous appelez cette méthode et que l'utilisateur réussit à sélectionner un fichier, les propriétés de cet objet FileReference sont renseignées par les propriétés de ce fichier. Toutes les fois suivantes où `FileReference.browse()` est appelé, les propriétés de l'objet FileReference sont restaurées pour le fichier choisi par l'utilisateur dans la boîte de dialogue.

Une seule session `browse()` ou `download()` peut être effectuée à la fois (car une seule boîte de dialogue peut être appelée à la fois).

Vous pouvez remplir un tableau des types de fichiers pour déterminer lesquels seront affichés par la boîte de dialogue.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

## Paramètres

*typelist*:Array [facultatif] - Tableau de types de fichiers utilisés pour filtrer les fichiers qui s'affichent dans la boîte de dialogue. Si vous omettez ce paramètre, tous les fichiers s'affichent. Si vous incluez ce paramètre, le tableau doit contenir un ou plusieurs éléments placés entre accolades { }. Vous pouvez appliquer l'un des deux formats au tableau :

- Une liste de types de fichiers suivis de leur extension Windows uniquement. Chaque élément du tableau doit contenir une chaîne décrivant le type de fichier et une liste, séparée par des points-virgules, des extensions de fichiers Windows, chaque extension étant précédée d'un caractère joker (*). La syntaxe de chaque élément se présente comme suit : `[{description: "string describing the first set of file types", extension: "semicolon-delimited list of file extensions"}]`
  Exemple :
  ```
  [{description: "Images", extension: "*.jpg;*.gif;*.png"}, {description:
  "Flash Movies", extension: "*.swf"}, {description: "Documents",
  extension: "*.doc;*.pdf"}]
  ```
- Une liste de types de fichiers suivis de leur extension Windows et de leur type de fichier Macintosh. Chaque élément du tableau doit contenir une chaîne décrivant le type de fichier ; une liste, séparée par des points-virgules, des extensions de fichiers Windows, chaque extension étant précédée d'un caractère joker (*) et une liste, séparée par des points-virgules, des types de fichiers Macintosh, chaque type étant précédé d'un caractère joker (*). La syntaxe de chaque élément se présente comme suit :
  ```
  [{description: "string describing the first set of file types",
  extension: "semicolon-delimited list of Windows file extensions",
  macType: "semicolon-delimited list of Macintosh file types"}]
  ```
  Exemple :
  ```
  [{description: "Image files", extension: "*.jpg;*.gif;*.png", macType:
  "JPEG;jp2_;GIFF"}, {description: "Flash Movies", extension: "*.swf",
  macType: "SWFL"}]
  ```

Les deux formats ne sont pas interchangeables dans un appel `browse()` unique. Vous devez utiliser l'un ou l'autre.

La liste des extensions est utilisée pour filtrer les fichiers sous Windows, selon le type de fichier sélectionné. Elle n'est pas réellement affichée dans la boîte de dialogue. Pour afficher les types de fichiers aux utilisateurs, vous devez lister les types de fichiers dans la chaîne descriptive ainsi que dans la liste d'extensions. Sous Windows, la chaîne descriptive est affichée dans la boîte de dialogue. (Elle n'est pas utilisée sous Macintosh.) Sous Macintosh, une liste de types de fichier Macintosh est utilisée pour filtrer les fichiers, si elle est fournie. Si vous ne fournissez pas de liste de types de fichiers Macintosh, c'est la liste d'extensions Windows qui est utilisée.

## Valeur renvoyée

`Boolean` - Renvoie `true` si les paramètres sont valides et que la boîte de dialogue est affichée. Renvoie `false` si la boîte de dialogue ne s'affiche pas, si une autre session browse est déjà en cours ou si vous utilisez le paramètre `typelist` sans fournir une description ou une chaîne d'extension concernant un élément du tableau.

## Evénements

onCancel

Invoqué quand l'utilisateur ferme la boîte de dialogue en cliquant sur Annuler ou en la fermant.

onSelect

Invoqué quand l'utilisateur réussit à choisir un élément à télécharger dans la boîte de dialogue.

## Exemple

L'exemple suivant affiche une boîte de dialogue dans laquelle l'utilisateur peut choisir un fichier image à télécharger.

```
import flash.net.FileReference;

var listener:Object = new Object();
listener.onSelect = function(file:FileReference):Void {
  trace("Opened " + file.name);
}

listener.onCancel = function(file:FileReference):Void {
  trace("User cancelled");
}

var fileRef:FileReference = new FileReference();
fileRef.addListener(listener);
fileRef.browse();
```

## Voir également

onSelect (écouteur d'événement FileReferenceList.onSelect), onCancel (écouteur d'événement FileReference.onCancel), download (méthode FileReference.download), browse (méthode FileReferenceList.browse)

## cancel (méthode FileReference.cancel)

```
public cancel() : Void
```

Annule une opération de chargement ou de téléchargement en cours sur cet objet FileReference.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

### Exemple

L'exemple suivant télécharge environ la moitié du fichier demandé et annule ensuite le téléchargement. Ce n'est évidemment pas une utilisation classique. Vous pourriez plus souvent utiliser cette méthode pour permettre aux utilisateurs de cliquer sur Annuler dans une boîte de dialogue de statut de téléchargement.

```
import flash.net.FileReference;

var listener:Object = new Object();

listener.onProgress = function(file:FileReference, bytesLoaded:Number,
  bytesTotal:Number):Void {
  trace("onProgress with bytesLoaded: " + bytesLoaded + " bytesTotal: " +
  bytesTotal);
  if(bytesLoaded >= (bytesTotal / 2)) {
    file.cancel();
  }
}

var fileRef:FileReference = new FileReference();
fileRef.addListener(listener);
var url:String = "http://www.macromedia.com/platform/whitepapers/
  platform_overview.pdf";
fileRef.download(url, "FlashPlatform.pdf");
```

## creationDate (propriété FileReference.creationDate)

```
public creationDate : Date [lecture seule]
```

Date de création du fichier sur le disque local. Si l'objet FileReference n'a pas été renseigné, l'appel effectué pour obtenir la valeur de cette propriété renvoie `null`.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

### Exemple

L'exemple suivant récupère la date de création d'un fichier choisi par l'utilisateur.

```
import flash.net.FileReference;

var listener:Object = new Object();
```

```
listener.onSelect = function(file:FileReference):Void {
  trace("creationDate: " + file.creationDate);
}

var fileRef:FileReference = new FileReference();
fileRef.addListener(listener);
fileRef.browse();
```

### Voir également

browse (méthode FileReference.browse)

## creator (propriété FileReference.creator)

```
public creator : String [lecture seule]
```

Type de créateur Macintosh du fichier. Sous Windows, cette propriété est `null`. Si l'objet FileReference n'a pas été renseigné, l'appel effectué pour obtenir la valeur de cette propriété renvoie `null`.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

### Exemple

L'exemple suivant récupère le type de créateur Macintosh d'un fichier choisi par l'utilisateur.

```
import flash.net.FileReference;

var listener:Object = new Object();
listener.onSelect = function(file:FileReference):Void {
  trace("creator: " + file.creator);
}

var fileRef:FileReference = new FileReference();
fileRef.addListener(listener);
fileRef.browse();
```

### Voir également

browse (méthode FileReference.browse)

## download (méthode FileReference.download)

```
public download(url:String, [defaultFileName:String]) : Boolean
```

Affiche une boîte de dialogue permettant à l'utilisateur de télécharger un fichier à partir d'un serveur distant. Flash Player peut télécharger des fichiers jusqu'à 100 Mo.

Cette méthode ouvre d'abord une boîte de dialogue d'un système d'exploitation demandant à l'utilisateur d'entrer un nom de fichier et de sélectionner un emplacement sur l'ordinateur local pour enregistrer le fichier. Quand l'utilisateur choisit un emplacement et confirme le téléchargement (par ex. en cliquant sur Enregistrer), celui-ci commence sur le serveur distant. Les écouteurs reçoivent des événements permettant d'indiquer la progression du téléchargement, s'il a réussi ou échoué. Pour déterminer le statut de la boîte de dialogue et l'opération de téléchargement après avoir appelé la méthode `download()`, votre code ActionScript doit écouter les événements tels que `onCancel`, `onOpen`, `onProgress`, et `onComplete`.

Une fois le fichier téléchargé, les propriétés de l'objet FileReference sont renseignées par les propriétés du fichier local et l'écouteur `onComplete` est invoqué.

Une seule session `browse()` ou `download()` peut être effectuée à la fois (car une seule boîte de dialogue peut être appelée à la fois).

Cette méthode prend en charge le téléchargement de tout type de fichier, via HTTP ou HTTPS. Vous pouvez également envoyer des données au serveur avec l'appel de `download()` en ajoutant des paramètres à l'URL pour que le script serveur les analyse.

*Remarque* : Si votre serveur requiert une authentification d'utilisateur, seuls les fichiers SWF s'exécutant dans un navigateur, (c'est-à-dire utilisant le plug-in du navigateur ou le contrôle ActiveX), peuvent fournir une boîte de dialogue pour demander à l'utilisateur un nom et un mot de passe d'authentification, ceci uniquement pour les téléchargements. Concernant les chargements effectués via le plug-in ou le contrôle ActiveX et le chargement/téléchargement via les lecteurs autonomes ou externes, le transfert de fichiers échoue.

Pour utiliser cette méthode, tenez compte du modèle de sécurité de Flash Player :

- Interdit si le fichier SWF appelant est dans une Sandbox locale non sûre.
- Par défaut, l'accès est refusé entre Sandboxes. Un site web peut autoriser l'accès à une ressource en ajoutant un fichier de régulation inter-domaines.

Pour plus d'informations, consultez les sections suivantes :

- Chapitre 17, « Compréhension de la sécurité » dans *Apprentissage d' ActionScript 2.0 dans Flash*
- Le livre blanc sur la sécurité dans Flash Player 8
- Le livre blanc sur la sécurité dans Flash Player 8 en liaison avec les API

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

## Paramètres

*url*:String - URL du fichier à télécharger sur l'ordinateur local. Vous pouvez également envoyer des données au serveur avec l'appel de download() en ajoutant des paramètres à l'URL pour que le script serveur les analyse. Voici un exemple d'utilisation : http://www.myserver.com/picture.jpg?userID=jdoe

Sur certains navigateurs, les chaînes d'URL ont une longueur limitée. Une longueur supérieure à 256 caractères peut échouer sur certains navigateurs ou serveurs.

*defaultFileName*:String [facultatif] - Nom de fichier par défaut affiché dans la boîte de dialogue du fichier à télécharger. Les caractères suivants sont interdits dans cette chaîne : / \ : * ? " < > | %

Si vous omettez ce paramètre, le nom de fichier de l'URL distante est analysé et utilisé par défaut.

## Valeur renvoyée

Boolean - Valeur true si la boîte de dialogue permettant à l'utilisateur de choisir un fichier est affichée. Si la boîte de dialogue ne s'affiche pas, la méthode renvoie false. La boîte de dialogue peut ne pas s'afficher pour l'une des raisons suivantes :

- Vous n'avez pas passé de valeur pour le paramètre url.
- Le type ou le format des paramètres transmis n'est pas correct.
- Le paramètre url a une longueur zéro.
- Une violation de sécurité s'est produite; c'est-à-dire que votre fichier SWF a essayé d'accéder à un fichier sur un serveur hors de la Sandbox de sécurité de votre fichier SWF.
- Une autre session browse est déjà en cours. Une session browse peut commencer par FileReference.browse(), FileReferenceList.browse(), ou FileReference.download().
- Le protocole utilisé n'est pas le protocole HTTP ou HTTPS.

## Evénements

onCancel

Invoqué lorsque l'utilisateur ferme la boîte de dialogue.

onComplete

Appelé en cas de réussite de l'opération de téléchargement.

onIOError

Invoqué pour l'une des raisons suivantes :

- Une erreur d'entrée/sortie se produit lors de la lecture ou de la transmission du fichier.
- Le fichier SWF tente de télécharger un fichier à partir d'un serveur nécessitant une authentification, dans le lecteur autonome ou externe. Au cours du téléchargement, les lecteurs autonomes et externes ne permettent pas aux utilisateurs d'entrer des mots de passe. Si un fichier SWF présent dans ces lecteurs tente de télécharger un fichier à partir d'un serveur nécessitant une authentification, le téléchargement échoue. Le téléchargement de fichiers peut réussir uniquement via les lecteurs avec contrôle ActiveX et plug-in navigateur.

onOpen

Invoqué lors du démarrage d'un téléchargement.

onProgress

Appelé régulièrement pendant le téléchargement du fichier.

onSecurityError

Appelé lorsqu'un téléchargement échoue en raison d'une erreur de sécurité.

onSelect

Invoqué quand l'utilisateur choisit un fichier à partir de la boîte de dialogue de téléchargement.

## Exemple

L'exemple suivant tente de télécharger un fichier par la méthode `download`. Veuillez noter qu'il y a des écouteurs pour tous les événements.

```
import flash.net.FileReference;

var listener:Object = new Object();

listener.onSelect = function(file:FileReference):Void {
  trace("onSelect: " + file.name);
}

listener.onCancel = function(file:FileReference):Void {
  trace("onCancel");
}

listener.onOpen = function(file:FileReference):Void {
```

```
  trace("onOpen: " + file.name);
}

listener.onProgress = function(file:FileReference, bytesLoaded:Number,
  bytesTotal:Number):Void {
  trace("onProgress with bytesLoaded: " + bytesLoaded + " bytesTotal: " +
  bytesTotal);
}

listener.onComplete = function(file:FileReference):Void {
  trace("onComplete: " + file.name);
}

listener.onIOError = function(file:FileReference):Void {
  trace("onIOError: " + file.name);
}

var fileRef:FileReference = new FileReference();
fileRef.addListener(listener);
var url:String = "http://www.macromedia.com/platform/whitepapers/
  platform_overview.pdf";
if(!fileRef.download(url, "FlashPlatform.pdf")) {
  trace("dialog box failed to open.");
}
```

### Voir également

browse (méthode FileReference.browse), browse (méthode FileReferenceList.browse), upload (méthode FileReference.upload)

## constructeur FileReference

```
public FileReference()
```

Crée un nouvel objet FileReference. Quand il est complété, un objet FileReference représente un fichier sur le disque local de l'utilisateur.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

### Exemple

L'exemple suivant crée un nouvel objet FileReference et initie le téléchargement d'un fichier PDF.

```
import flash.net.FileReference;

var listener:Object = new Object();
listener.onComplete = function(file:FileReference) {
  trace("onComplete : " + file.name);
```

```
}

var url:String = "http://www.macromedia.com/platform/whitepapers/
  platform_overview.pdf";
var fileRef:FileReference = new FileReference();
fileRef.addListener(listener);
fileRef.download(url, "FlashPlatform.pdf");
```

### Voir également

browse (méthode FileReference.browse)

## modificationDate (propriété FileReference.modificationDate)

```
public modificationDate : Date [lecture seule]
```

Date de la dernière modification du fichier sur le disque local. Si l'objet FileReference n'a pas été renseigné, l'appel effectué pour obtenir la valeur de cette propriété renvoie `null`.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

### Exemple

L'exemple suivant récupère la propriété `modificationDate` d'un fichier choisi par l'utilisateur.

```
import flash.net.FileReference;

var listener:Object = new Object();
listener.onSelect = function(file:FileReference):Void {
  trace("modificationDate: " + file.modificationDate);
}

var fileRef:FileReference = new FileReference();
fileRef.addListener(listener);
fileRef.browse();
```

### Voir également

browse (méthode FileReference.browse)

## name (propriété FileReference.name)

```
public name : String [lecture seule]
```

Nom du fichier sur le disque local. Si l'objet FileReference n'a pas été renseigné, l'appel effectué pour obtenir la valeur de cette propriété renvoie `null`.

Toutes les propriétés d'un objet FileReference sont complétées en appelant `browse()`. A la différence des autres propriétés FileReference, si vous appelez `download()`, la propriété `name` est renseignée quand `onSelect` est invoqué.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

### Exemple

L'exemple suivant récupère le nom d'un fichier choisi par l'utilisateur.

```
import flash.net.FileReference;

var listener:Object = new Object();
listener.onSelect = function(file:FileReference):Void {
  trace("name: " + file.name);
}

var fileRef:FileReference = new FileReference();
fileRef.addListener(listener);
fileRef.browse();
```

### Voir également

`browse (méthode FileReference.browse)`

## onCancel (écouteur d'événement FileReference.onCancel)

onCancel = function(fileRef:FileReference) {}

Appelé lorsque l'utilisateur ferme la boîte de dialogue de recherche de fichier. Cette boîte de dialogue est affichée quand vous appelez `FileReference.browse()`, `FileReferenceList.browse()`, ou `FileReference.download()`.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

### Paramètres

*`fileRef`*`:flash.net.FileReference` - Objet FileReference qui a initié l'opération.

### Exemple

L'exemple suivant trace un message si l'utilisateur ferme la boîte de dialogue de recherche de fichier. Cette méthode est déclenchée uniquement si l'utilisateur clique sur Annuler ou appuie sur la touche Echap après l'affichage de la boîte de dialogue.

```
import flash.net.FileReference;

var listener:Object = new Object();
```

```
listener.onCancel = function(file:FileReference):Void {
  trace("onCancel");
}

var fileRef:FileReference = new FileReference();
fileRef.addListener(listener);
var url:String = "http://www.macromedia.com/platform/whitepapers/
  platform_overview.pdf";
if(!fileRef.download(url, "FlashPlatform.pdf")) {
  trace("dialog box failed to open.");
}
```

# onCancel (écouteur d'événement FileReference.onComplete)

onCancel = function(fileRef:FileReference) {}

Appelé en cas de réussite de l'opération d'envoi ou de téléchargement. L'exécution réussie signifie que la totalité du fichier a été téléchargée (reçue ou envoyée).

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

## Paramètres

*fileRef*:flash.net.FileReference - Objet FileReference qui a initié l'opération.

## Exemple

L'exemple suivant trace un message quand l'événement `onComplete` est déclenché.

```
import flash.net.FileReference;

var listener:Object = new Object();

listener.onComplete = function(file:FileReference):Void {
  trace("onComplete: " + file.name);
}

var fileRef:FileReference = new FileReference();
fileRef.addListener(listener);
var url:String = "http://www.macromedia.com/platform/whitepapers/
  platform_overview.pdf";
fileRef.download(url, "FlashPlatform.pdf");
```

# onHTTPError (écouteur d'événement FileReference.onHTTPError)

onHTTPError = function(fileRef:FileReference, httpError:Number) {}

Appelé lorsqu'un envoi échoue en raison d'une erreur HTTP.

Compte tenu de la façon dont Flash Player s'appuie sur l'ordre d'empilement du navigateur au cours du téléchargement de fichiers, cette erreur n'est pas applicable aux échecs de téléchargement. Si un téléchargement échoue en raison d'une erreur HTTP, l'erreur est signalée en tant qu'erreur d'E/S.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

## Paramètres

*fileRef*:flash.net.FileReference - Objet FileReference qui a initié l'opération.

*httpError*:Number - Erreur HTTP à l'origine de l'échec de ce chargement. Par exemple, une erreur httpError 404 indique qu'une page n'a pas été trouvée. Les valeurs d'erreur HTTP sont répertoriées dans les sections 10.4 et 10.5 de la spécification HTTP à l'adresse ftp://ftp.isi.edu/in-notes/rfc2616.txt.

## Exemple

L'exemple suivant crée un objet FileReference comprenant un écouteur pour chaque événement possible, y compris onHttpError. Cet écouteur est déclenché uniquement en cas d'échec du chargement dû à une erreur HTTP.

```
import flash.net.FileReference;

var listener:Object = new Object();

listener.onSelect = function(file:FileReference):Void {
  trace("onSelect: " + file.name);
  if(!file.upload("http://www.yourdomain.com/
  yourUploadHandlerScript.cfm")) {
    trace("Upload dialog failed to open.");
  }
}

listener.onCancel = function(file:FileReference):Void {
  trace("onCancel");
}

listener.onOpen = function(file:FileReference):Void {
  trace("onOpen: " + file.name);
}
```

```
listener.onProgress = function(file:FileReference, bytesLoaded:Number,
  bytesTotal:Number):Void {
  trace("onProgress with bytesLoaded: " + bytesLoaded + " bytesTotal: " +
  bytesTotal);
}

listener.onComplete = function(file:FileReference):Void {
  trace("onComplete: " + file.name);
}

listener.onHTTPError = function(file:FileReference):Void {
  trace("onHTTPError: " + file.name);
}

listener.onIOError = function(file:FileReference):Void {
  trace("onIOError: " + file.name);
}

listener.onSecurityError = function(file:FileReference,
  errorString:String):Void {
  trace("onSecurityError: " + file.name + " errorString: " + errorString);
}

var fileRef:FileReference = new FileReference();
fileRef.addListener(listener);
fileRef.browse();
```

# onIOError (écouteur d'événement FileReference.onIOError)

## onCancel = function(fileRef:FileReference) {}

Appelé lorsqu'une erreur d'entrée/sortie se produit.

Cet écouteur est invoqué lorsque le chargement ou téléchargement échoue pour l'une des raisons suivantes :

- Une erreur d'entrée/sortie se produit lors de la lecture, l'écriture ou la transmission du fichier.
- Le fichier SWF tente de charger un fichier sur un serveur nécessitant une authentification (comme un nom d'utilisateur et un mot de passe). Au cours du chargement, Flash Player ne permet pas aux utilisateurs d'entrer des mots de passe. Si un fichier SWF tente de charger un fichier sur un serveur nécessitant une authentification, le chargement échoue.

- Le fichier SWF tente de télécharger un fichier à partir d'un serveur nécessitant une authentification, dans le lecteur autonome ou externe. Au cours du téléchargement, les lecteurs autonomes et externes ne permettent pas aux utilisateurs d'entrer des mots de passe. Si un fichier SWF présent dans ces lecteurs tente de télécharger un fichier à partir d'un serveur nécessitant une authentification, le téléchargement échoue. Le téléchargement de fichiers peut réussir uniquement via les lecteurs avec contrôle ActiveX et plug-in navigateur.
- La valeur passée au paramètre `url` dans `upload()` contient un protocole invalide. Les protocoles valides sont HTTP et HTTPS.

Important : Seules les applications Flash s'exécutant dans un navigateur, via le plug-in du navigateur ou le contrôle ActiveX, peuvent fournir une boîte de dialogue pour inviter l'utilisateur à entrer un nom d'utilisateur et un mot de passe en vue de l'authentification, et ceci uniquement pour les téléchargements. Concernant les chargements effectués via le plug-in ou le contrôle ActiveX, ou le chargement/téléchargement via les lecteurs autonomes ou externes, le transfert de fichiers échoue.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

## Paramètres

*`fileRef`*`:flash.net.FileReference` - Objet FileReference qui a initié l'opération.

## Exemple

L'exemple suivant trace un message quand l'événement `onIOError` est déclenché. Pour simplifier, aucun des autres écouteurs d'événements n'est inclus dans cet exemple.

```
import flash.net.FileReference;

var listener:Object = new Object();

listener.onIOError = function(file:FileReference):Void {
  trace("onIOError");
}

var fileRef:FileReference = new FileReference();
fileRef.addListener(listener);
fileRef.download("http://www.macromedia.com/NonExistentFile.pdf",
  "NonExistentFile.pdf");
```

# onOpen (écouteur d'événement FileReference.onOpen)

onOpen = function(fileRef:FileReference) {}

Appelé au début d'une opération d'envoi ou de téléchargement.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

## Paramètres

*fileRef*:flash.net.FileReference - Objet FileReference qui a initié l'opération.

## Exemple

L'exemple suivant trace un message quand l'événement onOpen est déclenché.

```
import flash.net.FileReference;

var listener:Object = new Object();

listener.onOpen = function(file:FileReference):Void {
  trace("onOpen: " + file.name);
}

var fileRef:FileReference = new FileReference();
fileRef.addListener(listener);
var url:String = "http://www.macromedia.com/platform/whitepapers/
  platform_overview.pdf";
fileRef.download(url, "FlashPlatform.pdf");
```

# onProgress (écouteur d'événement FileReference.onProgress)

onProgress = function(fileRef:FileReference, bytesLoaded:Number, bytesTotal:Number) {}

Appelé régulièrement pendant l'opération d'envoi ou de téléchargement. L'écouteur onProgress est invoqué pendant que Flash Player transmet des octets à un serveur, il est appelé régulièrement pendant la transmission, même si elle ne réussit pas au final. Pour déterminer si la transmission du fichier est entièrement terminée, et à quel moment, utilisez onComplete.

Dans certains cas, les écouteurs onProgress ne sont pas invoqués, par exemple lorsque le fichier transmis est de très petite taille ou lorsque le chargement ou le téléchargement s'effectuent très rapidement.

La progression du chargement d'un fichier ne peut pas être déterminée sur les plate-formes Macintosh avec version antérieure à OS X 10.3. L'événement `onProgress` est appelé au cours du chargement, mais la valeur du paramètre `bytesLoaded` à -1 indique que la progression ne peut pas être déterminée.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

## Paramètres

*`fileRef`*`:flash.net.FileReference` - Objet FileReference qui a initié l'opération.

*`bytesLoaded`*`:Number` - Nombre d'octets transmis jusque là.

*`bytesTotal`*`:Number` - Taille totale du fichier à transmettre, en octets. Si la taille ne peut pas être déterminée, la valeur est -1.

## Exemple

L'exemple suivant trace la progression d'un téléchargement utilisant l'écouteur d'événement `onProgress`.

```
import flash.net.FileReference;

var listener:Object = new Object();

listener.onProgress = function(file:FileReference, bytesLoaded:Number,
  bytesTotal:Number):Void {
  trace("onProgress: " + file.name + " with bytesLoaded: " + bytesLoaded +
  " bytesTotal: " + bytesTotal);
}

var fileRef:FileReference = new FileReference();
fileRef.addListener(listener);
var url:String = "http://www.macromedia.com/platform/whitepapers/
  platform_overview.pdf";
fileRef.download(url, "FlashPlatform.pdf");
```

## Voir également

# onSecurityError (écouteur d'événement FileReference.onSecurityError)

onSecurityError = function(fileRef:FileReference, errorString:String)

Appelé lorsqu'un envoi ou un téléchargement échoue en raison d'une erreur de sécurité. Le fichier SWF effectuant l'appel a peut-être essayé d'accéder à un fichier SWF hors de son domaine et ne dispose pas de l'autorisation requise à cet effet. Vous pouvez tenter de remédier à cette erreur via un fichier de régulation inter-domaines.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

## Paramètres

*fileRef*:flash.net.FileReference - Objet FileReference qui a initié l'opération.

*errorString*:String - Décrit l'erreur à l'origine de l'appel d'onSecurityError. La valeur est « securitySandboxError ».

## Exemple

L'exemple suivant crée un objet FileReference comprenant un écouteur pour chaque événement possible, y compris onSecurityError. L'écouteur onSecurityError est déclenché uniquement en cas d'échec du chargement dû à une erreur de sécurité.

```
import flash.net.FileReference;

var listener:Object = new Object();

listener.onSelect = function(file:FileReference):Void {
  trace("onSelect: " + file.name);
  if(!file.upload("http://www.yourdomain.com/
  yourUploadHandlerScript.cfm")) {
    trace("Upload dialog failed to open.");
  }
}

listener.onCancel = function(file:FileReference):Void {
  trace("onCancel");
}

listener.onOpen = function(file:FileReference):Void {
  trace("onOpen: " + file.name);
}

listener.onProgress = function(file:FileReference, bytesLoaded:Number,
  bytesTotal:Number):Void {
  trace("onProgress with bytesLoaded: " + bytesLoaded + " bytesTotal: " +
  bytesTotal);
```

```
}

listener.onComplete = function(file:FileReference):Void {
  trace("onComplete: " + file.name);
}

listener.onHTTPError = function(file:FileReference):Void {
  trace("onHTTPError: " + file.name);
}

listener.onIOError = function(file:FileReference):Void {
  trace("onIOError: " + file.name);
}

listener.onSecurityError = function(file:FileReference,
  errorString:String):Void {
  trace("onSecurityError: " + file.name + " errorString: " + errorString);
}

var fileRef:FileReference = new FileReference();
fileRef.addListener(listener);
fileRef.browse();
```

# onSelect (écouteur d'événement FileReference.onSelect)

## onSelect = function(fileRef:FileReference) {}

Appelé lorsque l'utilisateur sélectionne un fichier à envoyer ou télécharger dans la boîte de dialogue de recherche de fichiers. (Cette boîte de dialogue est affichée quand vous appelez `FileReference.browse()`, `FileReferenceList.browse()`, ou `FileReference.download()`.) Lorsque l'utilisateur sélectionne un fichier et confirme l'opération (par exemple, en cliquant sur OK), les propriétés de l'objet FileReference sont renseignées.

L'écouteur `onSelect` fonctionne un peu différemment selon la méthode qui l'invoque. Quand `onSelect` est invoqué après un appel `browse()`, Flash Player peut lire toutes les propriétés d'un objet FileReference, parce que le fichier choisi est dans le système local de fichiers. Quand `onSelect` est invoqué après un appel `download()`, Flash Player peut lire uniquement la propriété `name`, parce que le fichier n'a pas encore été téléchargé dans le système local de fichiers au moment où `onSelect` est invoqué. Quand le fichier a été téléchargé et `onComplete` invoqué, alors Flash Player peut lire toutes les autres propriétés de l'objet FileReference.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

### Paramètres

*fileRef*:flash.net.FileReference - Objet FileReference qui a initié l'opération.

### Exemple

L'exemple suivant trace un message dans l'écouteur d'événement onSelect.

```
import flash.net.FileReference;

var listener:Object = new Object();
listener.onSelect = function(file:FileReference):Void {
  trace("onSelect: " + file.name);
  if(!file.upload("http://www.yourdomain.com/
  yourUploadHandlerScript.cfm")) {
    trace("Upload dialog failed to open.");
  }
}

var fileRef:FileReference = new FileReference();
fileRef.addListener(listener);
fileRef.browse();
```

## removeListener (méthodeFileReference.removeListener)

```
public removeListener(listener:Object) : Boolean
```

Supprime un objet de la liste d'objets recevant des messages de notification d'événement.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

### Paramètres

*listener*:Object - Objet qui écoute une notification de rappel venant des écouteurs d'événements FileReference.

### Valeur renvoyée

Boolean - Renvoie true si l'objet spécifié dans le paramètre listener a été enlevé avec succès. Autrement, cette méthode renvoie false.

### Exemple

L'exemple suivant enlève un écouteur d'événement par la méthode removeListener. Si un utilisateur annule le téléchargement, l'écouteur est enlevé pour qu'il ne continue pas à recevoir les événements de l'objet FileReference.

```
import flash.net.FileReference;
```

```
var listener:Object = new Object();

listener.onCancel = function(file:FileReference):Void {
  trace(file.removeListener(this)); // true
}

var fileRef:FileReference = new FileReference();
fileRef.addListener(listener);
var url:String = "http://www.macromedia.com/platform/whitepapers/
  platform_overview.pdf";
fileRef.download(url, "FlashPlatform.pdf");
```

# size (propriété FileReference.size)

```
public size : Number [lecture seule]
```

Taille du fichier sur le disque local en octets. Si l'objet FileReference n'a pas été renseigné, l'appel effectué pour obtenir la valeur de cette propriété renvoie `null`.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

## Exemple

L'exemple suivant récupère la taille d'un fichier choisi par l'utilisateur.

```
import flash.net.FileReference;

var listener:Object = new Object();
listener.onSelect = function(file:FileReference):Void {
  trace("size: " + file.size + " bytes");
}

var fileRef:FileReference = new FileReference();
fileRef.addListener(listener);
fileRef.browse();
```

## Voir également

`browse (méthode FileReference.browse)`

# type (propriété FileReference.type)

```
public type : String [lecture seule]
```

Type de fichier. Sous Windows, cette propriété est l'extension de fichier. Sous Macintosh, cette propriété est le type de fichier à quatre caractères. Si l'objet FileReference n'a pas été renseigné, l'appel effectué pour obtenir la valeur de cette propriété renvoie `null`.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

### Exemple

L'exemple suivant récupère le type d'un fichier choisi par l'utilisateur.

```
import flash.net.FileReference;

var listener:Object = new Object();
listener.onSelect = function(file:FileReference):Void {
  trace("type: " + file.type);
}

var fileRef:FileReference = new FileReference();
fileRef.addListener(listener);
fileRef.browse();
```

### Voir également

`browse (méthode FileReference.browse)`

## upload (méthode FileReference.upload)

```
public upload(url:String) : Boolean
```

Procède au chargement d'un fichier sélectionné par un utilisateur sur un serveur distant. Flash Player peut télécharger des fichiers jusqu'à 100 Mo. Il est nécessaire d'appeler `FileReference.browse()` ou `FileReferenceList.browse()` avant d'appeler cette méthode.

Les écouteurs reçoivent des événements permettant d'indiquer la progression du chargement, s'il a réussi ou échoué. Bien que vous puissiez utiliser l'objet FileReferenceList permettant aux utilisateurs un choix multiple de fichiers à charger, vous devez charger les fichiers un par un. Pour cela, faites une itération dans le tableau `FileReferenceList.fileList` des objets FileReference.

Le fichier est chargé vers l'URL passée dans le paramètre `url`. L'URL doit être un script serveur configuré pour accepter les téléchargements montants. Flash Player télécharge les fichiers en utilisant la méthode HTTP `POST`. Le script serveur qui gère le chargement doit attendre une requête `POST` comportant les éléments suivants :

- Un élément `Content-Type` de `multipart/form-data`
- Un élément `Content-Disposition` avec l'attribut `name` défini sur `"Filedata"` et un attribut `filename` défini sur le nom du fichier d'origine.
- Le contenu binaire actuel du fichier

Voici un exemple de requête `POST` :

```
Content-Type: multipart/form-data; boundary=AaB03x
--AaB03x
Content-Disposition: form-data; name="Filedata"; filename="example.jpg"
```

```
Content-Type: application/octet-stream
... contents of example.jpg ...
--AaB03x--
```

Vous pouvez envoyer des données au serveur avec l'appel `upload()` en ajoutant les paramètres de l'URL.

*Remarque* : Si votre serveur requiert une authentification d'utilisateur, seuls les fichiers SWF s'exécutant dans un navigateur, (c'est-à-dire utilisant le plug-in du navigateur ou le contrôle ActiveX), peuvent fournir une boîte de dialogue pour demander à l'utilisateur un nom et un mot de passe d'authentification, ceci uniquement pour les téléchargements. Concernant les chargements effectués via le plug-in ou le contrôle ActiveX et le chargement/téléchargement via les lecteurs autonomes ou externes, le transfert de fichiers échoue.

Pour utiliser cette méthode, tenez compte du modèle de sécurité de Flash Player :

- Interdit si le fichier SWF appelant est dans une Sandbox locale non sûre.
- Par défaut, l'accès est refusé entre Sandboxes. Un site web peut autoriser l'accès à une ressource en ajoutant un fichier de régulation inter-domaines.

Pour plus d'informations, consultez les sections suivantes :

- Chapitre 17, « Compréhension de la sécurité » dans *Apprentissage d' ActionScript 2.0 dans Flash*
- Le livre blanc sur la sécurité dans Flash Player 8
- Le livre blanc sur la sécurité dans Flash Player 8 en liaison avec les API

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

### Paramètres

*`url`*`:String` - URL du script serveur configuré pour gérer le chargement par appels HTTP `POST`. L'URL peut être de type HTTP ou de type HTTPS pour les chargements sécurisés.

Vous pouvez envoyer des données au serveur avec l'appel `upload()` en ajoutant les paramètres de l'URL ; par ex.`http://www.myserver.com/upload.cgi?userID=jdoe`.

Sur certains navigateurs, les chaînes d'URL ont une longueur limitée. Une longueur supérieure à 256 caractères peut échouer sur certains navigateurs ou serveurs.

### Valeur renvoyée

`Boolean` : une valeur `false` dans l'un des cas suivants :

- `FileReference.browse()` n'a pas été appelé avec succès sur cet objet, ou `FileReferenceList.browse()` n'a pas été appelé avec succès avec cet objet dans son tableau `filelist`.

- Le protocole utilisé n'est pas le protocole HTTP ou HTTPS.
- Une violation de sécurité se produit, c'est-à-dire : si votre fichier SWF essaye d'accéder à un fichier sur un serveur hors de la Sandbox de sécurité de votre fichier SWF.
- Le type ou le format du paramètre `url` n'est pas correct.
- L'appel ne comporte pas le nombre requis de paramètres.

## Evénements

onCancel

Invoqué lorsque l'utilisateur ferme la boîte de dialogue.

onComplete

Appelé en cas de réussite de l'opération de téléchargement montant.

onHTTPError

Appelé lorsqu'un envoi échoue en raison d'une erreur HTTP.

onIOError

Invoqué dans l'un des cas suivants :

- Le chargement échoue en raison d'une erreur d'entrée/sortie lors de la lecture, l'écriture ou la transmission du fichier par Flash Player.
- Le fichier SWF tente de charger un fichier sur un serveur nécessitant une authentification (comme un nom d'utilisateur et un mot de passe). Au cours du chargement, Flash Player ne permet pas aux utilisateurs d'entrer des mots de passe.
- Le chargement échoue parce que le paramètre `url` renferme un protocole invalide. `FileReference.upload()` doit utiliser HTTP ou HTTPS.

onOpen

Invoqué lors du démarrage d'un téléchargement montant.

onProgress

Appelé régulièrement pendant le téléchargement du fichier.

onSecurityError

Appelé lorsqu'un envoi ou un téléchargement échoue en raison d'une erreur de sécurité.

## Exemple

L'exemple suivant montre l'implémentation de la méthode `upload()` en informant d'abord l'utilisateur de choisir un fichier à télécharger, puis en gérant les écouteurs `onSelect` et `onCancel`, pour finalement gérer les résultats du téléchargement en cours.

```
import flash.net.FileReference;

var allTypes:Array = new Array();
var imageTypes:Object = new Object();
imageTypes.description = "Images (*.jpg, *.jpeg, *.gif, *.png)";
imageTypes.extension = "*.jpg; *.jpeg; *.gif; *.png";
allTypes.push(imageTypes);

var listener:Object = new Object();

listener.onSelect = function(file:FileReference):Void {
  trace("onSelect: " + file.name);
  if(!file.upload("http://www.yourdomain.com/
  yourUploadHandlerScript.cfm")) {
    trace("Upload dialog failed to open.");
  }
}

listener.onCancel = function(file:FileReference):Void {
  trace("onCancel");
}

listener.onOpen = function(file:FileReference):Void {
  trace("onOpen: " + file.name);
}

listener.onProgress = function(file:FileReference, bytesLoaded:Number,
  bytesTotal:Number):Void {
  trace("onProgress with bytesLoaded: " + bytesLoaded + " bytesTotal: " +
  bytesTotal);
}

listener.onComplete = function(file:FileReference):Void {
  trace("onComplete: " + file.name);
}

listener.onHTTPError = function(file:FileReference):Void {
  trace("onHTTPError: " + file.name);
}

listener.onIOError = function(file:FileReference):Void {
  trace("onIOError: " + file.name);
```

```
}

listener.onSecurityError = function(file:FileReference,
  errorString:String):Void {
  trace("onSecurityError: " + file.name + " errorString: " + errorString);
}

var fileRef:FileReference = new FileReference();
fileRef.addListener(listener);
fileRef.browse(allTypes);
```

### Voir également

browse (méthode FileReference.browse), browse (méthode FileReferenceList.browse), download (méthode FileReference.download), fileList (propriété FileReferenceList.fileList)

# FileReferenceList (flash.net.FileReferenceList)

```
Object
  |
  +-flash.net.FileReferenceList
```

`public class` ***FileReferenceList***
`extends Object`

La classe FileReferenceList permet aux utilisateurs de sélectionner un ou plusieurs fichiers à charger. Un objet FileReferenceList représente un groupe de fichiers locaux présents sur le disque de l'utilisateur sous forme de tableau d'objets FileReference. Pour obtenir les informations détaillées et les principales caractéristiques relatives aux objets et à la classe FileReference, que vous utilisez avec FileReferenceList, consultez la classe FileReference.

Pour utiliser la classe FileReferenceList :

- Instanciez la classe : `var myFileRef = new FileReferenceList();`
- Appelez `FileReferenceList.browse()`, pour afficher une boîte de dialogue dans laquelle l'utilisateur peut choisir un ou plusieurs fichiers à télécharger : `myFileRef.browse();`
- Une fois la méthode `browse()` appelée avec succès, la propriété `fileList` de l'objet FileReferenceList est renseignée à l'aide d'un tableau d'objets FileReference.
- Appelez `FileReference.upload()` sur chaque élément du tableau `fileList`.

La classe FileReferenceList inclut une méthode `browse()` et une propriété `fileList` pour travailler avec plusieurs fichiers.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

## Exemple

L'exemple suivant permet à l'utilisateur de choisir plusieurs fichiers et ensuite de charger chacun d'entre eux sur un serveur.

```
import flash.net.FileReferenceList;
import flash.net.FileReference;

var listener:Object = new Object();

listener.onSelect = function(fileRefList:FileReferenceList) {
  trace("onSelect");
  var list:Array = fileRefList.fileList;
  var item:FileReference;
  for(var i:Number = 0; i < list.length; i++) {
    item = list[i];
    trace("name: " + item.name);
    trace(item.addListener(this));
    item.upload("http://www.yourdomain.com/");
  }
}

listener.onCancel = function():Void {
  trace("onCancel");
}

listener.onOpen = function(file:FileReference):Void {
  trace("onOpen: " + file.name);
}

listener.onProgress = function(file:FileReference, bytesLoaded:Number,
  bytesTotal:Number):Void {
  trace("onProgress with bytesLoaded: " + bytesLoaded + " bytesTotal: " +
  bytesTotal);
}

listener.onComplete = function(file:FileReference):Void {
  trace("onComplete: " + file.name);
}

listener.onHTTPError = function(file:FileReference, httpError:Number):Void
  {
  trace("onHTTPError: " + file.name + " httpError: " + httpError);
}
```

```
listener.onIOError = function(file:FileReference):Void {
  trace("onIOError: " + file.name);
}

listener.onSecurityError = function(file:FileReference,
  errorString:String):Void {
  trace("onSecurityError: " + file.name + " errorString: " + errorString);
}

var fileRef:FileReferenceList = new FileReferenceList();
fileRef.addListener(listener);
fileRef.browse();
```

## Voir également

FileReference (flash.net.FileReference)

## Résumé des propriétés

| Modificateurs | Propriété | Description |
|---|---|---|
| | `fileList:Array` | Un tableau d'objets FileReference. |

*Propriétés héritées de la classe Object*

constructeur (propriété Object.constructor), __proto__ (Object.__proto__ property), prototype (Object.prototype, propriété), __resolve (Object.__resolve, propriété)

## Résumé des événements

| Evénement | Description |
|---|---|
| `onCancel = function(fileReferenceList:FileReferenceList) {}` | Appelé lorsque l'utilisateur ferme la boîte de dialogue de recherche de fichier. |
| `onSelect = function(fileReferenceList:FileReferenceList) {}` | Appelé lorsque l'utilisateur sélectionne un ou plusieurs fichiers à télécharger à partir de la boîte de dialogue de recherche de fichiers. |

## Récapitulatif des constructeurs

| Signature | Description |
|---|---|
| `FileReferenceList()` | Crée un nouvel objet FileReferenceList. |

Résumé de la méthode

| Modificateurs | Signature | Description |
|---|---|---|
| | `addListener(listener:Object) : Void` | Enregistre un objet pour recevoir une notification lorsqu'un écouteur d'événements FileReferenceList est invoqué. |
| | `browse([typelist:Array]) : Boolean` | Affiche une boîte de dialogue de recherche de fichier dans laquelle l'utilisateur peut sélectionner un ou plusieurs fichiers locaux à envoyer. |
| | `removeListener(listener:Object) : Boolean` | Supprime un objet de la liste d'objets recevant des messages de notification d'événement. |

*Méthodes héritées de la classe Object*

`addProperty (méthode Object.addProperty), hasOwnProperty (méthode Object.hasOwnProperty), isPropertyEnumerable (méthode Object.isPropertyEnumerable), isPrototypeOf (méthode Object.isPrototypeOf), registerClass (méthode Object.registerClass), toString (méthode Object.toString), unwatch (méthode Object.unwatch), valueOf (méthode Object.valueOf), watch (méthode Object.watch)`

# addListener (méthode FileReferenceList.addListener)

```
public addListener(listener:Object) : Void
```

Enregistre un objet pour recevoir une notification lorsqu'un écouteur d'événements FileReferenceList est invoqué.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

## Paramètres

*`listener`*`:Object` - Objet qui écoute une notification de rappel venant des écouteurs d'événements FileReferenceList.

## Exemple

L'exemple suivant illustre la méthode `addListener()`.

```
import flash.net.FileReferenceList;

var listener:Object = new Object();
listener.onCancel = function(fileRefList:FileReferenceList) {
  trace("onCancel");
```

```
}

listener.onSelect = function(fileRefList:FileReferenceList) {
  trace("onSelect: " + fileRefList.fileList.length);
}

var fileRef:FileReferenceList = new FileReferenceList();
fileRef.addListener(listener);
fileRef.browse();
```

# browse (méthode FileReferenceList.browse)

```
public browse([typelist:Array]) : Boolean
```

Affiche une boîte de dialogue de recherche de fichier dans laquelle l'utilisateur peut sélectionner un ou plusieurs fichiers locaux à envoyer. La boîte de dialogue est spécifique au système d'exploitation de l'utilisateur. Lorsque vous appelez cette méthode et que l'utilisateur réussit à sélectionner les fichiers, la propriété `fileList` de cet objet FileReferenceList est renseignée par un tableau d'objets FileReference, à savoir un pour chaque fichier sélectionné par l'utilisateur. Toutes les fois suivantes où `FileReferenceList.browse()` est appelé, les propriétés de l'objet FileReferenceList sont restaurées pour le ou les fichiers choisis par l'utilisateur dans la boîte de dialogue.

Vous pouvez remplir un tableau des types de fichiers pour déterminer lesquels seront affichés par la boîte de dialogue.

Une seule session `browse()` ou `download()` peut être effectuée à la fois sur un objet FileReferenceList (car une seule boîte de dialogue peut être appelée à la fois).

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

## Paramètres

*`typelist`*`:Array` [facultatif] - Tableau de types de fichiers utilisés pour filtrer les fichiers qui s'affichent dans la boîte de dialogue. Si vous omettez ce paramètre, tous les fichiers s'affichent. Si vous incluez ce paramètre, le tableau doit contenir un ou plusieurs éléments placés entre accolades { }. Vous pouvez appliquer l'un des deux formats au tableau :

- Une liste de types de fichiers suivis de leur extension Windows uniquement. Chaque élément du tableau doit contenir une chaîne décrivant le type de fichier et une liste, séparée par des points-virgules, des extensions de fichiers Windows, chaque extension étant précédée d'un caractère joker (*). La syntaxe de chaque élément se présente comme suit : `[{description: "string describing the first set of file types", extension: "semicolon-delimited list of file extensions"}]`
  Exemple :
  ```
  [{description: "Images", extension: "*.jpg;*.gif;*.png"}, {description:
  "Flash Movies", extension: "*.swf"}, {description: "Documents",
  extension: "*.doc;*.pdf"}]
  ```
- Une liste de types de fichiers suivis de leur extension Windows et de leur type de fichier Macintosh. Chaque élément du tableau doit contenir une chaîne décrivant le type de fichier ; une liste, séparée par des points-virgules, des extensions de fichiers Windows, chaque extension étant précédée d'un caractère joker (*) et une liste, séparée par des points-virgules, des types de fichiers Macintosh, chaque type étant précédé d'un caractère joker (*). La syntaxe de chaque élément se présente comme suit :
  ```
  [{description: "string describing the first set of file types",
  extension: "semicolon-delimited list of Windows file extensions",
  macType: "semicolon-delimited list of Macintosh file types"}]
  ```
  Exemple :
  ```
  [{description: "Image files", extension: "*.jpg;*.gif;*.png", macType:
  "JPEG;jp2_;GIFF"}, {description: "Flash Movies", extension: "*.swf",
  macType: "SWFL"}]
  ```

Les deux formats ne sont pas interchangeables dans un appel `browse()` unique. Vous devez utiliser l'un ou l'autre.

La liste des extensions est utilisée pour filtrer les fichiers sous Windows, selon le type de fichier sélectionné par l'utilisateur. Elle n'est pas réellement affichée dans la boîte de dialogue. Pour afficher les types de fichiers aux utilisateurs, vous devez lister les types de fichiers dans la chaîne descriptive ainsi que dans la liste d'extensions. Sous Windows, la chaîne descriptive est affichée dans la boîte de dialogue. (Elle n'est pas utilisée sous Macintosh.) Sous Macintosh, une liste de types de fichier Macintosh est utilisée pour filtrer les fichiers, si elle est fournie. Si vous ne fournissez pas de liste de types de fichiers Macintosh, c'est la liste d'extensions Windows qui est utilisée.

## Valeur renvoyée

`Boolean` - Renvoie `true` si les paramètres sont valides et que la boîte de dialogue de recherche de fichiers est affichée. Renvoie `false` si la boîte de dialogue ne s'affiche pas, si une autre session browse est déjà en cours ou si vous utilisez le paramètre `typelist` sans fournir une description ou une chaîne d'extension concernant un élément du tableau.

## Evénements

onCancel

Invoqué quand l'utilisateur quitte la boîte de dialogue en cliquant sur Annuler ou en la fermant.

onSelect

Invoqué quand l'utilisateur réussit à choisir un élément à télécharger dans la boîte de dialogue.

## Exemple

L'exemple suivant illustre la méthode `browse()`.

```
import flash.net.FileReferenceList;

var allTypes:Array = new Array();
var imageTypes:Object = new Object();
imageTypes.description = "Images (*.JPG;*.JPEG;*.JPE;*.GIF;*.PNG;)";
imageTypes.extension = "*.jpg; *.jpeg; *.jpe; *.gif; *.png;";
allTypes.push(imageTypes);

var textTypes:Object = new Object();
textTypes.description = "Text Files (*.TXT;*.RTF;)";
textTypes.extension = "*.txt; *.rtf";
allTypes.push(textTypes);

var fileRef:FileReferenceList = new FileReferenceList();
fileRef.browse(allTypes);
```

## Voir également

`browse (méthode FileReference.browse)`, `FileReference (flash.net.FileReference)`

# fileList (propriété FileReferenceList.fileList)

`public fileList : Array`

Un tableau d'objets FileReference.

Quand la méthode `FileReferenceList.browse()` a été appelée et que l'utilisateur a sélectionné un ou plusieurs fichiers à partir de la boîte de dialogue ouverte par `browse()`, cette propriété est renseignée par un tableau d'objets FileReference , chacun d'entre eux représentant un fichier que l'utilisateur a sélectionné. Vous pouvez ensuite utiliser ce tableau pour charger les fichiers via la méthode `FileReference.upload()`. Vous devez charger les fichiers un par un.

La propriété `fileList` est renseignée à nouveau à chaque fois que `browse()` est appelé sur cet objet FileReferenceList.

Les propriétés des objets FileReference sont décrites dans la documentation relative à la classe FileReference.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

## Exemple

L'exemple suivant illustre la propriété `fileList`.

```
import flash.net.FileReferenceList;
import flash.net.FileReference;

var listener:Object = new Object();
listener.onSelect = function(fileRefList:FileReferenceList) {
  trace("onSelect");
  var list:Array = fileRefList.fileList;
  var item:FileReference;
  for(var i:Number = 0; i < list.length; i++) {
    item = list[i];
    trace("name: " + item.name);
  }
}

var fileRef:FileReferenceList = new FileReferenceList();
fileRef.addListener(listener);
fileRef.browse();
```

## Voir également

`FileReference (flash.net.FileReference)`, `upload (méthode FileReference.upload)`, `browse (méthode FileReferenceList.browse)`

# constructeur FileReferenceList

```
public FileReferenceList()
```

Crée un nouvel objet FileReferenceList. Cet objet ne contient rien jusqu'à ce que vous appeliez `browse()` sur lui. Quand vous appelez `browse()` sur l'objet FileReference, la propriété `fileList` de l'objet est renseignée par un tableau d'objets FileReference.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

## Exemple

L'exemple suivant crée un nouvel objet `FileReferenceList`, effectue une itération sur chaque fichier sélectionné et sort leurs noms.

```
import flash.net.FileReferenceList;

var listener:Object = new Object();
listener.onSelect = function(fileRefList:FileReferenceList) {
  trace("onSelect");
  var arr:Array = fileRefList.fileList;
  for(var i:Number = 0; i < arr.length; i++) {
    trace("name: " + arr[i].name);
  }
}

var fileRef:FileReferenceList = new FileReferenceList();
fileRef.addListener(listener);
fileRef.browse();
```

## Voir également

`FileReference (flash.net.FileReference)`, `browse (méthode FileReferenceList.browse)`

# onCancel (écouteur d'événement FileReferenceList.onCancel)

```
onCancel = function(fileRefList:FileReferenceList) {}
```

Appelé lorsque l'utilisateur ferme la boîte de dialogue de recherche de fichier. (Cette boîte de dialogue est affichée quand vous appelez les méthodes `FileReferenceList.browse()`, `FileReference.browse()`, ou `FileReference.download()`.)

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

### Paramètres

*fileRefList*:flash.net.FileReferenceList - Objet FileReferenceList qui a initié l'opération.

### Exemple

L'exemple suivant illustre l'écouteur onCancel.

```
import flash.net.FileReferenceList;

var listener:Object = new Object();
listener.onCancel = function(fileRefList:FileReferenceList) {
  trace("onCancel");
}

var fileRef:FileReferenceList = new FileReferenceList();
fileRef.addListener(listener);
fileRef.browse();
```

### Voir également

browse (méthode FileReferenceList.browse)

## onSelect (écouteur d'événement FileReferenceList.onSelect)

onSelect = function(fileRefList:FileReferenceList) {}

Appelé lorsque l'utilisateur sélectionne un ou plusieurs fichiers à télécharger à partir de la boîte de dialogue de recherche de fichiers. (Cette boîte de dialogue est affichée quand vous appelez les méthodes FileReferenceList.browse(), FileReference.browse(), ou FileReference.download().) Lorsque l'utilisateur sélectionne un fichier et confirme l'opération (par exemple, en cliquant sur Enregistrer), l'objet FileReferenceList est renseigné par les objets FileReference représentant les fichiers sélectionnés par l'utilisateur.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

### Paramètres

*fileRefList*:flash.net.FileReferenceList - Objet FileReferenceList qui a initié l'opération.

### Exemple

L'exemple suivant illustre l'écouteur onSelect.

```
import flash.net.FileReferenceList;
import flash.net.FileReference;
```

```
var listener:Object = new Object();

listener.onSelect = function(fileRefList:FileReferenceList) {
  trace("onSelect");
  var list:Array = fileRefList.fileList;
  var item:FileReference;
  for(var i:Number = 0; i < list.length; i++) {
    item = list[i];
    trace("name: " + item.name);
    trace(item.addListener(this));
    item.upload("http://www.yourdomain.com/");
  }
}

listener.onComplete = function(file:FileReference):Void {
  trace("onComplete: " + file.name);
}

var fileRef:FileReferenceList = new FileReferenceList();
fileRef.addListener(listener);
fileRef.browse();
```

### Voir également

`browse (méthode FileReferenceList.browse)`

## removeListener (méthode FileReferenceList.removeListener)

`public removeListener(listener:Object) : valeur booléenne`

Supprime un objet de la liste d'objets recevant des messages de notification d'événement.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

### Paramètres

*`listener`*`:Object` - Objet qui écoute une notification de rappel à partir des écouteurs d'événements FileReferenceList.

### Valeur renvoyée

`Boolean` - Renvoie `true` si l'objet est supprimé. Dans le cas contraire, cette méthode renvoie `false`.

### Exemple

L'exemple suivant illustre la méthode `removeListener`.

```
import flash.net.FileReferenceList;

var listener:Object = new Object();
listener.onCancel = function(fileRefList:FileReferenceList) {
  trace("onCancel");
  trace(fileRefList.removeListener(this)); // true
}

listener.onSelect = function(fileRefList:FileReferenceList) {
  trace("onSelect: " + fileRefList.fileList.length);
}

var fileRef:FileReferenceList = new FileReferenceList();
fileRef.addListener(listener);
fileRef.browse();
```

# Function

```
Object
  |
  +-Function
```

`public dynamic class` ***Function***
`extends Object`

Les fonctions définies par l'utilisateur et les fonctions intégrées dans ActionScript sont représentées par des objets Function, qui sont des instances de la classe Function.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

### Résumé des propriétés

*Propriétés héritées de la classe Object*

| |
|---|
| constructeur (propriété Object.constructor), __proto__ (Object.__proto__ property), prototype (Object.prototype, propriété), __resolve (Object.__resolve, propriété) |

## Résumé de la méthode

| Modificateurs | Signature | Description |
|---|---|---|
| | `apply(thisObject:Object, [argArray:Array]) : Void` | Spécifie la valeur `thisObject` à utiliser dans toute fonction appelée par ActionScript. |
| | `call(thisObject:Object, [parameter1:Object]) : Object` | Appelle la fonction représentée par un objet Function. |

*Méthodes héritées de la classe Object*

`addProperty (méthode Object.addProperty), hasOwnProperty (méthode Object.hasOwnProperty), isPropertyEnumerable (méthode Object.isPropertyEnumerable), isPrototypeOf (méthode Object.isPrototypeOf), registerClass (méthode Object.registerClass), toString (méthode Object.toString), unwatch (méthode Object.unwatch), valueOf (méthode Object.valueOf), watch (méthode Object.watch)`

# apply (méthode Function.apply)

`public apply(thisObject:Object, [argArray:Array]) : Void`

Spécifie la valeur `thisObject` à utiliser dans toute fonction appelée par ActionScript. Cette méthode spécifie également les paramètres à transmettre à toute fonction appelée. Dans la mesure où `apply()` est une méthode de la classe Function, c'est également une méthode de chaque objet Function dans ActionScript.

Les paramètres sont spécifiés sous forme d'objet Array, contrairement à `Function.call()` qui spécifie les paramètres en tant que liste délimitée par des virgules. Ceci est souvent utile lorsque le nombre de paramètres à transmettre n'est pas connu avant l'exécution du script.

Renvoie la valeur spécifiée en tant que valeur renvoyée par la fonction appelée.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

## Paramètres

*`thisObject`*`:Object` - Objet auquel myFunction s'applique.

*`argArray`*`:Array` [facultatif] - Tableau dont les éléments sont transmis à myFunction en tant que paramètres.

## Exemple

Les invocations de fonction suivantes sont équivalentes :

```
Math.atan2(1, 0)
Math.atan2.apply(null, [1, 0])
```

L'exemple simple suivant illustre la façon dont la méthode `apply()` transmet un tableau de paramètres :

```
function theFunction() {
  trace(arguments);
}

// create a new array to pass as a parameter to apply()
var firstArray:Array = new Array(1,2,3);
theFunction.apply(null,firstArray);
// outputs: 1,2,3

// create a second array to pass as a parameter to apply()
var secondArray:Array = new Array("a", "b", "c");
theFunction.apply(null,secondArray);
// outputs a,b,c
```

L'exemple suivant illustre la façon dont la méthode `apply()` transmet un tableau de paramètres et spécifie la valeur this :

```
// define a function
function theFunction() {
  trace("this == myObj? " + (this == myObj));
  trace("arguments: " + arguments);
}

// instantiate an object
var myObj:Object = new Object();

// create arrays to pass as a parameter to apply()
var firstArray:Array = new Array(1,2,3);
var secondArray:Array = new Array("a", "b", "c");

// use apply() to set the value of this to be myObj and send firstArray
theFunction.apply(myObj,firstArray);
// output:
// this == myObj? true
// arguments: 1,2,3

// use apply() to set the value of this to be myObj and send secondArray
theFunction.apply(myObj,secondArray);
// output:
// this == myObj? true
// arguments: a,b,c
```

### Voir également

`call (méthode Function.call)`

## call (méthode Function.call)

```
public call(thisObject:Object, [parameter1:Object]) : Object
```

Appelle la fonction représentée par un objet Function. Toutes les fonctions dans ActionScript sont représentées par un objet Function, de sorte que toutes les fonctions prennent en charge cette méthode.

Dans presque tous les cas, l'opérateur d'appel de fonction (`()`) peut être utilisé au lieu de cette méthode. L'opérateur de la fonction call génère un code concis et lisible. Cette méthode est surtout utile lorsque le paramètre `thisObject` de l'invocation de fonction doit être explicitement contrôlé. Normalement, si une fonction est invoquée en tant que méthode d'un objet, dans le corps de la fonction, `thisObject` est défini sur `myObject`, comme illustré dans l'exemple suivant :

```
myObject.myMethod(1, 2, 3);
```

Dans certains cas, vous voudrez peut-être que `thisObject` pointe autre part ; par exemple, si une fonction doit être invoquée en tant que méthode d'un objet alors qu'elle n'est pas stockée comme méthode de cet objet :

```
myObject.myMethod.call(myOtherObject, 1, 2, 3);
```

Vous pouvez transmettre la valeur null pour le paramètre `thisObject` pour invoquer une fonction en tant que fonction ordinaire et non en tant que méthode d'un objet. Par exemple, les invocations de fonction suivantes sont équivalentes :

```
Math.sin(Math.PI / 4)
Math.sin.call(null, Math.PI / 4)
```

Renvoie la valeur spécifiée en tant que valeur renvoyée par la fonction appelée.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

### Paramètres

*`thisObject`*`:Object` - Objet qui spécifie la valeur de `thisObject` dans le corps de la fonction.

*`parameter1`*`:Object` [facultatif] - Paramètre à transmettre à myFunction. Vous pouvez spécifier zéro ou plusieurs paramètres.

### Valeur renvoyée

`Object` -

### Exemple

L'exemple suivant utilise `Function.call()` pour qu'une fonction adopte le comportement d'une méthode d'un autre objet, sans enregistrer la fonction dans l'objet :

```
function myObject() {
}
function myMethod(obj) {
  trace("this == obj? " + (this == obj));
}
var obj:Object = new myObject();
myMethod.call(obj, obj);
```

L'instruction `trace()` affiche :

```
this == obj? true
```

### Voir également

`apply (méthode Function.apply)`

# GlowFilter (flash.filters.GlowFilter)

```
Object
  |
  +-flash.filters.BitmapFilter
    |
    +-flash.filters.GlowFilter
```

`public class` ***GlowFilter***
`extends BitmapFilter`

La classe GlowFilter permet d'appliquer un effet de rayonnement à divers objets dans Flash. Vous disposez de plusieurs options pour définir le style de rayonnement, notamment le rayonnement interne ou externe et le mode de masquage. Le filtre de rayonnement est similaire au filtre d'ombre portée dont les propriétés `distance` et `angle` sont définies sur zéro.

L'utilisation de filtres dépend de l'objet auquel vous appliquez le filtre.

- Pour appliquer des filtres aux clips, champs de texte et boutons lors de l'exécution, utilisez la propriété `filters`. Lorsque vous définissez la propriété `filters` d'un objet, celui-ci n'est pas modifié. En outre, vous pouvez l'annuler en supprimant la propriété `filters`.
- Pour appliquer des filtres aux occurrences BitmapData, utilisez la méthode `BitmapData.applyFilter()`. L'appel de `applyFilter()` sur un objet BitmapData génère une image filtrée à partir de l'objet BitmapData source et de l'objet Filter.

Vous pouvez également appliquer des effets de filtre aux images et aux données vidéo durant la programmation. Pour plus d'informations, consultez la documentation relative à la programmation.

Si vous appliquez un filtre à un clip ou à un bouton, la propriété `cacheAsBitmap` du clip ou du bouton est définie sur `true`. Si vous supprimez tous les filtres, la valeur d'origine de `cacheAsBitmap` est restaurée.

Ce filtre prend en charge le redimensionnement de la scène. Toutefois, le redimensionnement général, la rotation et l'inclinaison ne sont pas pris en charge ; si l'objet est redimensionné (si le redimensionnement `_xscale` et le redimensionnement `_yscale` ne sont pas définis sur 100 %), l'effet de filtre n'est pas redimensionné. Le redimensionnement est effectué uniquement en cas de zoom sur la scène.

Un filtre ne sera pas appliqué si l'image obtenue dépasse 2 880 pixels en largeur ou en hauteur. Par exemple, lorsque vous effectuez un zoom avant sur un clip de grande taille auquel un filtre est appliqué, le filtre sera désactivé si l'image obtenue dépasse la limite de 2 880 pixels.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

## Voir également

`applyFilter (méthode BitmapData.applyFilter)`, `cacheAsBitmap (propriété Button.cacheAsBitmap)`, `filters (propriété Button.filters)`, `DropShadowFilter (flash.filters.DropShadowFilter)`, `cacheAsBitmap (propriété MovieClip.cacheAsBitmap)`, `filters (propriété MovieClip.filters)`, `filters (propriété TextField.filters)`

## Résumé des propriétés

| Modificateurs | Propriété | Description |
|---|---|---|
| | `alpha:Number` | La valeur de transparence alpha de la couleur. |
| | `blurX:Number` | Le niveau de flou horizontal. |
| | `blurY:Number` | Le niveau de flou vertical. |
| | `color:Number` | La couleur du rayonnement. |
| | `inner:Boolean` | Spécifie si le rayonnement est interne. |
| | `knockout:Boolean` | Spécifie si l'objet a un effet de poinçonnage. |
| | `quality:Number` | Le nombre d'applications du filtre. |
| | `strength:Number` | L'intensité de l'impression ou du recouvrement. |

*Propriétés héritées de la classe Object*

`constructeur (propriété Object.constructor)`, `__proto__ (Object.__proto__ property)`, `prototype (Object.prototype, propriété)`, `__resolve (Object.__resolve, propriété)`

### Récapitulatif des constructeurs

| Signature | Description |
|---|---|
| `GlowFilter([color:Number], [alpha:Number], [blurX:Number], [blurY:Number], [strength:Number], [quality:Number], [inner:Boolean], [knockout:Boolean])` | Initialise une nouvelle instance GlowFilter avec les paramètres spécifiés. |

### Résumé de la méthode

| Modificateurs | Signature | Description |
|---|---|---|
| | `clone() : GlowFilter` | Renvoie une copie de cet objet filtre. |

*Méthodes héritées de la classe BitmapFilter*

`clone (méthode BitmapFilter.clone )`

*Méthodes héritées de la classe Object*

`addProperty (méthode Object.addProperty), hasOwnProperty (méthode Object.hasOwnProperty), isPropertyEnumerable (méthode Object.isPropertyEnumerable), isPrototypeOf (méthode Object.isPrototypeOf), registerClass (méthode Object.registerClass), toString (méthode Object.toString), unwatch (méthode Object.unwatch), valueOf (méthode Object.valueOf), watch (méthode Object.watch)`

## alpha (propriété GlowFilter.alpha)

`public alpha : nombre`

La valeur de transparence alpha de la couleur. Les valeurs valides sont comprises entre 0 et 1. Par exemple, 0,25 définit une valeur de transparence de 25 %. La valeur par défaut est 1.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

### Exemple

L'exemple suivant modifie la propriété `alpha` sur un clip existant lorsqu'un utilisateur clique dessus.

```
import flash.filters.GlowFilter;
```

```
var mc:MovieClip = createGlowFilterRectangle("GlowFilterAlpha");
mc.onRelease = function() {
  var filter:GlowFilter = this.filters[0];
  filter.alpha = .4;
  this.filters = new Array(filter);
}

function createGlowFilterRectangle(name:String):MovieClip {
  var rect:MovieClip = this.createEmptyMovieClip(name,
  this.getNextHighestDepth());
  var w:Number = 100;
  var h:Number = 100;
  rect.beginFill(0x003366);
  rect.lineTo(w, 0);
  rect.lineTo(w, h);
  rect.lineTo(0, h);
  rect.lineTo(0, 0);
  rect._x = 20;
  rect._y = 20;

  var filter:GlowFilter = new GlowFilter(0x000000, .8, 16, 16, 1, 3, false,
  false);
  var filterArray:Array = new Array();
  filterArray.push(filter);
  rect.filters = filterArray;
  return rect;
}
```

# blurX (propriété GlowFilter.blurX)

```
public blurX : nombre
```

Le niveau de flou horizontal. Les valeurs valides sont comprises entre 0 et 255 (virgule flottante). La valeur par défaut est 6. Les valeurs correspondant à une puissance de 2 (2, 4, 8, 16 et 32) sont optimisées pour obtenir un rendu plus rapide qu'avec les autres valeurs.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

## Exemple

L'exemple suivant modifie la propriété `blurX` sur un clip existant lorsqu'un utilisateur clique dessus.

```
import flash.filters.GlowFilter;

var mc:MovieClip = createGlowFilterRectangle("GlowFilterBlurX");
mc.onRelease = function() {
  var filter:GlowFilter = this.filters[0];
  filter.blurX = 20;
```

```
  this.filters = new Array(filter);
}

function createGlowFilterRectangle(name:String):MovieClip {
  var rect:MovieClip = this.createEmptyMovieClip(name,
  this.getNextHighestDepth());
  var w:Number = 100;
  var h:Number = 100;
  rect.beginFill(0x003366);
  rect.lineTo(w, 0);
  rect.lineTo(w, h);
  rect.lineTo(0, h);
  rect.lineTo(0, 0);
  rect._x = 20;
  rect._y = 20;

  var filter:GlowFilter = new GlowFilter(0x000000, .8, 16, 16, 1, 3, false,
  false);
  var filterArray:Array = new Array();
  filterArray.push(filter);
  rect.filters = filterArray;
  return rect;
}
```

# blurY (propriété GlowFilter.blurY)

```
public blurY : nombre
```

Le niveau de flou vertical. Les valeurs valides sont comprises entre 0 et 255 (virgule flottante). La valeur par défaut est 6. Les valeurs correspondant à une puissance de 2 (2, 4, 8, 16 et 32) sont optimisées pour obtenir un rendu plus rapide qu'avec les autres valeurs.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

## Exemple

L'exemple suivant modifie la propriété `blurY` sur un clip existant lorsqu'un utilisateur clique dessus.

```
import flash.filters.GlowFilter;

var mc:MovieClip = createGlowFilterRectangle("GlowFilterBlurY");
mc.onRelease = function() {
  var filter:GlowFilter = this.filters[0];
  filter.blurY = 20;
  this.filters = new Array(filter);
}

function createGlowFilterRectangle(name:String):MovieClip {
```

```
  var rect:MovieClip = this.createEmptyMovieClip(name,
  this.getNextHighestDepth());
  var w:Number = 100;
  var h:Number = 100;
  rect.beginFill(0x003366);
  rect.lineTo(w, 0);
  rect.lineTo(w, h);
  rect.lineTo(0, h);
  rect.lineTo(0, 0);
  rect._x = 20;
  rect._y = 20;

  var filter:GlowFilter = new GlowFilter(0x000000, .8, 16, 16, 1, 3, false,
  false);
  var filterArray:Array = new Array();
  filterArray.push(filter);
  rect.filters = filterArray;
  return rect;
}
```

# clone (méthode GlowFilter.clone)

```
public clone() : GlowFilter
```

Renvoie une copie de cet objet filtre.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

## Valeur renvoyée

`flash.filters.GlowFilter` - Nouvelle instance GlowFilter dont toutes les propriétés sont identiques à celles de l'instance GlowFilter d'origine.

## Exemple

L'exemple suivant crée trois objets GlowFilter et les compare : `filter_1` est créé à l'aide du constructeur GlowFilter ; `filter_2` est créé en le définissant comme égal à `filter_1` et `clonedFilter` est créé par clonage de `filter_1`. Vous remarquerez que si `filter_2` est évalué comme égal à `filter_1`, ce n'est pas le cas de `clonedFilter`, qui contient pourtant les mêmes valeurs que `filter_1`.

```
import flash.filters.GlowFilter;

var filter_1:GlowFilter = new GlowFilter(0x33CCFF, .8, 35, 35, 2, 3, false,
  false);
var filter_2:GlowFilter = filter_1;
var clonedFilter:GlowFilter = filter_1.clone();

trace(filter_1 == filter_2); // true
```

```
trace(filter_1 == clonedFilter); // false

for(var i in filter_1) {
  trace(">> " + i + ": " + filter_1[i]);
  // >> clone: [type Function]
  // >> strength: 2
  // >> blurY: 35
  // >> blurX: 35
  // >> knockout: false
  // >> inner: false
  // >> quality: 3
  // >> alpha: 0.8
  // >> color: 3394815
}

for(var i in clonedFilter) {
  trace(">> " + i + ": " + clonedFilter[i]);
  // >> clone: [type Function]
  // >> strength: 2
  // >> blurY: 35
  // >> blurX: 35
  // >> knockout: false
  // >> inner: false
  // >> quality: 3
  // >> alpha: 0.8
  // >> color: 3394815
}
```

Pour illustrer plus précisément la relation entre `filter_1`, `filter_2` et `clonedFilter`, l'exemple suivant modifie la propriété `knockout` de `filter_1`. La modification de `knockout` démontre que la méthode `clone()` crée une nouvelle occurrence reposant sur les valeurs de `filter_1` au lieu de pointer vers elles par référence.

```
import flash.filters.GlowFilter;

var filter_1:GlowFilter = new GlowFilter(0x33CCFF, .8, 35, 35, 2, 3, false,
  false);
var filter_2:GlowFilter = filter_1;
var clonedFilter:GlowFilter = filter_1.clone();

trace(filter_1.knockout); // false
trace(filter_2.knockout); // false
trace(clonedFilter.knockout); // false

filter_1.knockout = true;

trace(filter_1.knockout); // true
trace(filter_2.knockout); // true
trace(clonedFilter.knockout); // false
```

# color (propriété GlowFilter.color)

```
public color : nombre
```

La couleur du rayonnement. Les valeurs valides sont au format hexadécimal 0x*RRVVBB*. La valeur par défaut est 0xFF0000.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

## Exemple

L'exemple suivant modifie la propriété `color` sur un clip existant lorsqu'un utilisateur clique dessus.

```
import flash.filters.GlowFilter;

var mc:MovieClip = createGlowFilterRectangle("GlowFilterColor");
mc.onRelease = function() {
  var filter:GlowFilter = this.filters[0];
  filter.color = 0x00FF33;
  this.filters = new Array(filter);
}

function createGlowFilterRectangle(name:String):MovieClip {
  var rect:MovieClip = this.createEmptyMovieClip(name,
  this.getNextHighestDepth());
  var w:Number = 100;
  var h:Number = 100;
  rect.beginFill(0x003366);
  rect.lineTo(w, 0);
  rect.lineTo(w, h);
  rect.lineTo(0, h);
  rect.lineTo(0, 0);
  rect._x = 20;
  rect._y = 20;

  var filter:GlowFilter = new GlowFilter(0x000000, .8, 16, 16, 1, 3, false,
  false);
  var filterArray:Array = new Array();
  filterArray.push(filter);
  rect.filters = filterArray;
  return rect;
}
```

# constructeur GlowFilter

```
public GlowFilter([color:Number], [alpha:Number], [blurX:Number],
  [blurY:Number], [strength:Number], [quality:Number], [inner:Boolean],
  [knockout:Boolean])
```

Initialise une nouvelle instance GlowFilter avec les paramètres spécifiés.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

## Paramètres

*color*:Number [facultatif] - Couleur du néon, au format hexadécimal 0x*RRVVBB*. La valeur par défaut est 0xFF0000.

*alpha*:Number [facultatif] - Valeur de transparence alpha de la couleur. Les valeurs valides sont comprises entre 0 et 1. Par exemple, 0,25 définit une valeur de transparence de 25 %. La valeur par défaut est 1.

*blurX*:Number [facultatif] - Niveau de flou horizontal. Les valeurs valides sont comprises entre 0 et 255 (virgule flottante). La valeur par défaut est 6. Les valeurs correspondant à une puissance de 2 (2, 4, 8, 16 et 32) sont optimisées pour obtenir un rendu plus rapide qu'avec les autres valeurs.

*blurY*:Number [facultatif] - Niveau de flou vertical. Les valeurs valides sont comprises entre 0 et 255 (virgule flottante). La valeur par défaut est 6. Les valeurs correspondant à une puissance de 2 (2, 4, 8, 16 et 32) sont optimisées pour obtenir un rendu plus rapide qu'avec les autres valeurs.

*strength*:Number [facultatif] - Intensité de l'impression ou du recouvrement. Plus la valeur est élevée, plus l'intensité des couleurs apparaît à l'impression et plus le contraste est important entre le rayonnement et l'arrière-plan. Les valeurs valides sont comprises entre 0 et 255. La valeur par défaut est 2.

*quality*:Number [facultatif] - Nombre d'applications du filtre. Les valeurs valides sont comprises entre 0 et 15. La valeur par défaut, 1, correspond à une qualité faible. La valeur 2 offre une qualité moyenne et la valeur 3, une qualité élevée.

*inner*:Boolean [facultatif] - Spécifie si le rayonnement est interne. La valeur `true` indique un rayonnement interne. La valeur par défaut est `false`, renvoyant un rayonnement externe (au niveau des bords extérieurs de l'objet).

*knockout*:Boolean [facultatif] - Spécifie si l'objet a un effet de poinçonnage. La valeur `true` applique un effet de poinçonnage qui rend le remplissage de l'objet transparent et révèle la couleur d'arrière-plan du document. La valeur par défaut est `false` (pas d'effet de poinçonnage).

## Exemple

L'exemple suivant crée une nouvelle instance GlowFilter et l'applique à un rectangle plat.

```
import flash.filters.GlowFilter;

var rect:MovieClip = createRectangle(100, 100, 0x003366,
  "gradientGlowFilterExample");
```

```
var color:Number = 0x33CCFF;
var alpha:Number = .8;
var blurX:Number = 35;
var blurY:Number = 35;
var strength:Number = 2;
var quality:Number = 3;
var inner:Boolean = false;
var knockout:Boolean = false;

var filter:GlowFilter = new GlowFilter(color,
                        alpha,
                        blurX,
                        blurY,
                        strength,
                        quality,
                        inner,
                        knockout);
var filterArray:Array = new Array();
filterArray.push(filter);
rect.filters = filterArray;

function createRectangle(w:Number, h:Number, bgColor:Number,
  name:String):MovieClip {
  var mc:MovieClip = this.createEmptyMovieClip(name,
  this.getNextHighestDepth());
  mc.beginFill(bgColor);
  mc.lineTo(w, 0);
  mc.lineTo(w, h);
  mc.lineTo(0, h);
  mc.lineTo(0, 0);
  mc._x = 20;
  mc._y = 20;
  return mc;
}
```

## inner (propriété GlowFilter.inner)

```
public inner : Boolean
```

Spécifie si le rayonnement est interne. La valeur `true` indique un rayonnement interne. La valeur par défaut est `false`, renvoyant un rayonnement externe (au niveau des bords extérieurs de l'objet).

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

### Exemple

L'exemple suivant modifie la propriété `inner` sur un clip existant lorsqu'un utilisateur clique dessus.

```
import flash.filters.GlowFilter;

var mc:MovieClip = createGlowFilterRectangle("GlowFilterInner");
mc.onRelease = function() {
  var filter:GlowFilter = this.filters[0];
  filter.inner = true;
  this.filters = new Array(filter);
}

function createGlowFilterRectangle(name:String):MovieClip {
  var rect:MovieClip = this.createEmptyMovieClip(name,
  this.getNextHighestDepth());
  var w:Number = 100;
  var h:Number = 100;
  rect.beginFill(0x003366);
  rect.lineTo(w, 0);
  rect.lineTo(w, h);
  rect.lineTo(0, h);
  rect.lineTo(0, 0);
  rect._x = 20;
  rect._y = 20;

  var filter:GlowFilter = new GlowFilter(0x000000, .8, 16, 16, 1, 3, false,
  false);
  var filterArray:Array = new Array();
  filterArray.push(filter);
  rect.filters = filterArray;
  return rect;
}
```

## knockout (propriété GlowFilter.knockout)

```
public knockout : Boolean
```

Spécifie si l'objet a un effet de poinçonnage. La valeur `true` applique un effet de poinçonnage qui rend le remplissage de l'objet transparent et révèle la couleur d'arrière-plan du document. La valeur par défaut est `false` (pas d'effet de poinçonnage).

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

### Exemple

L'exemple suivant modifie la propriété `knockout` sur un clip existant lorsqu'un utilisateur clique dessus.

```
import flash.filters.GlowFilter;
```

```
var mc:MovieClip = createGlowFilterRectangle("GlowFilterKnockout");
mc.onRelease = function() {
  var filter:GlowFilter = this.filters[0];
  filter.knockout = true;
  this.filters = new Array(filter);
}

function createGlowFilterRectangle(name:String):MovieClip {
  var rect:MovieClip = this.createEmptyMovieClip(name,
  this.getNextHighestDepth());
  var w:Number = 100;
  var h:Number = 100;
  rect.beginFill(0x003366);
  rect.lineTo(w, 0);
  rect.lineTo(w, h);
  rect.lineTo(0, h);
  rect.lineTo(0, 0);
  rect._x = 20;
  rect._y = 20;

  var filter:GlowFilter = new GlowFilter(0x000000, .8, 16, 16, 1, 3, false,
  false);
  var filterArray:Array = new Array();
  filterArray.push(filter);
  rect.filters = filterArray;
  return rect;
}
```

## quality (propriété GlowFilter.quality)

```
public quality : nombre
```

Le nombre d'applications du filtre. Les valeurs valides sont comprises entre 0 et 15. La valeur par défaut, 1, correspond à une qualité faible. La valeur 2 offre une qualité moyenne et la valeur 3, une qualité élevée. Les rendus des filtres de valeurs inférieures sont obtenus plus rapidement.

Pour la plupart des applications, une valeur de `quality` de 1, 2 ou 3 est suffisante. Il est possible d'utiliser des valeurs numériques jusqu'à 15 pour obtenir différents effets, toutefois le rendu des valeurs les plus élevées est moins rapide. Sans augmenter la valeur de `quality`, vous pouvez généralement obtenir un effet similaire, avec un rendu plus rapide, en augmentant simplement les valeurs de `blurX` et `blurY`.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

### Exemple

L'exemple suivant modifie la propriété `quality` sur un clip existant lorsqu'un utilisateur clique dessus.

```
import flash.filters.GlowFilter;

var mc:MovieClip = createGlowFilterRectangle("GlowFilterQuality");
mc.onRelease = function() {
  var filter:GlowFilter = this.filters[0];
  filter.quality = 1;
  this.filters = new Array(filter);
}

function createGlowFilterRectangle(name:String):MovieClip {
  var rect:MovieClip = this.createEmptyMovieClip(name,
  this.getNextHighestDepth());
  var w:Number = 100;
  var h:Number = 100;
  rect.beginFill(0x003366);
  rect.lineTo(w, 0);
  rect.lineTo(w, h);
  rect.lineTo(0, h);
  rect.lineTo(0, 0);
  rect._x = 20;
  rect._y = 20;

  var filter:GlowFilter = new GlowFilter(0x000000, .8, 16, 16, 1, 3, false,
  false);
  var filterArray:Array = new Array();
  filterArray.push(filter);
  rect.filters = filterArray;
  return rect;
}
```

## strength (propriété GlowFilter.strength)

```
public strength : nombre
```

L'intensité de l'impression ou du recouvrement. Plus la valeur est élevée, plus l'intensité des couleurs apparaît à l'impression et plus le contraste est important entre le rayonnement et l'arrière-plan. Les valeurs valides sont comprises entre 0 et 255. La valeur par défaut est 2.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

### Exemple

L'exemple suivant modifie la propriété `strength` sur un clip existant lorsqu'un utilisateur clique dessus.

```
import flash.filters.GlowFilter;
```

```
var mc:MovieClip = createGlowFilterRectangle("GlowFilterStrength");
mc.onRelease = function() {
  var filter:GlowFilter = this.filters[0];
  filter.strength = .8;
  this.filters = new Array(filter);
}

function createGlowFilterRectangle(name:String):MovieClip {
  var rect:MovieClip = this.createEmptyMovieClip(name,
  this.getNextHighestDepth());
  var w:Number = 100;
  var h:Number = 100;
  rect.beginFill(0x003366);
  rect.lineTo(w, 0);
  rect.lineTo(w, h);
  rect.lineTo(0, h);
  rect.lineTo(0, 0);
  rect._x = 20;
  rect._y = 20;

  var filter:GlowFilter = new GlowFilter(0x000000, .8, 16, 16, 1, 3, false,
  false);
  var filterArray:Array = new Array();
  filterArray.push(filter);
  rect.filters = filterArray;
  return rect;
}
```

# GradientBevelFilter (flash.filters.GradientBevelFilter)

```
Object
  |
  +-flash.filters.BitmapFilter
    |
    +-flash.filters.GradientBevelFilter
```

```
public class GradientBevelFilter
extends BitmapFilter
```

La classe GradientBevelFilter permet d'appliquer un effet de biseau en dégradés à divers objets dans Flash. Un biseau en dégradés est une bordure en relief améliorée par des couleurs dégradées à l'extérieur, à l'intérieur ou sur la partie supérieure d'un objet. Les bordures en relief donnent un aspect tridimensionnel aux objets.

L'utilisation de filtres dépend de l'objet auquel vous appliquez le filtre.

- Pour appliquer des filtres aux clips, champs de texte et boutons lors de l'exécution, utilisez la propriété `filters`. Lorsque vous définissez la propriété `filters` d'un objet, celui-ci n'est pas modifié. En outre, vous pouvez l'annuler en supprimant la propriété `filters`.
- Pour appliquer des filtres aux occurrences BitmapData, utilisez la méthode `BitmapData.applyFilter()`. L'appel de `applyFilter()` sur un objet BitmapData génère une image filtrée à partir de l'objet BitmapData source et de l'objet Filter.

Vous pouvez également appliquer des effets de filtre aux images et aux données vidéo durant la programmation. Pour plus d'informations, consultez la documentation relative à la programmation.

Si vous appliquez un filtre à un clip ou à un bouton, la propriété `cacheAsBitmap` du clip ou du bouton est définie sur `true`. Si vous supprimez tous les filtres, la valeur d'origine de `cacheAsBitmap` est restaurée.

Ce filtre prend en charge le redimensionnement de la scène. Toutefois, le redimensionnement général, la rotation et l'inclinaison ne sont pas pris en charge ; si l'objet est redimensionné (si le redimensionnement `_xscale` et le redimensionnement `_yscale` ne sont pas définis sur 100 %), l'effet de filtre n'est pas redimensionné. Le redimensionnement est effectué uniquement en cas de zoom sur la scène.

Un filtre ne sera pas appliqué si l'image obtenue dépasse 2 880 pixels en largeur ou en hauteur. Par exemple, lorsque vous effectuez un zoom avant sur un clip de grande taille auquel un filtre est appliqué, le filtre sera désactivé si l'image obtenue dépasse la limite de 2 880 pixels.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

## Voir également

`ratios (propriété GradientBevelFilter.ratios)`, `applyFilter (méthode BitmapData.applyFilter)`, `BevelFilter (flash.filters.BevelFilter)`, `filters (propriété Button.filters)`, `cacheAsBitmap (propriété Button.cacheAsBitmap)`, `cacheAsBitmap (propriété MovieClip.cacheAsBitmap)`, `filters (propriété MovieClip.filters)`, `filters (propriété TextField.filters)`

## Résumé des propriétés

| Modificateurs | Propriété | Description |
|---|---|---|
| | `alphas:Array` | Un tableau de valeurs de transparence alpha pour les couleurs correspondantes dans le tableau `colors`. |
| | `angle:Number` | Angle exprimé en degrés. |
| | `blurX:Number` | Le niveau de flou horizontal. |
| | `blurY:Number` | Le niveau de flou vertical. |
| | `colors:Array` | Un tableau de valeurs hexadécimales de couleur RVB à utiliser pour le dégradé. |
| | `distance:Number` | La distance de décalage. |
| | `knockout:Boolean` | Spécifie si l'objet a un effet de poinçonnage. |
| | `quality:Number` | Le nombre d'applications du filtre. |
| | `ratios:Array` | Un tableau de taux de distribution des couleurs, pour les couleurs correspondantes dans le tableau `colors`. |
| | `strength:Number` | L'intensité de l'impression ou du recouvrement. |
| | `type:String` | Le positionnement de l'effet de biseau. |

*Propriétés héritées de la classe Object*

`constructeur (propriété Object.constructor), __proto__ (Object.__proto__ property), prototype (Object.prototype, propriété), __resolve (Object.__resolve, propriété)`

## Résumé des constructeurs

| Signature | Description |
|---|---|
| `GradientBevelFilter( [distance:Number], [angle:Number], [colors:Array], [alphas:Array], [ratios:Array], [blurX:Number], [blurY:Number], [strength:Number], [quality:Number], [type:String], [knockout:Boolean])` | Initialise le filtre avec les paramètres spécifiés. |

### Résumé de la méthode

| Modificateurs | Signature | Description |
|---|---|---|
| | `clone() : GradientBevelFilter` | Renvoie une copie de cet objet filtre. |

*Méthodes héritées de la classe BitmapFilter*

`clone (méthode BitmapFilter.clone )`

*Méthodes héritées de la classe Object*

`addProperty (méthode Object.addProperty), hasOwnProperty (méthode Object.hasOwnProperty), isPropertyEnumerable (méthode Object.isPropertyEnumerable), isPrototypeOf (méthode Object.isPrototypeOf), registerClass (méthode Object.registerClass), toString (méthode Object.toString), unwatch (méthode Object.unwatch), valueOf (méthode Object.valueOf), watch (méthode Object.watch)`

## alphas (propriété GradientBevelFilter.alphas)

`public alphas : Array`

Un tableau de valeurs de transparence alpha pour les couleurs correspondantes dans le tableau `colors`. Les valeurs valides pour chaque élément du tableau sont comprises entre 0 et 1. Par exemple, 0,25 définit une valeur de transparence de 25 %.

La propriété `alphas` ne peut pas être modifiée en manipulant directement ses valeurs. Vous devez obtenir une référence à `alphas`, effectuer les modifications sur la référence, puis définir la propriété `alphas` sur la référence.

Les propriétés `colors`, `alphas` et `ratios` sont toutes liées. Le premier élément du tableau `colors` correspond au premier élément du tableau `alphas`, du tableau `ratios`, etc.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

### Exemple

L'exemple suivant montre comment définir la propriété `alphas` sur une entité existante.

```
import flash.filters.GradientBevelFilter;

var mc:MovieClip = setUpFilter("alphasExample");
mc.onPress = function() {
  var arr:Array = this.filters;
  var alphas:Array = [.2, 0, .2];
  arr[0].alphas = alphas;
```

```
  this.filters = arr;
}
mc.onRelease = function() {
  var arr:Array = this.filters;
  var alphas:Array = [1, 0, 1];
  arr[0].alphas = alphas;
  this.filters = arr;
}

function setUpFilter(name:String):MovieClip {
  var art:MovieClip = this.createEmptyMovieClip(name,
  this.getNextHighestDepth());
  var w:Number = 150;
  var h:Number = 150;
  art.beginFill(0xCCCCCC);
  art.lineTo(w, 0);
  art.lineTo(w, h);
  art.lineTo(0, h);
  art.lineTo(0, 0);

  var colors:Array = [0xFFFFFF, 0xCCCCCC, 0x000000];
  var alphas:Array = [1, 0, 1];
  var ratios:Array = [0, 128, 255];
  var filter:GradientBevelFilter = new GradientBevelFilter(5, 225, colors,
  alphas, ratios, 5, 5, 5, 2, "inner", false);

  art.filters = new Array(filter);
  return art;
}
```

### Voir également

colors (propriété GradientBevelFilter.colors), ratios (propriété GradientBevelFilter.ratios)

## angle (propriété GradientBevelFilter.angle)

```
public angle : nombre
```

Angle exprimé en degrés. Les valeurs valides sont comprises entre 0 et 360. La valeur par défaut est 45.

La valeur d'angle représente l'angle de la source lumineuse théorique éclairant l'objet. Elle détermine l'angle d'application des couleurs dégradées à l'objet : aux emplacements éclairés et ombrés ou à l'emplacement où la première couleur du tableau apparaît. Les couleurs s'appliquent alors selon leur ordre d'apparition dans le tableau.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

### Exemple

L'exemple suivant montre comment définir la propriété `angle` sur un objet existant.

```
import flash.filters.GradientBevelFilter;

var mc:MovieClip = setUpFilter("angleExample");
mc.onRelease = function() {
  var arr:Array = this.filters;
  arr[0].angle = 45;
  this.filters = arr;
}

function setUpFilter(name:String):MovieClip {
  var art:MovieClip = this.createEmptyMovieClip(name,
  this.getNextHighestDepth());
  var w:Number = 150;
  var h:Number = 150;
  art.beginFill(0xCCCCCC);
  art.lineTo(w, 0);
  art.lineTo(w, h);
  art.lineTo(0, h);
  art.lineTo(0, 0);

  var colors:Array = [0xFFFFFF, 0xCCCCCC, 0x000000];
  var alphas:Array = [1, 0, 1];
  var ratios:Array = [0, 128, 255];
  var filter:GradientBevelFilter = new GradientBevelFilter(5, 225, colors,
  alphas, ratios, 5, 5, 5, 3, "inner", false);

  art.filters = new Array(filter);
  return art;
}
```

### Voir également

`ratios (propriété GradientBevelFilter.ratios)`

## blurX (propriété GradientBevelFilter.blurX)

`public blurX : nombre`

Le niveau de flou horizontal. Les valeurs valides sont comprises entre 0 et 255. Un flou d'une valeur inférieure ou égale à 1 signifie que l'image d'origine n'est pas modifiée avant d'être copiée. La valeur par défaut est 4. Les valeurs correspondant à une puissance de 2 (2, 4, 8, 16 et 32) sont optimisées pour obtenir un rendu plus rapide qu'avec les autres valeurs.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

### Exemple

L'exemple suivant montre comment définir la propriété `blurX` sur un objet existant.

```
import flash.filters.GradientBevelFilter;

var mc:MovieClip = setUpFilter("blurXExample");
mc.onRelease = function() {
  var arr:Array = this.filters;
  arr[0].blurX = 16;
  this.filters = arr;
}

function setUpFilter(name:String):MovieClip {
  var art:MovieClip = this.createEmptyMovieClip(name,
  this.getNextHighestDepth());
  var w:Number = 150;
  var h:Number = 150;
  art.beginFill(0xCCCCCC);
  art.lineTo(w, 0);
  art.lineTo(w, h);
  art.lineTo(0, h);
  art.lineTo(0, 0);

  var colors:Array = [0xFFFFFF, 0xCCCCCC, 0x000000];
  var alphas:Array = [1, 0, 1];
  var ratios:Array = [0, 128, 255];
  var filter:GradientBevelFilter = new GradientBevelFilter(5, 225, colors,
  alphas, ratios, 5, 5, 5, 3, "inner", false);

  art.filters = new Array(filter);
  return art;
}
```

## blurY (propriété GradientBevelFilter.blurY)

```
public blurY : nombre
```

Le niveau de flou vertical. Les valeurs valides sont comprises entre 0 et 255. Un flou d'une valeur inférieure ou égale à 1 signifie que l'image d'origine n'est pas modifiée avant d'être copiée. La valeur par défaut est 4. Les valeurs correspondant à une puissance de 2 (2, 4, 8, 16 et 32) sont optimisées pour obtenir un rendu plus rapide qu'avec les autres valeurs.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

### Exemple

L'exemple suivant montre comment définir la propriété `blurY` sur un objet existant.

```
import flash.filters.GradientBevelFilter;
```

```
var mc:MovieClip = setUpFilter("blurYExample");
mc.onRelease = function() {
  var arr:Array = this.filters;
  arr[0].blurY = 16;
  this.filters = arr;
}

function setUpFilter(name:String):MovieClip {
  var art:MovieClip = this.createEmptyMovieClip(name,
  this.getNextHighestDepth());
  var w:Number = 150;
  var h:Number = 150;
  art.beginFill(0xCCCCCC);
  art.lineTo(w, 0);
  art.lineTo(w, h);
  art.lineTo(0, h);
  art.lineTo(0, 0);

  var colors:Array = [0xFFFFFF, 0xCCCCCC, 0x000000];
  var alphas:Array = [1, 0, 1];
  var ratios:Array = [0, 128, 255];
  var filter:GradientBevelFilter = new GradientBevelFilter(5, 225, colors,
  alphas, ratios, 5, 5, 5, 3, "inner", false);

  art.filters = new Array(filter);
  return art;
}
```

## clone (méthode GradientBevelFilter.clone)

```
public clone() : GradientBevelFilter
```

Renvoie une copie de cet objet filtre.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

### Valeur renvoyée

`flash.filters.GradientBevelFilter` - Nouvelle instance GradientBevelFilter dont toutes les propriétés sont identiques à celles de l'instance GradientBevelFilter d'origine.

### Exemple

L'exemple suivant crée deux formes rectangulaires. Un effet de biseau est appliqué à la première, `sourceClip`. Aucun effet n'est appliqué à la deuxième, `resultClip`, tant que vous ne cliquez pas dessus.

```
import flash.filters.GradientBevelFilter;
```

```
var sourceClip:MovieClip = setUpFlatRectangle(150, 150, 0xCCCCCC,
  "cloneSourceClip");
var resultClip:MovieClip = setUpFlatRectangle(150, 150, 0xCCCCCC,
  "cloneResultClip");

resultClip.source = sourceClip;

var sourceFilter:GradientBevelFilter = getNewFilter();
sourceClip.filters = new Array(sourceFilter);

resultClip._x = 180;
resultClip.onRelease = function() {
  this.filters = new Array(this.source.filters[0].clone());
}

function setUpFlatRectangle(w:Number, h:Number, bgColor:Number,
  name:String):MovieClip {
  var mc:MovieClip = this.createEmptyMovieClip(name,
  this.getNextHighestDepth());
  mc.beginFill(bgColor);
  mc.lineTo(w, 0);
  mc.lineTo(w, h);
  mc.lineTo(0, h);
  mc.lineTo(0, 0);
  return mc;
}

function getNewFilter():GradientBevelFilter {
  var colors:Array = [0xFFFFFF, 0xCCCCCC, 0x000000];
  var alphas:Array = [1, 0, 1];
  var ratios:Array = [0, 128, 255];
  return new GradientBevelFilter(5, 225, colors, alphas, ratios, 5, 5, 5,
  2, "inner", false);
}
```

## colors (propriété GradientBevelFilter.colors)

```
public colors : Array
```

Un tableau de valeurs hexadécimales de couleur RVB à utiliser pour le dégradé. Par exemple, rouge correspond à 0xFF0000, bleu à 0x0000FF, etc.

La propriété `colors` ne peut pas être modifiée en manipulant directement ses valeurs. Vous devez obtenir une référence à `colors`, effectuer les modifications sur la référence, puis définir la propriété `colors` sur la référence.

Les propriétés `colors`, `alphas` et `ratios` sont toutes liées. Le premier élément du tableau `colors` correspond au premier élément du tableau `alphas`, du tableau `ratios`, etc.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

## Exemple

L'exemple suivant montre comment définir la propriété `colors` sur une entité existante.

```
import flash.filters.GradientBevelFilter;

var mc:MovieClip = setUpFilter("colorsExample");
mc.onPress = function() {
  var arr:Array = this.filters;
  var colors:Array = [0x000000, 0xCCCCCC, 0xFFFFFF];
  arr[0].colors = colors;
  this.filters = arr;
}
mc.onRelease = function() {
  var arr:Array = this.filters;
  var colors:Array = [0xFFFFFF, 0xCCCCCC, 0x000000];
  arr[0].colors = colors;
  this.filters = arr;
}

function setUpFilter(name:String):MovieClip {
  var art:MovieClip = this.createEmptyMovieClip(name,
  this.getNextHighestDepth());
  var w:Number = 150;
  var h:Number = 150;
  art.beginFill(0xCCCCCC);
  art.lineTo(w, 0);
  art.lineTo(w, h);
  art.lineTo(0, h);
  art.lineTo(0, 0);

  var colors:Array = [0xFFFFFF, 0xCCCCCC, 0x000000];
  var alphas:Array = [1, 0, 1];
  var ratios:Array = [0, 128, 255];
  var filter:GradientBevelFilter = new GradientBevelFilter(5, 225, colors,
  alphas, ratios, 5, 5, 5, 2, "inner", false);

  art.filters = new Array(filter);
  return art;
}
```

## Voir également

`alphas (propriété GradientBevelFilter.alphas)`, `ratios (propriété GradientBevelFilter.ratios)`

# distance (propriété GradientBevelFilter.distance)

```
public distance : nombre
```

La distance de décalage. La valeur par défaut est 4.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

## Exemple

L'exemple suivant montre comment définir la propriété `distance` sur un objet existant.

```
import flash.filters.GradientBevelFilter;

var mc:MovieClip = setUpFilter("distanceExample");
mc.onRelease = function() {
  var arr:Array = this.filters;
  arr[0].distance = 1;
  this.filters = arr;
}

function setUpFilter(name:String):MovieClip {
  var art:MovieClip = this.createEmptyMovieClip(name,
  this.getNextHighestDepth());
  var w:Number = 150;
  var h:Number = 150;
  art.beginFill(0xCCCCCC);
  art.lineTo(w, 0);
  art.lineTo(w, h);
  art.lineTo(0, h);
  art.lineTo(0, 0);

  var colors:Array = [0xFFFFFF, 0xCCCCCC, 0x000000];
  var alphas:Array = [1, 0, 1];
  var ratios:Array = [0, 128, 255];
  var filter:GradientBevelFilter = new GradientBevelFilter(5, 225, colors,
  alphas, ratios, 5, 5, 5, 3, "inner", false);

  art.filters = new Array(filter);
  return art;
}
```

# Constructeur GradientBevelFilter

```
public GradientBevelFilter([distance:Number], [angle:Number],
  [colors:Array], [alphas:Array], [ratios:Array], [blurX:Number],
  [blurY:Number], [strength:Number], [quality:Number], [type:String],
  [knockout:Boolean])
```

Initialise le filtre avec les paramètres spécifiés.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

## Paramètres

*distance*:Number [facultatif] - Distance de décalage. Les valeurs valides sont comprises entre 0 et 8. La valeur par défaut est 4.

*angle*:Number [facultatif] - Angle exprimé en degrés. Les valeurs valides sont comprises entre 0 et 360. La valeur par défaut est 45.

*colors*:Array [facultatif] - Tableau de valeurs hexadécimales de couleur RVB à utiliser pour le dégradé. Par exemple, rouge correspond à 0xFF0000, bleu à 0x0000FF, etc.

*alphas*:Array [facultatif] - Tableau de valeurs de transparence alpha pour les couleurs correspondantes dans le tableau colors. Les valeurs valides pour chaque élément du tableau sont comprises entre 0 et 1. Par exemple, 0,25 définit une valeur de transparence de 25 %.

*ratios*:Array [facultatif] - Tableau de taux de distribution des couleurs ; les valeurs valides sont comprises entre 0 et 255.

*blurX*:Number [facultatif] - Niveau de flou horizontal. Les valeurs valides sont comprises entre 0 et 255. Un flou d'une valeur inférieure ou égale à 1 signifie que l'image d'origine n'est pas modifiée avant d'être copiée. La valeur par défaut est 4. Les valeurs correspondant à une puissance de 2 (2, 4, 8, 16 et 32) sont optimisées pour obtenir un rendu plus rapide qu'avec les autres valeurs.

*blurY*:Number [facultatif] - Niveau de flou vertical. Les valeurs valides sont comprises entre 0 et 255. Un flou d'une valeur inférieure ou égale à 1 signifie que l'image d'origine n'est pas modifiée avant d'être copiée. La valeur par défaut est 4. Les valeurs correspondant à une puissance de 2 (2, 4, 8, 16 et 32) sont optimisées pour obtenir un rendu plus rapide qu'avec les autres valeurs.

*strength*:Number [facultatif] - Intensité de l'impression ou du recouvrement. Plus la valeur est élevée, plus l'intensité des couleurs apparaît à l'impression et plus le contraste est important entre le biseau et l'arrière-plan. Les valeurs valides sont comprises entre 0 et 255. La valeur 0 signifie que le filtre n'est pas appliqué. La valeur par défaut est 1.

*quality*:Number [facultatif] - Qualité du filtre. Les valeurs valides sont comprises entre 0 et 15. La valeur par défaut est 1. Dans la plupart des cas, les valeurs utiles sont 1 (qualité basse), 2 (qualité moyenne) et 3 (qualité élevée). Les rendus des filtres de valeurs inférieures sont obtenus plus rapidement.

*type*:String [facultatif] - Positionnement de l'effet de biseau. Les valeurs possibles sont :

- "outer" : Biseau sur le bord extérieur de l'objet
- "inner" : Biseau sur le bord intérieur de l'objet
- "full" : Biseau sur le bord supérieur de l'objet

La valeur par défaut est "inner".

*knockout*:Boolean [facultatif] - Spécifie si un effet de poinçonnage est appliqué. La valeur `true` applique un effet de poinçonnage qui rend le remplissage de l'objet transparent et révèle la couleur d'arrière-plan du document. La valeur par défaut est `false` (pas de poinçonnage).

## Exemple

L'exemple suivant crée une nouvelle instance GradientBevelFilter, affecte ses valeurs et l'applique à un rectangle plat.

```
import flash.filters.GradientBevelFilter;
import flash.filters.BitmapFilter;
var art:MovieClip = setUpFlatRectangle(150, 150, 0xCCCCCC,
  "gradientBevelFilterExample");
var distance:Number = 5;
var angleInDegrees:Number = 225; // opposite 45 degrees
var colors:Array = [0xFFFFFF, 0xCCCCCC, 0x000000];
var alphas:Array = [1, 0, 1];
var ratios:Array = [0, 128, 255];
var blurX:Number = 8;
var blurY:Number = 8;
var strength:Number = 2;
var quality:Number = 3;
var type:String = "inner";
var knockout:Boolean = true;

var filter:GradientBevelFilter = new GradientBevelFilter(distance,
                                 angleInDegrees,
                                 colors,
                                 alphas,
                                 ratios,
                                 blurX,
                                 blurY,
                                 strength,
                                 quality,
                                 type,
                                 knockout);
var filterArray:Array = new Array();
filterArray.push(filter);
art.filters = filterArray;

function setUpFlatRectangle(w:Number, h:Number, bgColor:Number,
  name:String):MovieClip {
  var mc:MovieClip = this.createEmptyMovieClip(name,
  this.getNextHighestDepth());
  mc.beginFill(bgColor);
  mc.lineTo(w, 0);
  mc.lineTo(w, h);
  mc.lineTo(0, h);
  mc.lineTo(0, 0);
  return mc;
```

```
}
```

### Voir également

ratios (propriété GradientBevelFilter.ratios)

## knockout (propriété GradientBevelFilter.knockout)

```
public knockout : Boolean
```

Spécifie si l'objet a un effet de poinçonnage. Un effet de poinçonnage rend le remplissage de l'objet transparent et révèle la couleur d'arrière-plan du document. La valeur `true` spécifie un effet de poinçonnage ; la valeur par défaut est `false` (pas d'effet de poinçonnage).

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

### Exemple

L'exemple suivant montre comment définir la propriété `knockout` sur un objet existant.

```
import flash.filters.GradientBevelFilter;

var mc:MovieClip = setUpFilter("knockoutExample");
mc.onRelease = function() {
  var arr:Array = this.filters;
  arr[0].knockout = true;
  this.filters = arr;
}

function setUpFilter(name:String):MovieClip {
  var art:MovieClip = this.createEmptyMovieClip(name,
  this.getNextHighestDepth());
  var w:Number = 150;
  var h:Number = 150;
  art.beginFill(0xCCCCCC);
  art.lineTo(w, 0);
  art.lineTo(w, h);
  art.lineTo(0, h);
  art.lineTo(0, 0);

  var colors:Array = [0xFFFFFF, 0xCCCCCC, 0x000000];
  var alphas:Array = [1, 0, 1];
  var ratios:Array = [0, 128, 255];
  var filter:GradientBevelFilter = new GradientBevelFilter(5, 225, colors,
  alphas, ratios, 5, 5, 5, 3, "inner", false);

  art.filters = new Array(filter);
  return art;
}
```

# quality (propriété GradientBevelFilter.quality)

```
public quality : nombre
```

Le nombre d'applications du filtre. Les valeurs valides sont comprises entre 0 et 15. La valeur par défaut, 1, correspond à une qualité faible. La valeur 2 offre une qualité moyenne et la valeur 3, une qualité élevée. Les rendus des filtres de valeurs inférieures sont obtenus plus rapidement.

Pour la plupart des applications, une valeur de `quality` de 1, 2 ou 3 est suffisante. Il est possible d'utiliser des valeurs numériques jusqu'à 15 pour obtenir différents effets, toutefois le rendu des valeurs les plus élevées est moins rapide. Sans augmenter la valeur de `quality`, vous pouvez généralement obtenir un effet similaire, avec un rendu plus rapide, en augmentant simplement les valeurs de `blurX` et `blurY`.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

## Exemple

L'exemple suivant montre comment définir la propriété `quality` sur un objet existant.

```
import flash.filters.GradientBevelFilter;

var mc:MovieClip = setUpFilter("qualityExample");
mc.onRelease = function() {
  var arr:Array = this.filters;
  arr[0].quality = 1; // low quality
  this.filters = arr;
}

function setUpFilter(name:String):MovieClip {
  var art:MovieClip = this.createEmptyMovieClip(name,
  this.getNextHighestDepth());
  var w:Number = 150;
  var h:Number = 150;
  art.beginFill(0xCCCCCC);
  art.lineTo(w, 0);
  art.lineTo(w, h);
  art.lineTo(0, h);
  art.lineTo(0, 0);

  var colors:Array = [0xFFFFFF, 0xCCCCCC, 0x000000];
  var alphas:Array = [1, 0, 1];
  var ratios:Array = [0, 128, 255];
  var filter:GradientBevelFilter = new GradientBevelFilter(5, 225, colors,
  alphas, ratios, 5, 5, 5, 3, "inner", false);

  art.filters = new Array(filter);
  return art;
}
```

### Voir également

ratios (propriété GradientBevelFilter.ratios)

## ratios (propriété GradientBevelFilter.ratios)

```
public ratios : Array
```

Un tableau de taux de distribution des couleurs, pour les couleurs correspondantes dans le tableau `colors`. Les valeurs valides pour chaque élément du tableau sont comprises entre 0 et 255.

La propriété `ratios` ne peut pas être modifiée en manipulant directement ses valeurs. Vous devez obtenir une référence à `ratios`, effectuer les modifications sur la référence, puis définir la propriété `ratios` sur la référence.

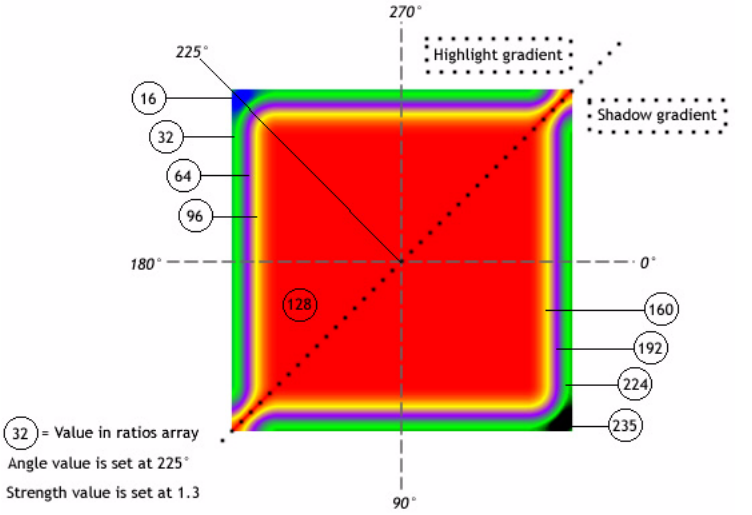
Les propriétés `colors`, `alphas` et `ratios` sont toutes liées. Le premier élément du tableau `colors` correspond au premier élément du tableau `alphas`, du tableau `ratios`, etc.

Pour comprendre comment les couleurs du biseau dégradé sont appliquées, considérez les couleurs que vous souhaitez intégrer à votre biseau dégradé. Un biseau simple possède une couleur d'éclairage et une couleur d'ombre. Un biseau dégradé possède un dégradé éclairé et un dégradé ombré. Supposons que l'éclairage apparaît dans l'angle supérieur gauche et l'ombre, dans l'angle inférieur droit. Et imaginons que l'une des utilisations possibles du filtre comporte quatre couleurs dans la zone éclairée et quatre dans l'ombre. En plus de l'éclairage et de l'ombre, le filtre utilise une couleur de remplissage de base qui apparaît à la jonction des zones éclairées et ombrées. Le nombre total de couleurs est donc neuf, et le nombre d'éléments correspondants dans le tableau des rapports, également neuf.

Si vous souhaitez obtenir un dégradé composé de bandes de différentes couleurs qui se mêlent les unes aux autres, chaque valeur de ratio définit l'emplacement de la couleur sur le rayon du dégradé, 0 représentant le point le plus éloigné et 255, le point le plus proche du centre du dégradé. En général, la valeur moyenne, 128, constitue la valeur de remplissage de base. Pour obtenir l'effet de biseau illustré ci-dessous, affectez les valeurs de ratio comme suit, en vous aidant des neuf couleurs d'exemple :

- Les quatre premières couleurs se situent dans la plage 0-127, chaque valeur étant supérieure ou égale à la précédente. Elles définissent le bord du biseau éclairé.
- La cinquième couleur (la couleur du milieu) correspond au remplissage de base, soit 128. La valeur de pixel 128 définit le remplissage de base qui apparaît à l'extérieur de la forme (et au niveau des bords du biseau) si le type externe est spécifié ; ou à l'intérieur de la forme en couvrant effectivement le remplissage de l'objet, si le type spécifié est interne.
- Les quatre dernières couleurs se situent dans la plage 129-255, chaque valeur étant supérieure ou égale à la précédente. Elles définissent le bord du biseau ombré.

Pour obtenir une distribution équivalente des couleurs sur chaque bord, utilisez un nombre de couleurs pair, la couleur centrale constituant le remplissage de base. Distribuez les valeurs de manière homogène entre 0-127 et 129-255 pour vos couleurs, puis ajustez la valeur pour modifier la largeur de chaque bande du dégradé. Pour un biseau dégradé possédant neuf couleurs, le tableau [16, 32, 64, 96, 128, 160, 192, 224, 235] est possible. L'image suivante illustre le biseau dégradé décrit :




N'oubliez pas que la répartition des couleurs dans le dégradé varie en fonction des valeurs des propriétés `blurX`, `blurY`, `strength` et `quality`, ainsi que des valeurs de `ratios`.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

## Exemple

L'exemple suivant montre comment définir la propriété `ratios` sur une entité existante.

```
import flash.filters.GradientBevelFilter;

var mc:MovieClip = setUpFilter("ratiosExample");
mc.onPress = function() {
  var arr:Array = this.filters;
  var ratios:Array = [127, 128, 129];
  arr[0].ratios = ratios;
  this.filters = arr;
}
mc.onRelease = function() {
  var arr:Array = this.filters;
  var ratios:Array = [0, 128, 255];
  arr[0].ratios = ratios;
```

```
  this.filters = arr;
}

function setUpFilter(name:String):MovieClip {
  var art:MovieClip = this.createEmptyMovieClip(name,
  this.getNextHighestDepth());
  var w:Number = 150;
  var h:Number = 150;
  art.beginFill(0xCCCCCC);
  art.lineTo(w, 0);
  art.lineTo(w, h);
  art.lineTo(0, h);
  art.lineTo(0, 0);

  var colors:Array = [0xFFFFFF, 0xCCCCCC, 0x000000];
  var alphas:Array = [1, 0, 1];
  var ratios:Array = [0, 128, 255];
  var filter:GradientBevelFilter = new GradientBevelFilter(5, 225, colors,
  alphas, ratios, 5, 5, 5, 2, "inner", false);

  art.filters = new Array(filter);
  return art;
}
```

### Voir également

alphas (propriété GradientBevelFilter.alphas), colors (propriété GradientBevelFilter.colors), beginGradientFill (méthode MovieClip.beginGradientFill)

## strength (propriété GradientBevelFilter.strength)

```
public strength : nombre
```

L'intensité de l'impression ou du recouvrement. Plus la valeur est élevée, plus l'intensité des couleurs apparaît à l'impression et plus le contraste est important entre le biseau et l'arrière-plan. Les valeurs valides sont comprises entre 0 et 255. La valeur 0 signifie que le filtre n'est pas appliqué. La valeur par défaut est 1.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

### Exemple

L'exemple suivant montre comment définir la propriété `strength` sur un objet existant.

```
import flash.filters.GradientBevelFilter;

var mc:MovieClip = setUpFilter("strengthExample");
mc.onRelease = function() {
```

```
  var arr:Array = this.filters;
  arr[0].strength = 1;
  this.filters = arr;
}

function setUpFilter(name:String):MovieClip {
  var art:MovieClip = this.createEmptyMovieClip(name,
  this.getNextHighestDepth());
  var w:Number = 150;
  var h:Number = 150;
  art.beginFill(0xCCCCCC);
  art.lineTo(w, 0);
  art.lineTo(w, h);
  art.lineTo(0, h);
  art.lineTo(0, 0);

  var colors:Array = [0xFFFFFF, 0xCCCCCC, 0x000000];
  var alphas:Array = [1, 0, 1];
  var ratios:Array = [0, 128, 255];
  var filter:GradientBevelFilter = new GradientBevelFilter(5, 225, colors,
  alphas, ratios, 5, 5, 5, 3, "inner", false);

  art.filters = new Array(filter);
  return art;
}
```

### Voir également

ratios (propriété GradientBevelFilter.ratios)

## type (propriété GradientBevelFilter.type)

```
public type : String
```

Le positionnement de l'effet de biseau. Les valeurs possibles sont :

- `"outer"` : Biseau sur le bord extérieur de l'objet
- `"inner"` : Biseau sur le bord intérieur de l'objet
- `"full"` : Biseau sur le bord supérieur de l'objet

La valeur par défaut est `"inner"`.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

### Exemple

L'exemple suivant montre comment définir la propriété `type` sur un objet existant.

```
import flash.filters.GradientBevelFilter;

var mc:MovieClip = setUpFilter("typeExample");
```

```
mc.onRelease = function() {
  var arr:Array = this.filters;
  arr[0].type = "outer";
  this.filters = arr;
}

function setUpFilter(name:String):MovieClip {
  var art:MovieClip = this.createEmptyMovieClip(name,
  this.getNextHighestDepth());
  var w:Number = 150;
  var h:Number = 150;
  art.beginFill(0xCCCCCC);
  art.lineTo(w, 0);
  art.lineTo(w, h);
  art.lineTo(0, h);
  art.lineTo(0, 0);

  var colors:Array = [0xFFFFFF, 0xCCCCCC, 0x000000];
  var alphas:Array = [1, 0, 1];
  var ratios:Array = [0, 128, 255];
  var filter:GradientBevelFilter = new GradientBevelFilter(5, 225, colors,
  alphas, ratios, 5, 5, 5, 3, "inner", false);

  art.filters = new Array(filter);
  return art;
}
```

# GradientGlowFilter (flash.filters.GradientGlowFilter)

```
Object
  |
  +-flash.filters.BitmapFilter
    |
    +-flash.filters.GradientGlowFilter
```

```
public class GradientGlowFilter
extends BitmapFilter
```

La classe GradientGlowFilter vous permet d'appliquer un effet de rayonnement dégradé pour l'appliquer à divers objets dans Flash. L'aspect d'un rayonnement dégradé est réaliste et inclut un dégradé de couleurs que vous pouvez contrôler. Vous pouvez appliquer un rayonnement dégradé autour du bord intérieur ou extérieur d'un objet, ou encore sur son bord supérieur.

L'utilisation de filtres dépend de l'objet auquel vous appliquez le filtre.

- Pour appliquer des filtres aux clips, champs de texte et boutons lors de l'exécution, utilisez la propriété `filters`. Lorsque vous définissez la propriété `filters` d'un objet, celui-ci n'est pas modifié. En outre, vous pouvez l'annuler en supprimant la propriété `filters`.
- Pour appliquer des filtres aux occurrences BitmapData, utilisez la méthode `BitmapData.applyFilter()`. L'appel de `applyFilter()` sur un objet BitmapData génère une image filtrée à partir de l'objet BitmapData source et de l'objet Filter.

Vous pouvez également appliquer des effets de filtre aux images et aux données vidéo durant la programmation. Pour plus d'informations, consultez la documentation relative à la programmation.

Si vous appliquez un filtre à un clip ou à un bouton, la propriété `cacheAsBitmap` du clip ou du bouton est définie sur `true`. Si vous supprimez tous les filtres, la valeur d'origine de `cacheAsBitmap` est restaurée.

Ce filtre prend en charge le redimensionnement de la scène. Toutefois, le redimensionnement général, la rotation et l'inclinaison ne sont pas pris en charge ; si l'objet est redimensionné (si le redimensionnement `_xscale` et le redimensionnement `_yscale` ne sont pas définis sur 100 %), l'effet de filtre n'est pas redimensionné. Le redimensionnement est effectué uniquement en cas de zoom sur la scène.

Un filtre ne sera pas appliqué si l'image obtenue dépasse 2 880 pixels en largeur ou en hauteur. Par exemple, lorsque vous effectuez un zoom avant sur un clip de grande taille auquel un filtre est appliqué, le filtre sera désactivé si l'image obtenue dépasse la limite de 2 880 pixels.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

## Voir également

`ratios (propriété GradientGlowFilter.ratios)`, `applyFilter (méthode BitmapData.applyFilter)`, `cacheAsBitmap (propriété Button.cacheAsBitmap)`, `filters (propriété Button.filters)`, `GlowFilter (flash.filters.GlowFilter)`, `cacheAsBitmap (propriété MovieClip.cacheAsBitmap)`, `filters (propriété MovieClip.filters)`, `filters (propriété TextField.filters)`

## Résumé des propriétés

| Modificateurs | Propriété | Description |
|---|---|---|
| | `alphas:Array` | Un tableau de valeurs de transparence alpha pour les couleurs correspondantes dans le tableau `colors`. |
| | `angle:Number` | Angle exprimé en degrés. |
| | `blurX:Number` | Le niveau de flou horizontal. |
| | `blurY:Number` | Le niveau de flou vertical. |
| | `colors:Array` | Un tableau de couleurs définissant un dégradé. |
| | `distance:Number` | La distance de décalage du néon. |
| | `knockout:Boolean` | Spécifie si l'objet a un effet de poinçonnage. |
| | `quality:Number` | Le nombre d'applications du filtre. |
| | `ratios:Array` | Un tableau de taux de distribution des couleurs, pour les couleurs correspondantes dans le tableau `colors`. |
| | `strength:Number` | L'intensité de l'impression ou du recouvrement. |
| | `type:String` | Le positionnement de l'effet de filtre. |

*Propriétés héritées de la classe Object*

`constructeur (propriété Object.constructor), __proto__ (Object.__proto__ property), prototype (Object.prototype, propriété), __resolve (Object.__resolve, propriété)`

## Résumé des constructeurs

| Signature | Description |
|---|---|
| `GradientGlowFilter([distance:Number], [angle:Number], [colors:Array], [alphas:Array], [ratios:Array], [blurX:Number], [blurY:Number], [strength:Number], [quality:Number], [type:String], [knockout:Boolean])` | Initialise le filtre avec les paramètres spécifiés. |

### Résumé de la méthode

| Modificateurs | Signature | Description |
|---|---|---|
| | `clone() : GradientGlowFilter` | Renvoie une copie de cet objet filtre. |

*Méthodes héritées de la classe BitmapFilter*

`clone (méthode BitmapFilter.clone )`

*Méthodes héritées de la classe Object*

`addProperty (méthode Object.addProperty), hasOwnProperty (méthode Object.hasOwnProperty), isPropertyEnumerable (méthode Object.isPropertyEnumerable), isPrototypeOf (méthode Object.isPrototypeOf), registerClass (méthode Object.registerClass), toString (méthode Object.toString), unwatch (méthode Object.unwatch), valueOf (méthode Object.valueOf), watch (méthode Object.watch)`

## alphas (propriété GradientGlowFilter.alphas)

`public alphas : Array`

Un tableau de valeurs de transparence alpha pour les couleurs correspondantes dans le tableau `colors`. Les valeurs valides pour chaque élément du tableau sont comprises entre 0 et 1. Par exemple, 0,25 définit la valeur de transparence alpha à 25 %.

La propriété `alphas` ne peut pas être modifiée en manipulant directement ses valeurs. Vous devez obtenir une référence à `alphas`, effectuer les modifications sur la référence, puis définir la propriété `alphas` sur la référence.

Les propriétés `colors`, `alphas` et `ratios` sont toutes liées. Le premier élément du tableau `colors` correspond au premier élément du tableau `alphas`, du tableau `ratios`, etc.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

### Exemple

L'exemple suivant modifie la propriété `alphas` sur un clip existant lorsqu'un utilisateur clique dessus.

```
import flash.filters.GradientGlowFilter;
var mc:MovieClip = createGradientGlowRectangle("GlowAlphas");
mc.onRelease = function() {
  var filter:GradientGlowFilter = this.filters[0];
  var alphas:Array = filter.alphas;
  alphas.pop();
```

```
  alphas.pop();
  alphas.push(.3);
  alphas.push(1);
  filter.alphas = alphas;
  this.filters = new Array(filter);
}

function createGradientGlowRectangle(name:String):MovieClip {
  var art:MovieClip = this.createEmptyMovieClip(name,
  this.getNextHighestDepth());
  var w:Number = 100;
  var h:Number = 100;
  art.beginFill(0x003366);
  art.lineTo(w, 0);
  art.lineTo(w, h);
  art.lineTo(0, h);
  art.lineTo(0, 0);
  art._x = 20;
  art._y = 20;

  var colors:Array = [0xFFFFFF, 0xFF0000, 0xFFFF00, 0x00CCFF];
  var alphas:Array = [0, 1, 1, 1];
  var ratios:Array = [0, 63, 126, 255];
  var filter:GradientGlowFilter = new GradientGlowFilter(0, 45, colors,
  alphas, ratios, 55, 55, 2.5, 2, "outer", false);
  var filterArray:Array = new Array();
  filterArray.push(filter);
  art.filters = filterArray;
  return art;
}
```

### Voir également

colors (propriété GradientGlowFilter.colors), ratios (propriété GradientGlowFilter.ratios)

## angle (propriété GradientGlowFilter.angle)

```
public angle : nombre
```

Angle exprimé en degrés. Les valeurs valides sont comprises entre 0 et 360. La valeur par défaut est 45.

La valeur d'angle représente l'angle de la source lumineuse théorique éclairant l'objet et détermine l'emplacement de l'effet par rapport à l'objet. Si la valeur de `distance` est 0, il n'y a pas d'effet de décalage par rapport à l'objet. La propriété `angle` n'a donc aucun effet.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

### Exemple

L'exemple suivant modifie la propriété `angle` sur un clip existant lorsqu'un utilisateur clique dessus.

```
import flash.filters.GradientGlowFilter;
var mc:MovieClip = createGradientGlowRectangle("GlowAngle");
mc.onRelease = function() {
  var filter:GradientGlowFilter = this.filters[0];
  filter.distance = 50;
  filter.angle = 90;
  this.filters = new Array(filter);
}

function createGradientGlowRectangle(name:String):MovieClip {
  var art:MovieClip = this.createEmptyMovieClip(name,
  this.getNextHighestDepth());
  var w:Number = 100;
  var h:Number = 100;
  art.beginFill(0x003366);
  art.lineTo(w, 0);
  art.lineTo(w, h);
  art.lineTo(0, h);
  art.lineTo(0, 0);
  art._x = 20;
  art._y = 20;

  var colors:Array = [0xFFFFFF, 0xFF0000, 0xFFFF00, 0x00CCFF];
  var alphas:Array = [0, 1, 1, 1];
  var ratios:Array = [0, 63, 126, 255];
  var filter:GradientGlowFilter = new GradientGlowFilter(0, 45, colors,
  alphas, ratios, 55, 55, 2.5, 2, "outer", false);
  var filterArray:Array = new Array();
  filterArray.push(filter);
  art.filters = filterArray;
  return art;
}
```

## blurX (propriété GradientGlowFilter.blurX)

```
public blurX : nombre
```

Le niveau de flou horizontal. Les valeurs valides sont comprises entre 0 et 255. Un flou d'une valeur inférieure ou égale à 1 signifie que l'image d'origine n'est pas modifiée avant d'être copiée. La valeur par défaut est 4. Les valeurs correspondant à une puissance de 2 (2, 4, 8, 16 et 32) sont optimisées pour obtenir un rendu plus rapide qu'avec les autres valeurs.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

### Exemple

L'exemple suivant modifie la propriété `blurX` sur un clip existant lorsqu'un utilisateur clique dessus.

```
import flash.filters.GradientGlowFilter;
var mc:MovieClip = createGradientGlowRectangle("GlowBlurX");
mc.onRelease = function() {
  var filter:GradientGlowFilter = this.filters[0];
  filter.blurX = 255;
  this.filters = new Array(filter);
}

function createGradientGlowRectangle(name:String):MovieClip {
  var art:MovieClip = this.createEmptyMovieClip(name,
  this.getNextHighestDepth());
  var w:Number = 100;
  var h:Number = 100;
  art.beginFill(0x003366);
  art.lineTo(w, 0);
  art.lineTo(w, h);
  art.lineTo(0, h);
  art.lineTo(0, 0);
  art._x = 20;
  art._y = 20;

  var colors:Array = [0xFFFFFF, 0xFF0000, 0xFFFF00, 0x00CCFF];
  var alphas:Array = [0, 1, 1, 1];
  var ratios:Array = [0, 63, 126, 255];
  var filter:GradientGlowFilter = new GradientGlowFilter(0, 45, colors,
  alphas, ratios, 55, 55, 2.5, 2, "outer", false);
  var filterArray:Array = new Array();
  filterArray.push(filter);
  art.filters = filterArray;
  return art;
}
```

## blurY (propriété GradientGlowFilter.blurY)

```
public blurY : nombre
```

Le niveau de flou vertical. Les valeurs valides sont comprises entre 0 et 255. Un flou d'une valeur inférieure ou égale à 1 signifie que l'image d'origine n'est pas modifiée avant d'être copiée. La valeur par défaut est 4. Les valeurs correspondant à une puissance de 2 (2, 4, 8, 16 et 32) sont optimisées pour obtenir un rendu plus rapide qu'avec les autres valeurs.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

### Exemple

L'exemple suivant modifie la propriété `blurY` sur un clip existant lorsqu'un utilisateur clique dessus.

```
import flash.filters.GradientGlowFilter;
var mc:MovieClip = createGradientGlowRectangle("GlowBlurY");
mc.onRelease = function() {
  var filter:GradientGlowFilter = this.filters[0];
  filter.blurY = 255;
  this.filters = new Array(filter);
}

function createGradientGlowRectangle(name:String):MovieClip {
  var art:MovieClip = this.createEmptyMovieClip(name,
  this.getNextHighestDepth());
  var w:Number = 100;
  var h:Number = 100;
  art.beginFill(0x003366);
  art.lineTo(w, 0);
  art.lineTo(w, h);
  art.lineTo(0, h);
  art.lineTo(0, 0);
  art._x = 20;
  art._y = 20;

  var colors:Array = [0xFFFFFF, 0xFF0000, 0xFFFF00, 0x00CCFF];
  var alphas:Array = [0, 1, 1, 1];
  var ratios:Array = [0, 63, 126, 255];
  var filter:GradientGlowFilter = new GradientGlowFilter(0, 45, colors,
  alphas, ratios, 55, 55, 2.5, 2, "outer", false);
  var filterArray:Array = new Array();
  filterArray.push(filter);
  art.filters = filterArray;
  return art;
}
```

## clone (méthode GradientGlowFilter.clone)

```
public clone() : GradientGlowFilter
```

Renvoie une copie de cet objet filtre.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

### Valeur renvoyée

`flash.filters.GradientGlowFilter` - Nouvelle instance GradientGlowFilter dont toutes les propriétés sont identiques à celles de l'instance GradientGlowFilter d'origine.

## Exemple

L'exemple suivant crée trois objets GradientGlowFilter et les compare. `filter_1` est créé à l'aide du constructeur GradientGlowFilter; `filter_2` est créé en le définissant comme égal à `filter_1` et `clonedFilter` est créé par clonage de `filter_1`. Vous remarquerez que si `filter_2` est évalué comme égal à `filter_1`, ce n'est pas le cas de `clonedFilter`, qui contient pourtant les mêmes valeurs que `filter_1`.

```
import flash.filters.GradientGlowFilter;

var colors:Array = [0xFFFFFF, 0xFF0000, 0xFFFF00, 0x00CCFF];
var alphas:Array = [0, 1, 1, 1];
var ratios:Array = [0, 63, 126, 255];
var filter_1:GradientGlowFilter = new GradientGlowFilter(0, 45, colors,
  alphas, ratios, 55, 55, 2.5, 2, "outer", false);
var filter_2:GradientGlowFilter = filter_1;
var clonedFilter:GradientGlowFilter = filter_1.clone();

trace(filter_1 == filter_2); // true
trace(filter_1 == clonedFilter); // false

for(var i in filter_1) {
  trace(">> " + i + ": " + filter_1[i]);
  // >> clone: [type Function]
  // >> type: outer
  // >> knockout: false
  // >> strength: 2.5
  // >> quality: 2
  // >> blurY: 55
  // >> blurX: 55
  // >> ratios: 0,63,126,255
  // >> alphas: 0,1,1,1
  // >> colors: 16777215,16711680,16776960,52479
  // >> angle: 45
  // >> distance: 0
}

for(var i in clonedFilter) {
  trace(">> " + i + ": " + clonedFilter[i]);
  // >> clone: [type Function]
  // >> type: outer
  // >> knockout: false
  // >> strength: 2.5
  // >> quality: 2
  // >> blurY: 55
  // >> blurX: 55
  // >> ratios: 0,63,126,255
  // >> alphas: 0,1,1,1
  // >> colors: 16777215,16711680,16776960,52479
  // >> angle: 45
```

```
  // >> distance: 0
}
```

Pour illustrer plus précisément la relation entre `filter_1`, `filter_2` et `clonedFilter`, l'exemple suivant modifie la propriété `knockout` de `filter_1`. La modification de `knockout` démontre que la méthode `clone()` crée une nouvelle occurrence reposant sur les valeurs de `filter_1` au lieu de pointer vers elles par référence.

```
import flash.filters.GradientGlowFilter;

var colors:Array = [0xFFFFFF, 0xFF0000, 0xFFFF00, 0x00CCFF];
var alphas:Array = [0, 1, 1, 1];
var ratios:Array = [0, 63, 126, 255];
var filter_1:GradientGlowFilter = new GradientGlowFilter(0, 45, colors,
  alphas, ratios, 55, 55, 2.5, 2, "outer", false);
var filter_2:GradientGlowFilter = filter_1;
var clonedFilter:GradientGlowFilter = filter_1.clone();

trace(filter_1.knockout); // false
trace(filter_2.knockout); // false
trace(clonedFilter.knockout); // false

filter_1.knockout = true;

trace(filter_1.knockout); // true
trace(filter_2.knockout); // true
trace(clonedFilter.knockout); // false
```

# colors (propriété GradientGlowFilter.colors)

```
public colors : Array
```

Un tableau de couleurs définissant un dégradé. Par exemple, rouge correspond à 0xFF0000, bleu à 0x0000FF, etc.

La propriété `colors` ne peut pas être modifiée en manipulant directement ses valeurs. Vous devez obtenir une référence à `colors`, effectuer les modifications sur la référence, puis définir la propriété `colors` sur la référence.

Les propriétés `colors`, `alphas` et `ratios` sont toutes liées. Le premier élément du tableau `colors` correspond au premier élément du tableau `alphas`, du tableau `ratios`, etc.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

## Exemple

L'exemple suivant modifie la propriété `colors` sur un clip existant lorsqu'un utilisateur clique dessus.

```
import flash.filters.GradientGlowFilter;
```

```
var mc:MovieClip = createGradientGlowRectangle("GlowColors");
mc.onRelease = function() {
  var filter:GradientGlowFilter = this.filters[0];
  var colors:Array = filter.colors;
  colors.pop();
  colors.push(0xFF00FF);
  filter.colors = colors;
  this.filters = new Array(filter);
}

function createGradientGlowRectangle(name:String):MovieClip {
  var art:MovieClip = this.createEmptyMovieClip(name,
  this.getNextHighestDepth());
  var w:Number = 100;
  var h:Number = 100;
  art.beginFill(0x003366);
  art.lineTo(w, 0);
  art.lineTo(w, h);
  art.lineTo(0, h);
  art.lineTo(0, 0);
  art._x = 20;
  art._y = 20;

  var colors:Array = [0xFFFFFF, 0xFF0000, 0xFFFF00, 0x00CCFF];
  var alphas:Array = [0, 1, 1, 1];
  var ratios:Array = [0, 63, 126, 255];
  var filter:GradientGlowFilter = new GradientGlowFilter(0, 45, colors,
  alphas, ratios, 55, 55, 2.5, 2, "outer", false);
  var filterArray:Array = new Array();
  filterArray.push(filter);
  art.filters = filterArray;
  return art;
}
```

## Voir également

alphas (propriété GradientGlowFilter.alphas), ratios (propriété GradientGlowFilter.ratios)

# distance (propriété GradientGlowFilter.distance)

```
public distance : nombre
```

La distance de décalage du néon. La valeur par défaut est 4.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

## Exemple

L'exemple suivant modifie la propriété `distance` sur un clip existant lorsqu'un utilisateur clique dessus.

```
import flash.filters.GradientGlowFilter;
var mc:MovieClip = createGradientGlowRectangle("GlowDistance");
mc.onRelease = function() {
  var filter:GradientGlowFilter = this.filters[0];
  filter.distance = 20;
  this.filters = new Array(filter);
}

function createGradientGlowRectangle(name:String):MovieClip {
  var art:MovieClip = this.createEmptyMovieClip(name,
  this.getNextHighestDepth());
  var w:Number = 100;
  var h:Number = 100;
  art.beginFill(0x003366);
  art.lineTo(w, 0);
  art.lineTo(w, h);
  art.lineTo(0, h);
  art.lineTo(0, 0);
  art._x = 20;
  art._y = 20;

  var colors:Array = [0xFFFFFF, 0xFF0000, 0xFFFF00, 0x00CCFF];
  var alphas:Array = [0, 1, 1, 1];
  var ratios:Array = [0, 63, 126, 255];
  var filter:GradientGlowFilter = new GradientGlowFilter(0, 45, colors,
  alphas, ratios, 55, 55, 2.5, 2, "outer", false);
  var filterArray:Array = new Array();
  filterArray.push(filter);
  art.filters = filterArray;
  return art;
}
```

# Constructeur GradientGlowFilter

```
public GradientGlowFilter([distance:Number], [angle:Number],
  [colors:Array], [alphas:Array], [ratios:Array], [blurX:Number],
  [blurY:Number], [strength:Number], [quality:Number], [type:String],
  [knockout:Boolean])
```

Initialise le filtre avec les paramètres spécifiés.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

## Paramètres

*distance*:Number [facultatif] - Distance de décalage du néon. La valeur par défaut est 4.

*angle*:Number [facultatif] - Angle exprimé en degrés. Les valeurs valides sont comprises entre 0 et 360. La valeur par défaut est 45.

*colors*:Array [facultatif] - Tableau de couleurs définissant un dégradé. Par exemple, rouge correspond à 0xFF0000, bleu à 0x0000FF, etc.

*alphas*:Array [facultatif] - Tableau de valeurs de transparence alpha pour les couleurs correspondantes dans le tableau colors. Les valeurs valides pour chaque élément du tableau sont comprises entre 0 et 1. Par exemple, 0,25 définit la valeur de transparence alpha à 25 %.

*ratios*:Array [facultatif] - Tableau des taux de distribution des couleurs. Les valeurs valides sont comprises entre 0 et 255. Cette valeur définit le pourcentage de la largeur où la couleur est échantillonnée à 100 %.

*blurX*:Number [facultatif] - Niveau de flou horizontal. Les valeurs valides sont comprises entre 0 et 255. Un flou d'une valeur inférieure ou égale à 1 signifie que l'image d'origine n'est pas modifiée avant d'être copiée. La valeur par défaut est 4. Les valeurs correspondant à une puissance de 2 (2, 4, 8, 16 et 32) sont optimisées pour obtenir un rendu plus rapide qu'avec les autres valeurs.

*blurY*:Number [facultatif] - Niveau de flou vertical. Les valeurs valides sont comprises entre 0 et 255. Un flou d'une valeur inférieure ou égale à 1 signifie que l'image d'origine n'est pas modifiée avant d'être copiée. La valeur par défaut est 4. Les valeurs correspondant à une puissance de 2 (2, 4, 8, 16 et 32) sont optimisées pour obtenir un rendu plus rapide qu'avec les autres valeurs.

*strength*:Number [facultatif] - Intensité de l'impression ou du recouvrement. Plus la valeur est élevée, plus l'intensité des couleurs apparaît à l'impression et plus le contraste est important entre le rayonnement et l'arrière-plan. Les valeurs valides sont comprises entre 0 et 255. Plus la valeur est élevée, plus l'intensité de l'impression est élevée. La valeur 0 signifie que le filtre n'est pas appliqué. La valeur par défaut est 1.

*quality*:Number [facultatif] - Nombre d'applications du filtre. Les valeurs valides sont comprises entre 0 et 15. La valeur par défaut, 1, correspond à une qualité faible. La valeur 2 offre une qualité moyenne et la valeur 3, une qualité élevée.

*type*:String [facultatif] - Positionnement de l'effet de filtre. Les valeurs possibles sont :

- "outer" : Rayonnement sur le bord extérieur de l'objet
- "inner" : Rayonnement sur le bord intérieur de l'objet ; il s'agit de la valeur par défaut
- "full" : Rayonnement sur le bord supérieur de l'objet

La valeur par défaut est "inner".

*knockout*:Boolean [facultatif] - Spécifie si l'objet a un effet de poinçonnage. Un effet de poinçonnage rend le remplissage de l'objet transparent et révèle la couleur d'arrière-plan du document. La valeur true spécifie un effet de poinçonnage ; la valeur par défaut est false (pas d'effet de poinçonnage).

## Exemple

L'exemple suivant crée un nouveau filtre de rayonnement dégradé, affecte ses valeurs et l'applique à un rectangle plat.

```
import flash.filters.GradientGlowFilter;
var art:MovieClip = createRectangle(100, 100, 0x003366,
  "gradientGlowFilterExample");
var distance:Number = 0;
var angleInDegrees:Number = 45;
var colors:Array = [0xFFFFFF, 0xFF0000, 0xFFFF00, 0x00CCFF];
var alphas:Array = [0, 1, 1, 1];
var ratios:Array = [0, 63, 126, 255];
var blurX:Number = 50;
var blurY:Number = 50;
var strength:Number = 2.5;
var quality:Number = 3;
var type:String = "outer";
var knockout:Boolean = false;

var filter:GradientGlowFilter = new GradientGlowFilter(distance,
                                angleInDegrees,
                                colors,
                                alphas,
                                ratios,
                                blurX,
                                blurY,
                                strength,
                                quality,
                                type,
                                knockout);
var filterArray:Array = new Array();
```

```
filterArray.push(filter);
art.filters = filterArray;

function createRectangle(w:Number, h:Number, bgColor:Number,
  name:String):MovieClip {
  var mc:MovieClip = this.createEmptyMovieClip(name,
  this.getNextHighestDepth());
  mc.beginFill(bgColor);
  mc.lineTo(w, 0);
  mc.lineTo(w, h);
  mc.lineTo(0, h);
  mc.lineTo(0, 0);
  mc._x = 20;
  mc._y = 20;
  return mc;
}
```

## knockout (propriété GradientGlowFilter.knockout)

```
public knockout : Boolean
```

Spécifie si l'objet a un effet de poinçonnage. Un effet de poinçonnage rend le remplissage de l'objet transparent et révèle la couleur d'arrière-plan du document. La valeur `true` spécifie un effet de poinçonnage ; la valeur par défaut est `false` (pas d'effet de poinçonnage).

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

### Exemple

L'exemple suivant modifie la propriété `knockout` sur un clip existant lorsqu'un utilisateur clique dessus.

```
import flash.filters.GradientGlowFilter;
var mc:MovieClip = createGradientGlowRectangle("GlowKnockout");
mc.onRelease = function() {
  var filter:GradientGlowFilter = this.filters[0];
  filter.knockout = true;
  this.filters = new Array(filter);
}

function createGradientGlowRectangle(name:String):MovieClip {
  var art:MovieClip = this.createEmptyMovieClip(name,
  this.getNextHighestDepth());
  var w:Number = 100;
  var h:Number = 100;
  art.beginFill(0x003366);
  art.lineTo(w, 0);
  art.lineTo(w, h);
  art.lineTo(0, h);
  art.lineTo(0, 0);
```

```
  art._x = 20;
  art._y = 20;

  var colors:Array = [0xFFFFFF, 0xFF0000, 0xFFFF00, 0x00CCFF];
  var alphas:Array = [0, 1, 1, 1];
  var ratios:Array = [0, 63, 126, 255];
  var filter:GradientGlowFilter = new GradientGlowFilter(0, 45, colors,
  alphas, ratios, 55, 55, 2.5, 2, "outer", false);
  var filterArray:Array = new Array();
  filterArray.push(filter);
  art.filters = filterArray;
  return art;
}
```

## quality (propriété GradientGlowFilter.quality)

```
public quality : nombre
```

Le nombre d'applications du filtre. Les valeurs valides sont comprises entre 0 et 15. La valeur par défaut, 1, correspond à une qualité faible. La valeur 2 offre une qualité moyenne et la valeur 3, une qualité élevée. Les rendus des filtres de valeurs inférieures sont obtenus plus rapidement.

Pour la plupart des applications, une valeur de `quality` de 1, 2 ou 3 est suffisante. Il est possible d'utiliser des valeurs numériques jusqu'à 15 pour obtenir différents effets, toutefois le rendu des valeurs les plus élevées est moins rapide. Sans augmenter la valeur de `quality`, vous pouvez généralement obtenir un effet similaire, avec un rendu plus rapide, en augmentant simplement les valeurs de `blurX` et `blurY`.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

### Exemple

L'exemple suivant modifie la propriété `quality` sur un clip existant lorsqu'un utilisateur clique dessus.

```
import flash.filters.GradientGlowFilter;
var mc:MovieClip = createGradientGlowRectangle("GlowQuality");
mc.onRelease = function() {
  var filter:GradientGlowFilter = this.filters[0];
  filter.quality = 3;
  this.filters = new Array(filter);
}

function createGradientGlowRectangle(name:String):MovieClip {
  var art:MovieClip = this.createEmptyMovieClip(name,
  this.getNextHighestDepth());
  var w:Number = 100;
```

```
    var h:Number = 100;
    art.beginFill(0x003366);
    art.lineTo(w, 0);
    art.lineTo(w, h);
    art.lineTo(0, h);
    art.lineTo(0, 0);
    art._x = 20;
    art._y = 20;

    var colors:Array = [0xFFFFFF, 0xFF0000, 0xFFFF00, 0x00CCFF];
    var alphas:Array = [0, 1, 1, 1];
    var ratios:Array = [0, 63, 126, 255];
    var filter:GradientGlowFilter = new GradientGlowFilter(0, 45, colors,
    alphas, ratios, 55, 55, 2.5, 2, "outer", false);
    var filterArray:Array = new Array();
    filterArray.push(filter);
    art.filters = filterArray;
    return art;
}
```

## ratios (propriété GradientGlowFilter.ratios)

```
public ratios : Array
```

Un tableau de taux de distribution des couleurs, pour les couleurs correspondantes dans le tableau `colors`. Les valeurs valides sont comprises entre 0 et 255.

La propriété `ratios` ne peut pas être modifiée en manipulant directement ses valeurs. Vous devez obtenir une référence à `ratios`, effectuer les modifications sur la référence, puis définir la propriété `ratios` sur la référence.

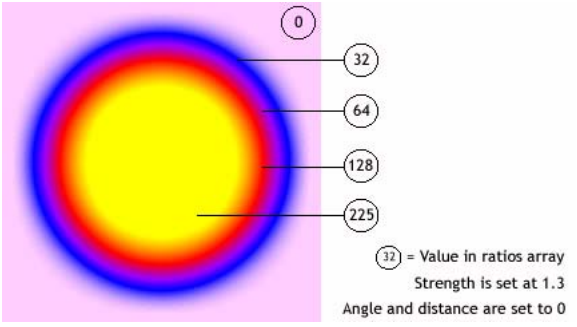
Les propriétés `colors`, `alphas` et `ratios` sont toutes liées. Le premier élément du tableau `colors` correspond au premier élément du tableau `alphas`, du tableau `ratios`, etc.

Considérez le filtre de rayonnement dégradé comme un rayonnement émanant du centre de l'objet (si la valeur `distance` est définie sur 0), avec des bandes de couleurs se mêlant les unes aux autres pour constituer le dégradé. La première couleur du tableau `colors` est la couleur la plus proche de l'extérieur du rayonnement. La dernière couleur est la plus au centre du rayonnement.

Chaque valeur du tableau `ratios` définit l'emplacement de la couleur sur le rayon du dégradé, 0 représentant le point le plus éloigné et 255, le point le plus proche du centre du dégradé. Les valeurs de ratio s'échelonnent progressivement de 0 à 255 pixels : par exemple [0, 64, 128, 200, 255]. Les valeurs comprises entre 0 et 128 apparaissent sur les bords extérieurs du rayonnement. Les valeurs comprises entre 129 et 255 apparaissent sur les bords intérieurs du rayonnement. Selon les valeurs de ratio des couleurs et la valeur `type` du filtre, les couleurs du filtre peuvent être obscurcies par l'objet auquel le filtre est appliqué.

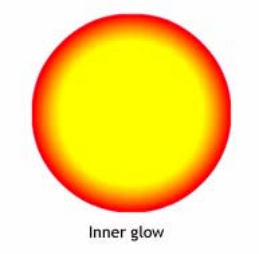
Dans le code et l'image suivante, un filtre est appliqué à un cercle noir animé, le type étant défini avec la valeur `"full"`. Pour les besoins de la démonstration, la première couleur du tableau `colors`, rose, a une valeur `alpha` de 1 pour qu'elle se détache bien sur le fond blanc du document. (Dans la pratique, vous ne choisirez probablement pas la première couleur ainsi.) Vous pouvez constater que la dernière couleur du tableau, jaune, obscurcit le cercle noir auquel le filtre est appliqué :

```
var colors = [0xFFCCFF, 0x0000FF, 0x9900FF, 0xFF0000, 0xFFFF00]; var alphas
= [1, 1, 1, 1, 1]; var ratios = [0, 32, 64, 128, 225]; var myGGF = new
GradientGlowFilter(0, 0, colors, alphas, ratios, 50, 50, 1, 2, "full",
false);
```




Pour obtenir un effet de transparence sur le fond de votre document lorsque vous définissez la valeur `type` sur `"outer"` ou `"full"`, définissez la première couleur du tableau comme identique à celle de l'arrière-plan du document, ou définissez la valeur alpha de la première couleur sur 0. Ces deux techniques entraînent le mélange du filtre avec l'arrière-plan.

Deux petites modifications au code peuvent se traduire par un effet du rayonnement complètement différent et ce, même si les tableaux `ratios` et `colors` restent les mêmes. Définissez la valeur alpha de la première colonne du tableau sur 0 pour que le filtre se mélange à l'arrière-plan blanc du document ; puis définissez la propriété `type` sur `"outer"` ou `"inner"`. Observez le résultat sur les illustrations suivantes.


N'oubliez pas que la répartition des couleurs dans le dégradé varie en fonction des valeurs des propriétés `blurX`, `blurY`, `strength` et `quality`, ainsi que des valeurs de `ratios`.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

## Exemple

L'exemple suivant modifie la propriété `ratios` sur un clip existant lorsqu'un utilisateur clique dessus.

```
import flash.filters.GradientGlowFilter;
var mc:MovieClip = createGradientGlowRectangle("GlowRatios");
mc.onRelease = function() {
  var filter:GradientGlowFilter = this.filters[0];
  var ratios:Array = filter.ratios;
  ratios.shift();
  ratios.unshift(40);
  filter.ratios = ratios;
  this.filters = new Array(filter);
```

```
}

function createGradientGlowRectangle(name:String):MovieClip {
  var art:MovieClip = this.createEmptyMovieClip(name,
  this.getNextHighestDepth());
  var w:Number = 100;
  var h:Number = 100;
  art.beginFill(0x003366);
  art.lineTo(w, 0);
  art.lineTo(w, h);
  art.lineTo(0, h);
  art.lineTo(0, 0);
  art._x = 20;
  art._y = 20;

  var colors:Array = [0xFFFFFF, 0xFF0000, 0xFFFF00, 0x00CCFF];
  var alphas:Array = [0, 1, 1, 1];
  var ratios:Array = [0, 63, 126, 255];
  var filter:GradientGlowFilter = new GradientGlowFilter(0, 45, colors,
  alphas, ratios, 55, 55, 2.5, 2, "outer", false);
  var filterArray:Array = new Array();
  filterArray.push(filter);
  art.filters = filterArray;
  return art;
}
```

### Voir également

colors (propriété GradientGlowFilter.colors), alphas (propriété GradientGlowFilter.alphas), beginGradientFill (méthode MovieClip.beginGradientFill)

## strength (propriété GradientGlowFilter.strength)

```
public strength : nombre
```

L'intensité de l'impression ou du recouvrement. Plus la valeur est élevée, plus l'intensité des couleurs apparaît à l'impression et plus le contraste est important entre le rayonnement et l'arrière-plan. Les valeurs valides sont comprises entre 0 et 255. La valeur 0 signifie que le filtre n'est pas appliqué. La valeur par défaut est 1.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

### Exemple

L'exemple suivant modifie la propriété `strength` sur un clip existant lorsqu'un utilisateur clique dessus.

```
import flash.filters.GradientGlowFilter;
var mc:MovieClip = createGradientGlowRectangle("GlowStrength");
mc.onRelease = function() {
  var filter:GradientGlowFilter = this.filters[0];
  filter.strength = 1;
  this.filters = new Array(filter);
}

function createGradientGlowRectangle(name:String):MovieClip {
  var art:MovieClip = this.createEmptyMovieClip(name,
  this.getNextHighestDepth());
  var w:Number = 100;
  var h:Number = 100;
  art.beginFill(0x003366);
  art.lineTo(w, 0);
  art.lineTo(w, h);
  art.lineTo(0, h);
  art.lineTo(0, 0);
  art._x = 20;
  art._y = 20;

  var colors:Array = [0xFFFFFF, 0xFF0000, 0xFFFF00, 0x00CCFF];
  var alphas:Array = [0, 1, 1, 1];
  var ratios:Array = [0, 63, 126, 255];
  var filter:GradientGlowFilter = new GradientGlowFilter(0, 45, colors,
  alphas, ratios, 55, 55, 2.5, 2, "outer", false);
  var filterArray:Array = new Array();
  filterArray.push(filter);
  art.filters = filterArray;
  return art;
}
```

## type (propriété GradientGlowFilter.type)

```
public type : String
```

Le positionnement de l'effet de filtre. Les valeurs possibles sont :

- `"outer"` : Rayonnement sur le bord extérieur de l'objet
- `"inner"` : Rayonnement sur le bord intérieur de l'objet ; il s'agit de la valeur par défaut
- `"full"` : Rayonnement sur le bord supérieur de l'objet

La valeur par défaut est `"inner"`.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

### Exemple

L'exemple suivant modifie la propriété `type` sur un clip existant lorsqu'un utilisateur clique dessus.

```
import flash.filters.GradientGlowFilter;
var mc:MovieClip = createGradientGlowRectangle("GlowType");
mc.onRelease = function() {
  var filter:GradientGlowFilter = this.filters[0];
  filter.type = "inner";
  filter.strength = 1;
  this.filters = new Array(filter);
}

function createGradientGlowRectangle(name:String):MovieClip {
  var art:MovieClip = this.createEmptyMovieClip(name,
  this.getNextHighestDepth());
  var w:Number = 100;
  var h:Number = 100;
  art.beginFill(0x003366);
  art.lineTo(w, 0);
  art.lineTo(w, h);
  art.lineTo(0, h);
  art.lineTo(0, 0);
  art._x = 20;
  art._y = 20;

  var colors:Array = [0xFFFFFF, 0xFF0000, 0xFFFF00, 0x00CCFF];
  var alphas:Array = [0, 1, 1, 1];
  var ratios:Array = [0, 63, 126, 255];
  var filter:GradientGlowFilter = new GradientGlowFilter(0, 45, colors,
  alphas, ratios, 55, 55, 2.5, 2, "outer", false);
  var filterArray:Array = new Array();
  filterArray.push(filter);
  art.filters = filterArray;
  return art;
}
```

# IME (System.IME)

```
Object
  |
  +-System.IME
```

```
public class IME
extends Object
```

La classe IME permet de manipuler directement l'IME (Input Method Editor) du système d'exploitation sous lequel l'application Flash Player s'exécute sur l'ordinateur client. Vous pouvez déterminer si un IME est installé, qu'il soit activé ou non, et quel IME est activé. Vous pouvez désactiver ou activer l'IME dans l'application Flash Player et exécuter d'autres fonctions limitées, selon votre système d'exploitation.

Les IME permettent aux utilisateurs de taper des caractères de texte non ASCII en langues asiatiques, tels que le Chinois, le Japonais et le Coréen. Pour plus d'informations sur les IME, consultez la documentation relative au système d'exploitation correspondant à la plate-forme pour laquelle vous développez des applications. Vous trouverez des ressources supplémentaires d'informations concernant les méthodes d'entrée aux adresses suivantes :

- http://www.microsoft.com/globaldev/default.mspx
- http://developer.apple.com/documentation/
- http://java.sun.com

Le tableau ci-après indique les plates-formes prises en charge par cette classe :

| Fonction | Windows | Macintosh OSX | Macintosh Classic | Linux / Solaris XIM |
|---|---|---|---|---|
| Déterminer si l'IME est installé `System.capabilities.hasIME` | Oui | Oui | Oui | Oui |
| Activer ou désactiver l'IME `System.IME.setEnabled()` | Oui | Oui | Oui | Oui |
| Déterminer si l'IME est activé ou désactivé `System.IME.getEnabled()` | Oui | Oui | Oui | Oui |
| Définir le mode de conversion de l'IME `System.IME.setConversionMode()` | Oui | Oui ** | Non | Oui |

| Fonction | Windows | Macintosh OSX | Macintosh Classic | Linux / Solaris XIM |
|---|---|---|---|---|
| Obtenir le mode de conversion de l'IME `System.IME.getConversionMode()` | Oui | Oui ** | Non | Oui |
| Envoyer la chaîne à convertir à l'IME `System.IME.setCompositionString()` | Oui * | Non | Non | Oui |
| Obtenir la chaîne d'origine auprès de l'IME avant sa conversion `System.IME.addListener()` `listener.onIMEComposition()` `System.IME.removeListener()` | Oui * | Non | Non | Oui |
| Envoyer une requête de conversion à l'IME `System.IME.doConversion()` | Oui * | Non | Non | Oui |

* Ces opérations ne sont pas toutes prises en charge par tous les IME de Windows. Jusqu'à présent, seul l'IME japonais les prend toutes en charge. La prise en charge des appels du système d'exploitation diffère selon les IME.

** Sous Macintosh, ces méthodes sont uniquement prises en charge pour le japonais ; elles ne sont pas prises en charge pour les IME tiers.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

## Résumé des propriétés

| Modificateurs | Propriété | Description |
|---|---|---|
| static | ALPHANUMERIC_FULL:String | Une chaîne ayant la valeur "ALPHANUMERIC_FULL" à utiliser avec setConversionMode() et getConversionMode(). |
| static | ALPHANUMERIC_HALF:String | Une chaîne ayant la valeur "ALPHANUMERIC_HALF" à utiliser avec setConversionMode() et getConversionMode(). |
| static | CHINESE:String | Une chaîne ayant la valeur "CHINESE" à utiliser avec setConversionMode() et getConversionMode(). |
| static | JAPANESE_HIRAGANA:String | Une chaîne ayant la valeur "JAPANESE_HIRAGANA" à utiliser avec setConversionMode() et getConversionMode(). |
| static | JAPANESE_KATAKANA_FULL:String | Une chaîne ayant la valeur "JAPANESE_KATAKANA_FULL" à utiliser avec setConversionMode() et getConversionMode(). |
| static | JAPANESE_KATAKANA_HALF:String | Une chaîne ayant la valeur "JAPANESE_KATAKANA_HALF" à utiliser avec setConversionMode() et getConversionMode(). |
| static | KOREAN:String | Une chaîne ayant la valeur "KOREAN" à utiliser avec setConversionMode() et getConversionMode(). |
| static | UNKNOWN:String | Une chaîne ayant la valeur "UNKNOWN" à utiliser avec getConversionMode(). |

*Propriétés héritées de la classe Object*

constructeur (propriété Object.constructor), __proto__ (Object.__proto__ property), prototype (Object.prototype, propriété), __resolve (Object.__resolve, propriété)

## Résumé des événements

| Evénement | Description |
|---|---|
| onIMEComposition = function([readingString:String]) {} | Notifié lorsque la chaîne de composition IME est en cours de définition. |

Résumé de la méthode

| Modificateurs | Signature | Description |
|---|---|---|
| static | addListener(listener:Object) : Void | Enregistre un objet pour qu'il reçoive une notification lorsqu'un gestionnaire d'événements IME est appelé par l'événement onIMEComposition. |
| static | doConversion() : Boolean | Demande à l'IME de sélectionner le premier candidat pour la chaîne de composition actuelle. |
| static | getConversionMode() : String | Indique le mode de conversion de l'IME actuel. |
| static | getEnabled() : Boolean | Indique si le système IME est activé. |
| static | removeListener(listener:Object) : Boolean | Supprime un objet d'écoute qui avait été enregistré dans une instance IME avec IME.addListener(). |
| static | setCompositionString(composition:String) : Boolean | Définit la chaîne de composition IME. |
| static | setConversionMode(mode:String) : Boolean | Définit le mode de conversion de l'IME actuel. |
| static | setEnabled(enabled:Boolean) : Boolean | Active ou désactive le système IME. |

*Méthodes héritées de la classe Object*

addProperty (méthode Object.addProperty), hasOwnProperty (méthode Object.hasOwnProperty), isPropertyEnumerable (méthode Object.isPropertyEnumerable), isPrototypeOf (méthode Object.isPrototypeOf), registerClass (méthode Object.registerClass), toString (méthode Object.toString), unwatch (méthode Object.unwatch), valueOf (méthode Object.valueOf), watch (méthode Object.watch)

# addListener (méthode IME.addListener)

```
public static addListener(listener:Object) : Void
```

Enregistre un objet pour qu'il reçoive une notification lorsqu'un gestionnaire d'événements IME est appelé par l'événement onIMEComposition.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

### Paramètres

*listener*:Object - Objet, auquel est associée une méthode onIMEComposition (readingString), qui écoute une notification de rappel à partir des gestionnaires d'événements IME. La chaîne de lecture transmise à cette méthode se trouve dans le mode de composition de l'IME. Par exemple, si l'utilisateur tape le texte Hiragana, puis sélectionne un candidat Kanji, la chaîne de lecture correspond à la valeur Hiragana d'origine.

### Exemple

L'exemple suivant décrit l'ajout d'un objet écouteur à System.IME qui émet une notification lorsqu'un utilisateur définit la chaîne de composition en cliquant dans le champ de texte.

```
var IMEListener:Object = new Object();
IMEListener.onIMEComposition = function(str:String) {
  trace(">> onIMEComposition: " + str);
}
System.IME.addListener(IMEListener);
trace(System.IME.length);

var mc:MovieClip = this.createEmptyMovieClip("mc",
  this.getNextHighestDepth());
mc.createTextField("txt", this.getNextHighestDepth(), 0, 0, 0, 0);
mc.txt.border = true;
mc.txt.background = true;
mc.txt.autoSize = "left";
mc.txt.text = "Click this text to add a listener.";

mc.onPress = function() {
  if(System.capabilities.hasIME) {
    Selection.setFocus(mc.txt);
    System.IME.setCompositionString(mc.txt.text);
  }
}
```

## ALPHANUMERIC_FULL (propriété IME.ALPHANUMERIC_FULL)

```
statique publique ALPHANUMERIC_FULL : String
```

Une chaîne ayant la valeur "ALPHANUMERIC_FULL" à utiliser avec setConversionMode() et getConversionMode(). Cette constante est utilisée sur tous les IME.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

### Exemple

L'exemple suivant définit l'IME sur `ALPHANUMERIC_FULL` si le système est doté d'un IME (Input Method Editor) (`System.capabilities.hasIME`).

```
if(System.capabilities.hasIME) {
  trace(System.IME.getConversionMode());

  System.IME.setConversionMode(System.IME.ALPHANUMERIC_FULL);
  trace(System.IME.getConversionMode());
}
```

### Voir également

`setConversionMode (méthode IME.setConversionMode)`, `getConversionMode (méthode IME.getConversionMode)`, `hasIME (propriété capabilities.hasIME)`

## ALPHANUMERIC_HALF (propriété IME.ALPHANUMERIC_HALF)

```
statique publique ALPHANUMERIC_HALF : String
```

Une chaîne ayant la valeur `"ALPHANUMERIC_HALF"` à utiliser avec `setConversionMode()` et `getConversionMode()`. Cette constante est utilisée sur tous les IME.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

### Exemple

L'exemple suivant définit l'IME sur `ALPHANUMERIC_HALF` si le système est doté d'un IME (Input Method Editor) (`System.capabilities.hasIME`).

```
if(System.capabilities.hasIME) {
  trace(System.IME.getConversionMode());

  System.IME.setConversionMode(System.IME.ALPHANUMERIC_HALF);
  trace(System.IME.getConversionMode());
}
```

### Voir également

`setConversionMode (méthode IME.setConversionMode)`, `getConversionMode (méthode IME.getConversionMode)`

# CHINESE (propriété IME.CHINESE)

`statique publique CHINESE : String`

Une chaîne ayant la valeur `"CHINESE"` à utiliser avec `setConversionMode()` et `getConversionMode()`. Cette constante est utilisée sur les IME chinois simplifié et traditionnel.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

## Exemple

L'exemple suivant définit l'IME sur `CHINESE` si le système est doté d'un IME (Input Method Editor) (`System.capabilities.hasIME`).

```
if(System.capabilities.hasIME) {
  trace(System.IME.getConversionMode());

  System.IME.setConversionMode(System.IME.CHINESE);
  trace(System.IME.getConversionMode());
}
```

## Voir également

`setConversionMode (méthode IME.setConversionMode)`, `getConversionMode (méthode IME.getConversionMode)`

# doConversion (méthode IME.doConversion)

`statique publique doConversion() : Boolean`

Demande à l'IME de sélectionner le premier candidat pour la chaîne de composition actuelle.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

## Renvoie

`Boolean` - Renvoie `true` si l'appel réussit ; sinon `false`.

## Exemple

L'exemple suivant illustre la méthode de sélection du premier candidat pour la chaîne de composition IME. Si le système utilisateur est équipé d'un IME, un clic dans le champ de texte permet de sélectionner le candidat.

```
var mc:MovieClip = this.createEmptyMovieClip("mc",
  this.getNextHighestDepth());
mc.createTextField("txt", this.getNextHighestDepth(), 0, 0, 0, 0);
mc.txt.border = true;
mc.txt.background = true;
```

```
mc.txt.autoSize = "left";
mc.txt.text = "Set this text as the composition string and convert it.";

mc.onPress = function() {
  if(System.capabilities.hasIME) {
    Selection.setFocus(mc.txt);
    System.IME.setCompositionString(mc.txt.text);
    trace(System.IME.doConversion());
  }
}
```

# getConversionMode (méthode IME.getConversionMode)

```
statique publique getConversionMode() : String
```

Indique le mode de conversion de l'IME actuel.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

## Renvoie

`String` - Le mode de conversion. Les valeurs possibles sont les constantes de chaîne de mode IME qui spécifient le mode de conversion :

- `ALPHANUMERIC_FULL`
- `ALPHANUMERIC_HALF`
- `CHINESE`
- `JAPANESE_HIRAGANA`
- `JAPANESE_KATAKANA_FULL`
- `JAPANESE_KATAKANA_HALF`
- `KOREAN`
- `UNKNOWN`

## Exemple

L'exemple suivant permet d'obtenir l'IME si le système est doté d'un IME (Input Method Editor) ( `System.capabilities.hasIME`).

```
var mode:String = System.IME.UNKNOWN;
if(System.capabilities.hasIME) {
  mode = System.IME.getConversionMode();
}
trace(mode);
```

### Voir également

ALPHANUMERIC_FULL (propriété IME.ALPHANUMERIC_FULL), ALPHANUMERIC_HALF (propriété IME.ALPHANUMERIC_HALF), CHINESE (propriété IME.CHINESE), JAPANESE_HIRAGANA (propriété IME.JAPANESE_HIRAGANA), JAPANESE_KATAKANA_FULL (propriété IME.JAPANESE_KATAKANA_FULL), JAPANESE_KATAKANA_HALF (propriété IME.JAPANESE_KATAKANA_HALF), KOREAN (propriété IME.KOREAN), UNKNOWN (propriété IME.UNKNOWN)

## getEnabled (méthode IME.getEnabled)

```
statique publique getEnabled() : Boolean
```

Indique si le système IME est activé. Un IME activé effectue une entrée multi-octets ; un IME désactivé effectue une entrée alphanumérique.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

### Renvoie

`Boolean` -Renvoie `true` si l'IME du système est activé ; `false` s'il est désactivé.

### Exemple

L'exemple suivant permet de vérifier que l'IME est activé en appelant la méthode `isEnabled()`.

```
if(System.capabilities.hasIME) {
  var isImeEnabled:Boolean = System.IME.getEnabled();
  trace(isImeEnabled);
}
```

## JAPANESE_HIRAGANA (propriété IME.JAPANESE_HIRAGANA)

```
statique publique JAPANESE_HIRAGANA : String
```

Une chaîne ayant la valeur `"JAPANESE_HIRAGANA"` à utiliser avec `setConversionMode()` et `getConversionMode()`. Cette constante est utilisée sur les IME japonais.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

### Exemple

L'exemple suivant définit l'IME sur `JAPANESE_HIRAGANA` si le système est doté d'un IME (Input Method Editor) (`System.capabilities.hasIME`).

```
if(System.capabilities.hasIME) {
```

```
  trace(System.IME.getConversionMode());

  System.IME.setConversionMode(System.IME.JAPANESE_HIRAGANA);
  trace(System.IME.getConversionMode());
}
```

### Voir également

setConversionMode (méthode IME.setConversionMode), getConversionMode (méthode IME.getConversionMode)

## JAPANESE_KATAKANA_FULL (propriété IME.JAPANESE_KATAKANA_FULL)

```
statique publique JAPANESE_KATAKANA_FULL : String
```

Une chaîne ayant la valeur "JAPANESE_KATAKANA_FULL" à utiliser avec setConversionMode() et getConversionMode(). Cette constante est utilisée sur les IME japonais.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

### Exemple

L'exemple suivant définit l'IME sur JAPANESE_KATAKANA_FULL si le système est doté d'un IME (Input Method Editor) (System.capabilities.hasIME).

```
if(System.capabilities.hasIME) {
  trace(System.IME.getConversionMode());

  System.IME.setConversionMode(System.IME.JAPANESE_KATAKANA_FULL);
  trace(System.IME.getConversionMode());
}
```

### Voir également

setConversionMode (méthode IME.setConversionMode), getConversionMode (méthode IME.getConversionMode)

## JAPANESE_KATAKANA_HALF (propriété IME.JAPANESE_KATAKANA_HALF)

```
statique publique JAPANESE_KATAKANA_HALF : String
```

Une chaîne ayant la valeur "JAPANESE_KATAKANA_HALF" à utiliser avec setConversionMode() et getConversionMode(). Cette constante est utilisée sur les IME japonais.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

### Exemple

L'exemple suivant définit l'IME sur `JAPANESE_KATAKANA_HALF` si le système est doté d'un IME (Input Method Editor) (`System.capabilities.hasIME`).

```
if(System.capabilities.hasIME) {
  trace(System.IME.getConversionMode());

  System.IME.setConversionMode(System.IME.JAPANESE_KATAKANA_HALF);
  trace(System.IME.getConversionMode());
}
```

### Voir également

`setConversionMode (méthode IME.setConversionMode)`, `getConversionMode (méthode IME.getConversionMode)`

## KOREAN (propriété IME.KOREAN)

```
statique publique KOREAN : String
```

Une chaîne ayant la valeur `"KOREAN"` à utiliser avec `setConversionMode()` et `getConversionMode()`. Cette constante est utilisée sur les IME coréens.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

### Exemple

L'exemple suivant définit l'IME sur `KOREAN` si le système est doté d'un IME (Input Method Editor) (`System.capabilities.hasIME`).

```
if(System.capabilities.hasIME) {
  trace(System.IME.getConversionMode());

  System.IME.setConversionMode(System.IME.KOREAN);
  trace(System.IME.getConversionMode());
}
```

### Voir également

`setConversionMode (méthode IME.setConversionMode)`, `getConversionMode (méthode IME.getConversionMode)`

# onIMEComposition (écouteur d'événements IME.onIMEComposition)

onIMEComposition = fonction([readingString:String]) {}

Notifié lorsque la chaîne de composition IME est en cours de définition. Pour utiliser cet écouteur, vous devez créer un objet écouteur. Vous pouvez ensuite définir une fonction pour cet écouteur et utiliser `IME.addListener()` pour enregistrer l'écouteur auprès de l'objet IME, comme indiqué dans le code suivant :

```
var someListener:Object = new Object();
someListener.onIMEComposition = function () {
  // statements
}
System.IME.addListener(someListener);
```

Les écouteurs permettent à divers blocs de code de coopérer car plusieurs écouteurs peuvent recevoir une notification sur un événement unique.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

## Paramètres

*`readingString`*`:String` [facultatif] - Texte d'origine tapé dans l'IME avant que l'utilisateur n'ait commencé à choisir des candidats.

## Exemple

L'exemple suivant illustre la méthode d'ajout d'un objet écouteur à l'aide de la méthode de rappel `onIMEComposition()` to `System.IME`, qui envoie une notification lorsqu'un utilisateur définit la chaîne de composition en cliquant dans le champ de texte.

```
var IMEListener:Object = new Object();
IMEListener.onIMEComposition = function(str:String) {
  trace(">> onIMEComposition: " + str);
}
System.IME.addListener(IMEListener);
trace(System.IME.length);

var mc:MovieClip = this.createEmptyMovieClip("mc",
  this.getNextHighestDepth());
mc.createTextField("txt", this.getNextHighestDepth(), 0, 0, 0, 0);
mc.txt.border = true;
mc.txt.background = true;
mc.txt.autoSize = "left";
mc.txt.text = "Click this text to add a listener.";

mc.onPress = function() {
  if(System.capabilities.hasIME) {
```

```
    Selection.setFocus(mc.txt);
    System.IME.setCompositionString(mc.txt.text);
  }
}
```

### Voir également

addListener (méthode IME.addListener), setCompositionString (méthode IME.setCompositionString)

## removeListener (méthode IME.removeListener)

statique publique removeListener(listener:Object) : Boolean

Supprime un objet écouteur qui était auparavant enregistré dans une occurrence IME auprès de IME.addListener().

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

### Paramètres

*listener*:Object - L'objet qui ne recevra plus de notification de rappel émanant des gestionnaires d'événements IME.

### Renvoie

Boolean - Renvoie true si l'objet écouteur est supprimé, sinon false.

### Exemple

L'exemple suivant indique comment supprimer un objet écouteur de System.IME lorsqu'un utilisateur définit la chaîne de composition en cliquant dans le champ de texte.

```
var IMEListener:Object = new Object();
IMEListener.onIMEComposition = function(str:String) {
  trace(">> onIMEComposition: " + str);

  System.IME.removeListener(this);
  trace(System.IME.length); // 0
}
System.IME.addListener(IMEListener);
trace(System.IME.length); // 1

var mc:MovieClip = this.createEmptyMovieClip("mc",
  this.getNextHighestDepth());
mc.createTextField("txt", this.getNextHighestDepth(), 0, 0, 0, 0);
mc.txt.border = true;
mc.txt.background = true;
mc.txt.autoSize = "left";
mc.txt.text = "Click this text to add and remove a listener.";
```

```
mc.onPress = function() {
  if(System.capabilities.hasIME) {
    Selection.setFocus(mc.txt);
    System.IME.setCompositionString(mc.txt.text);
  }
}
```

# setCompositionString (méthode IME.setCompositionString)

```
statique publique setCompositionString(composition:String) : Boolean
```

Définit la chaîne de composition IME. Lorsque cette chaîne est définie, l'utilisateur peut sélectionner des candidats IME avant d'enregistrer le résultat dans le champ de texte ayant actuellement le focus.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

## Paramètres

*composition*:String - La chaîne à envoyer à l'IME.

## Renvoie

Boolean- Si la chaîne de composition IME est définie avec succès, renvoie `true`. Cette méthode échoue et renvoie `false` si aucun champ de texte n'a le focus.

## Exemple

L'exemple suivant illustre la méthode de définition de la chaîne de composition IME. Si le système utilisateur est équipé d'un IME, un clic dans le champ de texte permet d'afficher les options IME.

```
var mc:MovieClip = this.createEmptyMovieClip("mc",
  this.getNextHighestDepth());
mc.createTextField("txt", this.getNextHighestDepth(), 0, 0, 0, 0);
mc.txt.border = true;
mc.txt.background = true;
mc.txt.autoSize = "left";
mc.txt.text = "Set this text as the composition string.";

mc.onPress = function() {
  if(System.capabilities.hasIME) {
    Selection.setFocus(mc.txt);
    trace(System.IME.setCompositionString(mc.txt.text));
  }
}
```

# setConversionMode (méthode IME.setConversionMode)

`statique publique setConversionMode(mode:String) : Boolean`

Définit le mode de conversion de l'IME actuel.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

## Paramètres

*mode*:`String` - Le mode de conversion. Les valeurs possibles sont les constantes de chaîne de mode IME :

- `ALPHANUMERIC_FULL`
- `ALPHANUMERIC_HALF`
- `CHINESE`
- `JAPANESE_HIRAGANA`
- `JAPANESE_KATAKANA_FULL`
- `JAPANESE_KATAKANA_HALF`
- `KOREAN`

## Renvoie

`Boolean` - Renvoie `true` si le mode de conversion a été correctement défini ; sinon `false`.

## Exemple

L'exemple suivant permet d'obtenir l'IME si le système est doté d'un IME (Input Method Editor) ( `System.capabilities.hasIME`) et définit la variable `mode` sur cette valeur.

```
var mode:String = System.IME.UNKNOWN;
if(System.capabilities.hasIME) {
  mode = System.IME.getConversionMode();
}
System.IME.setConversionMode(mode);
trace(System.IME.getConversionMode());
```

## Voir également

`ALPHANUMERIC_FULL (propriété IME.ALPHANUMERIC_FULL)`, `ALPHANUMERIC_HALF (propriété IME.ALPHANUMERIC_HALF)`, `CHINESE (propriété IME.CHINESE)`, `JAPANESE_HIRAGANA (propriété IME.JAPANESE_HIRAGANA)`, `JAPANESE_KATAKANA_FULL (propriété IME.JAPANESE_KATAKANA_FULL)`, `JAPANESE_KATAKANA_HALF (propriété IME.JAPANESE_KATAKANA_HALF)`, `KOREAN (propriété IME.KOREAN)`

# setEnabled (méthode IME.setEnabled)

```
statique publique setEnabled(enabled:Boolean) : Boolean
```

Active ou désactive le système IME. Un IME activé effectue une entrée multi-octets ; un IME désactivé effectue une entrée alphanumérique.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

## Paramètres

*enabled*:Boolean - Définir sur `true` pour activer l'IME du système, sur `false` pour le désactiver.

## Renvoie

`Boolean` - Si la tentative d'activation du système IME réussit, renvoie `true` ; sinon `false`.

## Exemple

L'exemple suivant permet de vérifier que l'IME est activé en appelant la méthode `isEnabled()` puis remplace son état activé par son état opposé en appelant la méthode `setEnabled()`.

```
if(System.capabilities.hasIME) {
  var isImeEnabled:Boolean = System.IME.getEnabled();
  trace(isImeEnabled);

  if(isImeEnabled) {
    System.IME.setEnabled(false);
  }
  else {
    System.IME.setEnabled(true);
  }

  var isImeEnabled:Boolean = System.IME.getEnabled();
  trace(isImeEnabled);
}
```

# UNKNOWN (propriété IME.UNKNOWN)

```
statique publique UNKNOWN : String
```

Une chaîne ayant la valeur `"UNKNOWN"` à utiliser avec `getConversionMode()`. Cette constante est utilisée sur tous les IME.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

### Exemple

L'exemple suivant définit l'IME sur `UNKNOWN` si le système est doté d'un IME (Input Method Editor) (`System.capabilities.hasIME`).

```
if(System.capabilities.hasIME) {
  trace(System.IME.getConversionMode());

  System.IME.setConversionMode(System.IME.UNKNOWN);
  trace(System.IME.getConversionMode());
}
```

### Voir également

`getConversionMode (méthode IME.getConversionMode)`

# Key

```
Object
  |
  +-Key
```

`public class` ***Key***
`extends Object`

La classe Key est une classe de niveau supérieur dont vous pouvez utiliser les méthodes et les propriétés sans l'aide d'un constructeur. Utilisez les méthodes de la classe Key pour créer une interface pouvant être contrôlée par un utilisateur disposant d'un clavier standard. Les propriétés de la classe Key sont des constantes représentant les touches le plus fréquemment utilisées pour commander les applications telles que les touches de direction, Pg. Préc et Pg. Suiv.

Une application Flash ne peut contrôler que les événements de clavier qui se produisent dans son focus. Une application Flash ne peut pas détecter les événements de clavier dans une autre application.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

### Résumé des propriétés

| Modificateurs | Propriété | Description |
|---|---|---|
| `static` | `BACKSPACE:Number` | La valeur de code correspondant à la touche Retour arrière (8). |
| `static` | `CAPSLOCK:Number` | La valeur de code correspondant à la touche Verr Maj (20). |
| `static` | `CONTROL:Number` | La valeur de code correspondant à la touche Ctrl (17). |

| Modificateurs | Propriété | Description |
|---|---|---|
| static | DELETEKEY:Number | La valeur de code correspondant à la touche Suppr (46). |
| static | DOWN:Number | La valeur de code correspondant à la flèche Bas (40). |
| static | END:Number | La valeur de code correspondant à la touche Fin (35). |
| static | ENTER:Number | La valeur de code correspondant à la touche Entrée (13). |
| static | ESCAPE:Number | La valeur de code correspondant à la touche Echap (27). |
| static | HOME:Number | La valeur de code correspondant à la touche Origine (36). |
| static | INSERT:Number | La valeur de code correspondant à la touche Inser (45). |
| static | LEFT:Number | La valeur de code correspondant à la flèche Gauche (37). |
| static | _listeners:Array [lecture seule] | Une liste de références à tous les objets écouteurs enregistrés auprès de l'objet Key. |
| static | PGDN:Number | La valeur de code correspondant à la touche Pg. Suiv. (34). |
| static | PGUP:Number | La valeur de code correspondant à la touche Pg. Préc. (33). |
| static | RIGHT:Number | La valeur de code correspondant à la flèche Droite (39). |
| static | SHIFT:Number | La valeur de code correspondant à la touche Maj (16). |
| static | SPACE:Number | La valeur de code correspondant à la barre d'espace (32). |
| static | TAB:Number | La valeur de code correspondant à la touche Tab (9). |
| static | UP:Number | La valeur de code correspondant à la flèche Haut (38). |

*Propriétés héritées de la classe Object*

```
constructeur (propriété Object.constructor), __proto__ (Object.__proto__
property), prototype (Object.prototype, propriété), __resolve
(Object.__resolve, propriété)
```

## Résumé des événements

| Evénement | Description |
|---|---|
| `onKeyDown = function() {}` | Notifié lorsqu'une touche est enfoncée. |
| `onKeyUp = function() {}` | Notifié lorsqu'une touche est relâchée. |

## Résumé de la méthode

| Modificateurs | Signature | Description |
|---|---|---|
| `static` | `addListener(listener:Object) : Void` | Enregistre un objet pour qu'il reçoive la notification `onKeyDown` et `onKeyUp`. |
| `static` | `getAscii() : Number` | Renvoie le code ASCII de la dernière touche enfoncée ou relâchée. |
| `static` | `getCode() : Number` | Renvoie la valeur de code de touche de la dernière touche enfoncée. |
| `static` | `isAccessible() : Boolean` | Renvoie une valeur booléenne indiquant, en fonction des restrictions de sécurité, si la dernière touche enfoncée peut être accessible aux autres fichiers SWF. |
| `static` | `isDown(code:Number) : Boolean` | Renvoie `true` si la touche spécifiée dans *`keycode`* est enfoncée ; `false` sinon. |
| `static` | `isToggled(code:Number) : Boolean` | Renvoie `true` si la touche Verr Maj ou Verr num est activée (activation ou désactivation de l'état actif) ; `false` sinon. |
| `static` | `removeListener(listener:Object) : Boolean` | Supprime un objet précédemment enregistré auprès de `Key.addListener()`. |

*Méthodes héritées de la classe Object*

`addProperty (méthode Object.addProperty)`, `hasOwnProperty (méthode Object.hasOwnProperty)`, `isPropertyEnumerable (méthode Object.isPropertyEnumerable)`, `isPrototypeOf (méthode Object.isPrototypeOf)`, `registerClass (méthode Object.registerClass)`, `toString (méthode Object.toString)`, `unwatch (méthode Object.unwatch)`, `valueOf (méthode Object.valueOf)`, `watch (méthode Object.watch)`

# addListener (méthode Key.addListener)

`statique publique addListener(listener:Object) : Void`

Enregistre un objet pour qu'il reçoive la notification `onKeyDown` et `onKeyUp`. Lorsqu'une touche est enfoncée ou relâchée, quel que soit le focus d'entrée, la méthode `onKeyDown`ou `onKeyUp` de tous les objets d'écoute enregistrés avec `addListener()`est appelée. Plusieurs objets peuvent écouter les notifications de clavier. Si l'écouteur *`newListener`* est déjà enregistré, aucun changement ne se produit.

Une application Flash ne peut contrôler que les événements de clavier qui se produisent dans son focus. Une application Flash ne peut pas détecter les événements de clavier dans une autre application.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

## Paramètres

*`listener`*`:Object` - Un objet avec des méthodes `onKeyDown` et `onKeyUp`.

## Exemple

L'exemple suivant crée un nouvel objet écouteur et définit une fonction pour la méthode `onKeyDown`et `onKeyUp`. La dernière ligne utilise la méthode `addListener()` pour enregistrer l'écouteur auprès de l'objet Key afin qu'il puisse recevoir des notifications émanant des événements d'abaissement et de relâchement de touche.

```
var myListener:Object = new Object();
myListener.onKeyDown = function () {
  trace ("You pressed a key.");
}
myListener.onKeyUp = function () {
  trace ("You released a key.");
}
Key.addListener(myListener);
```

L'exemple suivant affecte le raccourci clavier Ctrl+7 à un bouton dont le nom d'occurrence est `my_btn`et rend les informations sur le raccourci clavier disponibles pour les logiciels de lecture d'écran (consultez `_accProps`). Dans cet exemple, lorsque vous appuyez sur Ctrl+7, la fonction `myOnPress` affiche le texte `hello` dans le panneau de sortie.

```
function myOnPress() {
  trace("hello");
}
function myOnKeyDown() {
  // 55 is key code for 7
  if (Key.isDown(Key.CONTROL) && Key.getCode() == 55) {
  Selection.setFocus(my_btn);
  my_btn.onPress();
```

```
  }
}
var myListener:Object = new Object();
myListener.onKeyDown = myOnKeyDown;
Key.addListener(myListener);
my_btn.onPress = myOnPress;
my_btn._accProps.shortcut = "Ctrl+7";
Accessibility.updateProperties();
```

### Voir également

getCode (méthode Key.getCode), isDown (méthode Key.isDown), onKeyDown (écouteur d'événement Key.onKeyDown), onKeyUp (écouteur d'événement Key.onKeyUp), removeListener (méthode Key.removeListener)

## BACKSPACE (propriété Key.BACKSPACE)

```
statique publique BACKSPACE : Number
```

La valeur de code correspondant à la touche Retour arrière (8).

**Disponibilité :** ActionScript 1.0 ; Flash Player 5

### Exemple

L'exemple suivant crée un nouvel objet écouteur et définit une fonction pour `onKeyDown`. La dernière ligne utilise la méthode `addListener()` pour enregistrer l'écouteur auprès de l'objet Key afin qu'il puisse recevoir des notifications émanant de l'événement de touche enfoncée.

```
var keyListener:Object = new Object();
keyListener.onKeyDown = function() {
  if (Key.isDown(Key.BACKSPACE)) {
  trace("you pressed the Backspace key.");
  } else {
  trace("you DIDN'T press the Backspace key.");
  }
};
Key.addListener(keyListener);
```

Lorsque vous utilisez cet exemple, assurez-vous de sélectionner Contrôle > Désactiver les raccourcis clavier dans l'environnement de test.

## CAPSLOCK (propriété Key.CAPSLOCK)

```
statique publique CAPSLOCK : Number
```

La valeur de code correspondant à la touche Verr Maj (20).

**Disponibilité :** ActionScript 1.0 ; Flash Player 5

### Exemple

L'exemple suivant crée un nouvel objet écouteur et définit une fonction pour `onKeyDown`. La dernière ligne utilise la méthode `addListener()` pour enregistrer l'écouteur auprès de l'objet Key afin qu'il puisse recevoir des notifications émanant de l'événement de touche enfoncée.

```
var keyListener:Object = new Object();
keyListener.onKeyDown = function() {
  if (Key.isDown(Key.CAPSLOCK)) {
  trace("you pressed the Caps Lock key.");
  trace("\tCaps Lock == "+Key.isToggled(Key.CAPSLOCK));
  }
};
Key.addListener(keyListener);
```

Les informations s'affichent dans le panneau de sortie lorsque vous appuyez sur la touche Verr Maj. Le panneau de sortie affiche la valeur `true` ou `false` selon l'activation, ou non, de la touche Verr Maj à l'aide de la méthode isToggled.

## CONTROL (propriété Key.CONTROL)

```
statique publique CONTROL : Number
```

La valeur de code correspondant à la touche Ctrl (17).

**Disponibilité :** ActionScript 1.0 ; Flash Player 5

### Exemple

L'exemple suivant affecte le raccourci clavier Ctrl+7 à un bouton dont le nom d'occurrence est `my_btn`et rend les informations sur le raccourci clavier disponibles pour les logiciels de lecture d'écran (consultez `_accProps`). Dans cet exemple, lorsque vous appuyez sur Ctrl+7, la fonction `myOnPress` affiche le texte `hello` dans le panneau de sortie.

```
function myOnPress() {
  trace("hello");
}
function myOnKeyDown() {
  // 55 is key code for 7
  if (Key.isDown(Key.CONTROL) && Key.getCode() == 55) {
  Selection.setFocus(my_btn);
  my_btn.onPress();
  }
}
var myListener:Object = new Object();
myListener.onKeyDown = myOnKeyDown;
Key.addListener(myListener);
my_btn.onPress = myOnPress;
my_btn._accProps.shortcut = "Ctrl+7";
Accessibility.updateProperties();
```

# DELETEKEY (propriété Key.DELETEKEY)

```
statique publique DELETEKEY : Number
```

La valeur de code correspondant à la touche Suppr (46).

**Disponibilité :** ActionScript 1.0 ; Flash Player 5

## Exemple

L'exemple suivant vous permet de tracer des lignes avec le pointeur de la souris par le biais de l'API de dessin et des objets écouteurs. Appuyez sur la touche Retour arrière ou Suppr pour supprimer les lignes tracées.

```
this.createEmptyMovieClip("canvas_mc", this.getNextHighestDepth());
var mouseListener:Object = new Object();
mouseListener.onMouseDown = function() {
  this.drawing = true;
  canvas_mc.moveTo(_xmouse, _ymouse);
  canvas_mc.lineStyle(3, 0x99CC00, 100);
};
mouseListener.onMouseUp = function() {
  this.drawing = false;
};
mouseListener.onMouseMove = function() {
  if (this.drawing) {
  canvas_mc.lineTo(_xmouse, _ymouse);
  }
  updateAfterEvent();
};
Mouse.addListener(mouseListener);
//
var keyListener:Object = new Object();
keyListener.onKeyDown = function() {
  if (Key.isDown(Key.DELETEKEY) || Key.isDown(Key.BACKSPACE)) {
  canvas_mc.clear();
  }
};
Key.addListener(keyListener);
```

Lorsque vous utilisez cet exemple, assurez-vous de sélectionner Contrôle > Désactiver les raccourcis clavier dans l'environnement de test.

La méthode `MovieClip.getNextHighestDepth()` utilisée dans cet exemple nécessite Flash Player 7 ou une version ultérieure. Si votre fichier SWF inclut un composant de la version 2, utilisez la classe DepthManager des composants de la version 2 à la place de la méthode `MovieClip.getNextHighestDepth()`.

# DOWN (propriété Key.DOWN)

```
statique publique DOWN : Number
```

La valeur de code correspondant à la flèche Bas (40).

**Disponibilité :** ActionScript 1.0 ; Flash Player 5

## Exemple

L'exemple suivant déplace un clip intitulé `car_mc` selon une distance constante (10) lorsque vous appuyez sur les touches fléchées. Un son est lu lorsque vous appuyez sur la barre d'espace. Pour cet exemple, affectez l'identificateur de liaison`horn_id` à un son figurant dans la bibliothèque.

```
var DISTANCE:Number = 10;
var horn_sound:Sound = new Sound();
horn_sound.attachSound("horn_id");
var keyListener_obj:Object = new Object();
keyListener_obj.onKeyDown = function() {
  switch (Key.getCode()) {
  case Key.SPACE :
  horn_sound.start();
  break;
  case Key.LEFT :
  car_mc._x -= DISTANCE;
  break;
  case Key.UP :
  car_mc._y -= DISTANCE;
  break;
  case Key.RIGHT :
  car_mc._x += DISTANCE;
  break;
  case Key.DOWN :
  car_mc._y += DISTANCE;
  break;
  }
};
Key.addListener(keyListener_obj);
```

# END (propriété Key.END)

```
statique publique END : Number
```

La valeur de code correspondant à la touche Fin (35).

**Disponibilité :** ActionScript 1.0 ; Flash Player 5

# ENTER (propriété Key.ENTER)

`statique publique ENTER : Number`

La valeur de code correspondant à la touche Entrée (13).

**Disponibilité :** ActionScript 1.0 ; Flash Player 5

## Exemple

L'exemple suivant déplace un clip intitulé `car_mc` selon une distance constante (10) lorsque vous appuyez sur les touches fléchées. L'occurrence `car_mc`s'arrête lorsque vous appuyez sur Entrée et supprimez l'événement `onEnterFrame`.

```
var DISTANCE:Number = 5;
var keyListener:Object = new Object();
keyListener.onKeyDown = function() {
  switch (Key.getCode()) {
  case Key.LEFT :
  car_mc.onEnterFrame = function() {
    this._x -= DISTANCE;
  };
  break;
  case Key.UP :
  car_mc.onEnterFrame = function() {
    this._y -= DISTANCE;
  };
  break;
  case Key.RIGHT :
  car_mc.onEnterFrame = function() {
    this._x += DISTANCE;
  };
  break;
  case Key.DOWN :
  car_mc.onEnterFrame = function() {
    this._y += DISTANCE;
  };
  break;
  case Key.ENTER :
  delete car_mc.onEnterFrame;
  break;
  }
};
Key.addListener(keyListener);
```

Lorsque vous utilisez cet exemple, assurez-vous de sélectionner Contrôle > Désactiver les raccourcis clavier dans l'environnement de test.

# ESCAPE (propriété Key.ESCAPE)

```
statique publique ESCAPE : Number
```

La valeur de code correspondant à la touche Echap (27).

**Disponibilité :** ActionScript 1.0 ; Flash Player 5

## Exemple

L'exemple suivant définit un compteur. Lorsque vous appuyez sur Echap, le panneau de sortie affiche des informations incluant le temps qu'il vous a fallu pour appuyer sur la touche.

```
var keyListener:Object = new Object();
keyListener.onKeyDown = function() {
  if (Key.isDown(Key.ESCAPE)) {
  // Get the current timer, convert the value to seconds and round it to two
  decimal places.
  var timer:Number = Math.round(getTimer()/10)/100;
  trace("you pressed the Esc key: "+getTimer()+" ms ("+timer+" s)");
  }
};
Key.addListener(keyListener);
```

Lorsque vous utilisez cet exemple, assurez-vous de sélectionner Contrôle > Désactiver les raccourcis clavier dans l'environnement de test.

# getAscii (méthode Key.getAscii)

```
statique publique getAscii() : Number
```

Renvoie le code ASCII de la dernière touche enfoncée ou relâchée. Les valeurs ASCII renvoyées sont des valeurs de clavier anglais. Par exemple, si vous appuyez sur Maj+2, `Key.getAscii()` renvoie @ sur un clavier japonais, ce qui correspond à ce qu'il renvoie sur un clavier anglais.

Une application Flash ne peut contrôler que les événements de clavier qui se produisent dans son focus. Une application Flash ne peut pas détecter les événements de clavier dans une autre application.

**Disponibilité :** ActionScript 1.0 ; Flash Player 5

## Renvoie

`Number`- La valeur ASCII de la dernière touche enfoncée. Cette méthode renvoie 0 si aucune touche n'a été enfoncée ou relâchée ou si le code n'est pas accessible pour des raisons de sécurité.

### Exemple

L'exemple suivant appelle la méthode `getAscii()` à chaque fois que l'utilisateur appuie sur une touche. Cet exemple crée un objet écouteur intitulé `keyListener` et définit une fonction qui répond à l'événement `onKeyDown` en appelant `Key.getAscii()`. L'objet `keyListener` est ensuite enregistré auprès de l'objet `Key`, qui envoie le message `onKeyDown` à chaque fois que l'utilisateur appuie sur une touche lors de la lecture du fichier SWF.

```
var keyListener:Object = new Object();
keyListener.onKeyDown = function() {
  trace("The ASCII code for the last key typed is: "+Key.getAscii());
};
Key.addListener(keyListener);
```

Lorsque vous utilisez cet exemple, assurez-vous de sélectionner Contrôle > Désactiver les raccourcis clavier dans l'environnement de test.

L'exemple suivant ajoute un appel de la méthode `Key.getAscii()` pour illustrer la façon dont les deux méthodes diffèrent. La principale différence réside dans le fait que `Key.getAscii()` fait la distinction entre les minuscules et les majuscules, contrairement à `Key.getCode()`.

```
var keyListener:Object = new Object();
keyListener.onKeyDown = function() {
  trace("For the last key typed:");
  trace("\tThe Key code is: "+Key.getCode());
  trace("\tThe ASCII value is: "+Key.getAscii());
  trace("");
};
Key.addListener(keyListener);
```

Lorsque vous utilisez cet exemple, assurez-vous de sélectionner Contrôle > Désactiver les raccourcis clavier dans l'environnement de test.

### Voir également

`isAccessible (méthode Key.isAccessible)`

## getCode (méthode Key.getCode)

```
statique publique getCode() : Number
```

Renvoie la valeur de code de touche de la dernière touche enfoncée.

Remarque : L'implémentation de Flash Lite de cette méthode renvoie une chaîne ou un nombre, en fonction du code transmis par la plate-forme. Les seuls codes valides sont les codes standard acceptés par cette classe et les codes spéciaux répertoriés comme propriétés de la classe `ExtendedKey`.

Une application Flash ne peut contrôler que les événements de clavier qui se produisent dans son focus. Une application Flash ne peut pas détecter les événements de clavier dans une autre application.

**Disponibilité :** ActionScript 1.0 ; Flash Player 5

## Renvoie

`Number` - Le code de la dernière touche enfoncée. Cette méthode renvoie 0 si aucune touche n'a été enfoncée ou relâchée ou si le code n'est pas accessible pour des raisons de sécurité.

## Exemple

L'exemple suivant appelle la méthode `getCode()` à chaque fois que l'utilisateur appuie sur une touche. Cet exemple crée un objet écouteur intitulé `keyListener` et définit une fonction qui répond à l'événement `onKeyDown` en appelant `Key.getCode()`. L'objet `keyListener` est ensuite enregistré auprès de l'objet `Key`, qui envoie le message `onKeyDown` à chaque fois que l'utilisateur appuie sur une touche lors de la lecture du fichier SWF.

```
var keyListener:Object = new Object();
keyListener.onKeyDown = function() {
  // compare return value of getCode() to constant
  if (Key.getCode() == Key.ENTER) {
    trace ("Virtual key code: "+Key.getCode()+" (ENTER key)");
  }
  else {
    trace("Virtual key code: "+Key.getCode());
  }
};
Key.addListener(keyListener);
```

Lorsque vous utilisez cet exemple, assurez-vous de sélectionner Contrôle > Désactiver les raccourcis clavier dans l'environnement de test.

L'exemple suivant ajoute un appel de la méthode `Key.getAscii()` pour illustrer la façon dont les deux méthodes diffèrent. La principale différence réside dans le fait que la méthode `Key.getAscii()` fait la distinction entre les minuscules et les majuscules, contrairement à `Key.getCode()`.

```
var keyListener:Object = new Object();
keyListener.onKeyDown = function() {
  trace("For the last key typed:");
  trace("\tThe Key code is: "+Key.getCode());
  trace("\tThe ASCII value is: "+Key.getAscii());
  trace("");
};
Key.addListener(keyListener);
```

Lorsque vous utilisez cet exemple, assurez-vous de sélectionner Contrôle > Désactiver les raccourcis clavier dans l'environnement de test.

### Voir également

`getAscii (méthode Key.getAscii)`, `isAccessible (méthode Key.isAccessible)`

## HOME (propriété Key.HOME)

```
statique publique HOME : Number
```

La valeur de code correspondant à la touche Origine (36).

**Disponibilité :** ActionScript 1.0 ; Flash Player 5

### Exemple

L'exemple suivant associe une occurrence déplaçable de clip intitulée `car_mc` aux coordonnées *x* et *y* de 0,0. Lorsque vous appuyez sur la touche Origine, `car_mc` renvoie 0,0. Créez un clip ayant un ID de liaison `car_id`, puis ajoutez le code ActionScript suivant sur l'image 1 du scénario :

```
this.attachMovie("car_id", "car_mc", this.getNextHighestDepth(), {_x:0,
  _y:0});
car_mc.onPress = function() {
  this.startDrag();
};
car_mc.onRelease = function() {
  this.stopDrag();
};
var keyListener:Object = new Object();
keyListener.onKeyDown = function() {
  if (Key.isDown(Key.HOME)) {
  car_mc._x = 0;
  car_mc._y = 0;
  }
};
Key.addListener(keyListener);
```

La méthode `MovieClip.getNextHighestDepth()` utilisée dans cet exemple nécessite Flash Player 7 ou une version ultérieure. Si votre fichier SWF inclut un composant de la version 2, utilisez la classe DepthManager des composants de la version 2 à la place de la méthode `MovieClip.getNextHighestDepth()`.

# INSERT (propriété Key.INSERT)

```
statique publique INSERT : Number
```

La valeur de code correspondant à la touche Inser (45).

**Disponibilité :** ActionScript 1.0 ; Flash Player 5

## Exemple

L'exemple suivant crée un nouvel objet écouteur et définit une fonction pour `onKeyDown`. La dernière ligne utilise la méthode `addListener()` pour enregistrer l'écouteur auprès de l'objet Key afin qu'il puisse recevoir des notifications émanant de l'événement de touche enfoncée et afficher des informations dans le panneau de sortie.

```
var keyListener:Object = new Object();
keyListener.onKeyDown = function() {
  if (Key.isDown(Key.INSERT)) {
  trace("You pressed the Insert key.");
  }
};
Key.addListener(keyListener);
```

# isAccessible (méthode Key.isAccessible)

```
statique publique isAccessible() : Boolean
```

Renvoie une valeur booléenne indiquant, en fonction des restrictions de sécurité, si la dernière touche enfoncée peut être accessible aux autres fichiers SWF. Le code par défaut d'un fichier SWF dans un domaine ne permet pas nécessairement d'accéder à une séquence de touches générée à partir d'un fichier SWF dans un autre domaine. Pour plus d'informations sur la sécurité interdomaines, consultez le chapitre « Fonctionnement de la sécurité » dans le guide *Formation à ActionScript 2.0 dans Flash*.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

## Renvoie

`Boolean` - La valeur `true` si la dernière touche enfoncée est accessible. Si l'acccès n'est pas autorisé, cette méthode renvoie `false`.

# isDown (méthode Key.isDown)

```
statique publique isDown(code:Number) : Boolean
```

Renvoie `true` si la touche spécifiée dans *`keycode`* est enfoncée ; `false` sinon. Sous Macintosh, les valeurs de code correspondant aux touches Verr Maj et Verr num sont identiques.

Une application Flash ne peut contrôler que les événements de clavier qui se produisent dans son focus. Une application Flash ne peut pas détecter les événements de clavier dans une autre application.

**Disponibilité :** ActionScript 1.0 ; Flash Player 5

### Paramètres

*code*:Number - La valeur de code de touche affectée à une touche spécifique ou une propriété de classe Key associée à une touche spécifique.

### Renvoie

Boolean - La valeur true si la touche spécifiée dans *keycode* est enfoncée ; false sinon.

### Exemple

Le script suivant permet à l'utilisateur de contrôler l'emplacement d'un clip (car_mc) :

```
car_mc.onEnterFrame = function() {
 if (Key.isDown(Key.RIGHT)) {
 this._x += 10;
 } else if (Key.isDown(Key.LEFT)) {
 this._x -= 10;
 }
};
```

## isToggled (méthode Key.isToggled)

```
statique publique isToggled(code:Number) : Boolean
```

Renvoie true si la touche Verr Maj ou Verr num est activée (activation ou désactivation de l'état actif) ; false sinon. Bien que le terme *basculé* signifie généralement un basculement entre deux options, la méthode Key.isToggled() renvoie true uniquement si la touche est définie sur un état actif. Sous Macintosh, les valeurs de code correspondant aux touches Verr Maj et Verr num sont identiques.

Une application Flash ne peut contrôler que les événements de clavier qui se produisent dans son focus. Une application Flash ne peut pas détecter les événements de clavier dans une autre application.

**Disponibilité :** ActionScript 1.0 ; Flash Player 5

### Paramètres

*code*:Number - Le code de touche de la touche Verr Maj (20) ou de la touche Verr num (144).

## Renvoie

`Boolean` - La valeur `true` si la touche Verr Maj ou Verr num est activée (activation ou désactivation de l'état actif) ; `false` sinon.

## Exemple

L'exemple suivant appelle la méthode `isToggled()` à chaque fois que l'utilisateur appuie sur une touche et exécute une instruction trace à chaque fois que la touche Verr Maj est activée. Cet exemple crée un objet écouteur intitulé `keyListener` et définit une fonction qui répond à l'événement `onKeyDown` en appelant `Key.isToggled()`. L'objet `keyListener` est ensuite enregistré auprès de l'objet `Key`, qui envoie le message `onKeyDown` à chaque fois que l'utilisateur appuie sur une touche lors de la lecture du fichier SWF.

```
var keyListener:Object = new Object();
keyListener.onKeyDown = function() {
  if (Key.isDown(Key.CAPSLOCK)) {
  trace("you pressed the Caps Lock key.");
  trace("\tCaps Lock == "+Key.isToggled(Key.CAPSLOCK));
  }
};
Key.addListener(keyListener);
```

Les informations s'affichent dans le panneau de sortie lorsque vous appuyez sur la touche Verr Maj. Le panneau de sortie affiche la valeur `true` ou `false` selon l'activation, ou non, de la touche Verr Maj à l'aide de la méthode isToggled.

L'exemple suivant crée deux champs de texte mis à jour lorsque les touches Verr Maj et Verr num sont activées. Chaque champ de texte affiche true lorsque la touche est activée et false lorsqu'elle est désactivée.

```
this.createTextField("capsLock_txt", this.getNextHighestDepth(), 0, 0, 100,
  22);
capsLock_txt.autoSize = true;
capsLock_txt.html = true;
this.createTextField("numLock_txt", this.getNextHighestDepth(), 0, 22, 100,
  22);
numLock_txt.autoSize = true;
numLock_txt.html = true;
//
var keyListener:Object = new Object();
keyListener.onKeyDown = function() {
  capsLock_txt.htmlText = "<b>Caps Lock:</b> "+Key.isToggled(Key.CAPSLOCK);
  numLock_txt.htmlText = "<b>Num Lock:</b> "+Key.isToggled(144);
};
Key.addListener(keyListener);
```

La méthode `MovieClip.getNextHighestDepth()` utilisée dans cet exemple nécessite Flash Player 7 ou une version ultérieure. Si votre fichier SWF inclut un composant de la version 2, utilisez la classe DepthManager des composants de la version 2 à la place de la méthode `MovieClip.getNextHighestDepth()`.

# LEFT (propriété Key.LEFT)

```
statique publique LEFT : Number
```

La valeur de code correspondant à la flèche Gauche (37).

**Disponibilité :** ActionScript 1.0 ; Flash Player 5

## Exemple

L'exemple suivant déplace un clip intitulé `car_mc` selon une distance constante (10) lorsque vous appuyez sur les touches fléchées. Un son est lu lorsque vous appuyez sur la barre d'espace. Pour cet exemple, affectez l'identificateur de liaison`horn_id` à un son figurant dans la bibliothèque.

```
var DISTANCE:Number = 10;
var horn_sound:Sound = new Sound();
horn_sound.attachSound("horn_id");
var keyListener_obj:Object = new Object();
keyListener_obj.onKeyDown = function() {
  switch (Key.getCode()) {
  case Key.SPACE :
  horn_sound.start();
  break;
  case Key.LEFT :
  car_mc._x -= DISTANCE;
  break;
  case Key.UP :
  car_mc._y -= DISTANCE;
  break;
  case Key.RIGHT :
  car_mc._x += DISTANCE;
  break;
  case Key.DOWN :
  car_mc._y += DISTANCE;
  break;
  }
};
Key.addListener(keyListener_obj);
```

# _listeners (propriété Key._listeners)

`statique publique _listeners : Array [lecture seule]`

Une liste de références à tous les objets écouteurs enregistrés auprès de l'objet Key. Cette propriété est réservée à un usage interne uniquement mais peut être utile si vous voulez déterminer le nombre d'écouteurs actuellement enregistrés auprès de l'objet Key. Les objets sont ajoutés et supprimés dans ce tableau en appelant les méthodes `addListener()` et `removelistener()`.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

## Exemple

L'exemple suivant indique comment utiliser la propriété `length` pour déterminer le nombre d'objets écouteurs actuellement enregistrés auprès de l'objet Key.

```
var myListener:Object = new Object();
myListener.onKeyDown = function () {
trace ("You pressed a key.");
}
Key.addListener(myListener);

trace(Key._listeners.length); // Output: 1
```

# onKeyDown (écouteur d'événement Key.onKeyDown)

`onKeyDown = fonction() {}`

Notifié lorsqu'une touche est enfoncée. Pour utiliser `onKeyDown`, vous devez créer un objet écouteur. Vous pouvez ensuite définir une fonction pour `onKeyDown` et utiliser `addListener()` pour enregistrer l'écouteur auprès de l'objet Key, comme indiqué dans l'exemple suivant :

```
var keyListener:Object = new Object();
keyListener.onKeyDown = function() {
  trace("DOWN -> Code: "+Key.getCode()+"\tACSII: "+Key.getAscii()+"\tKey:
  "+chr(Key.getAscii()));
};
keyListener.onKeyUp = function() {
  trace("UP -> Code: "+Key.getCode()+"\tACSII: "+Key.getAscii()+"\tKey:
  "+chr(Key.getAscii()));
};
Key.addListener(keyListener);
```

Les écouteurs permettent à divers blocs de code de coopérer car plusieurs écouteurs peuvent recevoir une notification sur un événement unique.

Une application Flash ne peut contrôler que les événements de clavier qui se produisent dans son focus. Une application Flash ne peut pas détecter les événements de clavier dans une autre application.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

### Voir également

`addListener (méthode Key.addListener)`

## onKeyUp (écouteur d'événement Key.onKeyUp)

onKeyUp = fonction() {}

Notifié lorsqu'une touche est relâchée. Pour utiliser `onKeyUp`, vous devez créer un objet écouteur. Vous pouvez ensuite définir une fonction pour `onKeyUp` et utiliser `addListener()` pour enregistrer l'écouteur auprès de l'objet Key, comme indiqué dans l'exemple suivant :

```
var keyListener:Object = new Object();
keyListener.onKeyDown = function() {
  trace("DOWN -> Code: "+Key.getCode()+"\tACSII: "+Key.getAscii()+"\tKey:
  "+chr(Key.getAscii()));
};
keyListener.onKeyUp = function() {
  trace("UP -> Code: "+Key.getCode()+"\tACSII: "+Key.getAscii()+"\tKey:
  "+chr(Key.getAscii()));
};
Key.addListener(keyListener);
```

Les écouteurs permettent à divers blocs de code de coopérer car plusieurs écouteurs peuvent recevoir une notification sur un événement unique.

Une application Flash ne peut contrôler que les événements de clavier qui se produisent dans son focus. Une application Flash ne peut pas détecter les événements de clavier dans une autre application.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

### Voir également

`addListener (méthode Key.addListener)`

## PGDN (propriété Key.PGDN)

`statique publique PGDN : Number`

La valeur de code correspondant à la touche Pg. Suiv. (34).

**Disponibilité :** ActionScript 1.0 ; Flash Player 5

### Exemple

L'exemple suivant fait pivoter un clip intitulé `car_mc` lorsque vous appuyez sur la touche Pg. Suiv. ou Pg. Préc.

```
var keyListener:Object = new Object();
keyListener.onKeyDown = function() {
  if (Key.isDown(Key.PGDN)) {
  car_mc._rotation += 5;
  } else if (Key.isDown(Key.PGUP)) {
  car_mc._rotation -= 5;
  }
};
Key.addListener(keyListener);
```

## PGUP (propriété Key.PGUP)

```
statique publique PGUP : Number
```

La valeur de code correspondant à la touche Pg. Préc. (33).

**Disponibilité :** ActionScript 1.0 ; Flash Player 5

### Exemple

L'exemple suivant fait pivoter un clip intitulé `car_mc` lorsque vous appuyez sur la touche Pg. Suiv. ou Pg. Préc.

```
var keyListener:Object = new Object();
keyListener.onKeyDown = function() {
  if (Key.isDown(Key.PGDN)) {
  car_mc._rotation += 5;
  } else if (Key.isDown(Key.PGUP)) {
  car_mc._rotation -= 5;
  }
};
Key.addListener(keyListener);
```

## removeListener (méthode Key.removeListener)

```
statique publique removeListener(listener:Object) : Boolean
```

Supprime un objet précédemment enregistré auprès de `Key.addListener()`.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

### Paramètres

*`listener`*`:Object` - Un objet.

### Renvoie

`Boolean` - Si *`listener`* a été supprimé avec succès, la méthode renvoie `true`. Si *`listener`* n'a pas été supprimé avec succès (par exemple, parce que *`listener`* ne figurait pas dans la liste des écouteurs de l'objet Key), la méthode renvoie `false`.

### Exemple

L'exemple suivant déplace un clip intitulé `car_mc` à l'aide des flèches gauche et droite. L'écouteur est supprimé lorsque vous appuyez sur Echap et `car_mc` ne se déplace plus.

```
var keyListener:Object = new Object();
keyListener.onKeyDown = function() {
  switch (Key.getCode()) {
  case Key.LEFT :
  car_mc._x -= 10;
  break;
  case Key.RIGHT :
  car_mc._x += 10;
  break;
  case Key.ESCAPE :
  Key.removeListener(keyListener);
  }
};
Key.addListener(keyListener);
```

## RIGHT (propriété Key.RIGHT)

```
statique publique RIGHT : Number
```

La valeur de code correspondant à la flèche Droite (39).

**Disponibilité :** ActionScript 1.0 ; Flash Player 5

### Exemple

L'exemple suivant déplace un clip intitulé `car_mc` selon une distance constante (10) lorsque vous appuyez sur les touches fléchées. Un son est lu lorsque vous appuyez sur la barre d'espace. Pour cet exemple, affectez l'identificateur de liaison`horn_id` à un son figurant dans la bibliothèque.

```
var DISTANCE:Number = 10;
var horn_sound:Sound = new Sound();
horn_sound.attachSound("horn_id");
var keyListener_obj:Object = new Object();
keyListener_obj.onKeyDown = function() {
  switch (Key.getCode()) {
  case Key.SPACE :
  horn_sound.start();
  break;
```

```
  case Key.LEFT :
  car_mc._x -= DISTANCE;
  break;
  case Key.UP :
  car_mc._y -= DISTANCE;
  break;
  case Key.RIGHT :
  car_mc._x += DISTANCE;
  break;
  case Key.DOWN :
  car_mc._y += DISTANCE;
  break;
  }
};
Key.addListener(keyListener_obj);
```

# SHIFT (propriété Key.SHIFT)

```
statique publique SHIFT : Number
```

La valeur de code correspondant à la touche Maj (16).

**Disponibilité :** ActionScript 1.0 ; Flash Player 5

## Exemple

L'exemple suivant redimensionne `car_mc` lorsque vous appuyez sur la touche Maj.

```
var keyListener:Object = new Object();
keyListener.onKeyDown = function() {
  if (Key.isDown(Key.SHIFT)) {
  car_mc._xscale = 2;
  car_mc._yscale = 2;
  } else if (Key.isDown(Key.CONTROL)) {
  car_mc._xscale /= 2;
  car_mc._yscale /= 2;
  }
};
Key.addListener(keyListener);
```

# SPACE (propriété Key.SPACE)

```
statique publique SPACE : Number
```

La valeur de code correspondant à la barre d'espace (32).

**Disponibilité :** ActionScript 1.0 ; Flash Player 5

### Exemple

L'exemple suivant déplace un clip intitulé `car_mc` selon une distance constante (10) lorsque vous appuyez sur les touches fléchées. Un son est lu lorsque vous appuyez sur la barre d'espace. Pour cet exemple, affectez l'identificateur de liaison`horn_id` à un son figurant dans la bibliothèque.

```
var DISTANCE:Number = 10;
var horn_sound:Sound = new Sound();
horn_sound.attachSound("horn_id");
var keyListener_obj:Object = new Object();
keyListener_obj.onKeyDown = function() {
  switch (Key.getCode()) {
  case Key.SPACE :
  horn_sound.start();
  break;
  case Key.LEFT :
  car_mc._x -= DISTANCE;
  break;
  case Key.UP :
  car_mc._y -= DISTANCE;
  break;
  case Key.RIGHT :
  car_mc._x += DISTANCE;
  break;
  case Key.DOWN :
  car_mc._y += DISTANCE;
  break;
  }
};
Key.addListener(keyListener_obj);
```

## TAB (propriété Key.TAB)

```
statique publique TAB : Number
```

La valeur de code correspondant à la touche Tab (9).

**Disponibilité :** ActionScript 1.0 ; Flash Player 5

### Exemple

L'exemple suivant crée un champ de texte et y affiche la date lorsque vous appuyez sur la touche Tabulation.

```
this.createTextField("date_txt", this.getNextHighestDepth(), 0, 0, 100,
  22);
date_txt.autoSize = true;
var keyListener:Object = new Object();
keyListener.onKeyDown = function() {
  if (Key.isDown(Key.TAB)) {
```

```
  var today_date:Date = new Date();
  date_txt.text = today_date.toString();
  }
};
Key.addListener(keyListener);
```

Lorsque vous utilisez cet exemple, assurez-vous de sélectionner Contrôle > Désactiver les raccourcis clavier dans l'environnement de test.

La méthode `MovieClip.getNextHighestDepth()` utilisée dans cet exemple nécessite Flash Player 7 ou une version ultérieure. Si votre fichier SWF inclut un composant de la version 2, utilisez la classe DepthManager des composants de la version 2 à la place de la méthode `MovieClip.getNextHighestDepth()`.

# UP (propriété Key.UP)

```
statique publique UP : Number
```

La valeur de code correspondant à la flèche Haut (38).

**Disponibilité :** ActionScript 1.0 ; Flash Player 5

## Exemple

L'exemple suivant déplace un clip intitulé `car_mc` selon une distance constante (10) lorsque vous appuyez sur les touches fléchées. Un son est lu lorsque vous appuyez sur la barre d'espace. Pour cet exemple, affectez l'identificateur de liaison`horn_id` à un son figurant dans la bibliothèque.

```
var DISTANCE:Number = 10;
var horn_sound:Sound = new Sound();
horn_sound.attachSound("horn_id");
var keyListener_obj:Object = new Object();
keyListener_obj.onKeyDown = function() {
  switch (Key.getCode()) {
  case Key.SPACE :
  horn_sound.start();
  break;
  case Key.LEFT :
  car_mc._x -= DISTANCE;
  break;
  case Key.UP :
  car_mc._y -= DISTANCE;
  break;
  case Key.RIGHT :
  car_mc._x += DISTANCE;
  break;
  case Key.DOWN :
  car_mc._y += DISTANCE;
  break;
```

```
  }
};
Key.addListener(keyListener_obj);
```

# LoadVars

```
Object
  |
  +-LoadVars
```

```
public dynamic class LoadVars
extends Object
```

Vous pouvez utiliser la classe LoadVars pour vous assurer que le chargement des données s'est effectué avec succès et pour surveiller la progression du téléchargement. La classe LoadVars constitue une alternative à la fonction `loadVariables()` permettant de transférer des variables entre une application Flash et un serveur.

La classe LoadVars permet d'envoyer toutes les variables d'un objet à une adresse URL déterminée et de charger toutes les variables d'une adresse URL déterminée dans un objet. Elle vous permet également d'envoyer des variables spécifiques plutôt que la totalité d'entre elles, ce qui peut rendre votre application plus efficace. Vous pouvez utiliser le gestionnaire `LoadVars.onLoad` pour vous assurer que votre application s'exécute une fois les données chargées, et pas avant.

La classe LoadVars fonctionne de manière à peu près identique à la classe XML ; elle utilise les méthodes `load()`, `send()`, et `sendAndLoad()` pour communiquer avec un serveur. La principale différence entre les classes LoadVars et XML réside dans le fait que LoadVars transfère les paires nom et valeur ActionScript, plutôt qu'une arborescence XML DOM (Document Object Model) stockée dans l'objet XML. La classe LoadVars applique les mêmes restrictions de sécurité que la classe XML.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

## Voir également

`Fonction loadVariables`, `onLoad (gestionnaire LoadVars.onLoad)`, `XML`

## Résumé des propriétés

| Modificateurs | Propriété | Description |
|---|---|---|
| | `contentType:String` | Le type MIME qui est envoyé au serveur lorsque vous appelez `LoadVars.send()` ou `LoadVars.sendAndLoad()`. |
| | `loaded:Boolean` | Valeur booléenne indiquant si une opération `load` ou `sendAndLoad` s'est terminée, `undefined` par défaut. |

*Propriétés héritées de la classe Object*

`constructeur (propriété Object.constructor), __proto__ (Object.__proto__ property), prototype (Object.prototype, propriété), __resolve (Object.__resolve, propriété)`

## Résumé des événements

| Evénement | Description |
|---|---|
| `onData = function(src:String) {}` | Invoqué lorsque les données ont été totalement téléchargées à partir du serveur ou lorsqu'une erreur se produit au cours du téléchargement des données à partir d'un serveur. |
| `onHTTPStatus = function(httpStatus:Number) {}` | Invoqué lorsque Flash Player reçoit un code d'état HTTP du serveur. |
| `onLoad = function(success:Boolean) {}` | Invoqué lorsqu'une opération `LoadVars.load()` ou `LoadVars.sendAndLoad()` s'est terminée. |

## Résumé des constructeurs

| Signature | Description |
|---|---|
| `LoadVars()` | Crée un objet LoadVars. |

## Résumé de la méthode

| Modificateurs | Signature | Description |
|---|---|---|
| | `addRequestHeader(header:Object, headerValue:String) : Void` | Ajoute ou modifie les en-têtes de requête HTTP (tels que `Content-Type` ou `SOAPAction`) envoyés avec les actions `POST`. |
| | `decode(queryString:String) : Void` | Convertit la chaîne de variable en propriétés de l'objet LoadVars spécifié. |
| | `getBytesLoaded() : Number` | Renvoie le nombre d'octets téléchargés par`LoadVars.load()` ou `LoadVars.sendAndLoad()`. |
| | `getBytesTotal() : Number` | Renvoie le nombre total d'octets téléchargés par `LoadVars.load()` ou `LoadVars.sendAndLoad()`. |
| | `load(url:String) : Boolean` | Télécharge des variables à partir de l'URL spécifiée, analyse les données de variable et place les variables obtenues dans `my_lv`. |
| | `send(url:String, target:String, [method:String]) : Boolean` | Envoie les variables de l'objet *`my_lv`* vers l'URL spécifiée. |
| | `sendAndLoad(url:String, target:Object, [method:String]) : Boolean` | Publie les variables de l'objet `my_lv` vers l'URL spécifiée. |
| | `toString() : String` | Renvoie une chaîne contenant toutes les variables énumérables dans *`my_lv`*, au format de codage du contenu MIME *application/x-www-form-urlencoded*. |

*Méthodes héritées de la classe Object*

`addProperty (méthode Object.addProperty), hasOwnProperty (méthode Object.hasOwnProperty), isPropertyEnumerable (méthode Object.isPropertyEnumerable), isPrototypeOf (méthode Object.isPrototypeOf), registerClass (méthode Object.registerClass), toString (méthode Object.toString), unwatch (méthode Object.unwatch), valueOf (méthode Object.valueOf), watch (méthode Object.watch)`

# addRequestHeader (méthode LoadVars.addRequestHeader)

```
public addRequestHeader(header:Object, headerValue:String) : Void
```

Ajoute ou modifie les en-têtes de requête HTTP (tels que `Content-Type` ou `SOAPAction`) envoyés avec les actions `POST`. Dans la première utilisation, vous transmettez deux chaînes à la méthode : `header` et `headerValue`. Au cours de la deuxième utilisation, vous transmettez un tableau de chaînes, en alternant les noms d'en-têtes et les valeurs d'en-têtes.

En cas d'appels multiples pour définir le même nom d'en-tête, chaque valeur successive remplace la valeur définie dans l'appel précédent.

Les en-têtes HTTP standard suivants *ne peuvent pas* être ajoutés ou modifiés à l'aide de cette méthode : `Accept-Ranges`, `Age`, `Allow`, `Allowed`, `Connection`, `Content-Length`, `Content-Location`, `Content-Range`, `ETag`, `Host`, `Last-Modified`, `Locations`, `Max-Forwards`, `Proxy-Authenticate`, `Proxy-Authorization`, `Public`, `Range`, `Retry-After`, `Server`, `TE`, `Trailer`, `Transfer-Encoding`, `Upgrade`, `URI`, `Vary`, `Via`, `Warning`, et `WWW-Authenticate`.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

## Paramètres

*`header`*`:Object` - Une chaîne ou un tableau de chaînes représentant un nom d'en-tête de requête HTTP.

*`headerValue`*`:String` - Une chaîne qui représente la valeur associée à `header`.

## Exemple

L'exemple suivant ajoute un en-tête HTTP personnalisé intitulé `SOAPAction`, incluant la valeur `Foo`, à l'objet `my_lv` :

```
my_lv.addRequestHeader("SOAPAction", "'Foo'");
```

L'exemple suivant crée un tableau appelé `headers` qui contient deux en-têtes HTTP interchangeables et leurs valeurs. Le tableau est transmis en tant qu'argument à `addRequestHeader()`.

```
var headers = ["Content-Type", "text/plain", "X-ClientAppVersion", "2.0"];
my_lv.addRequestHeader(headers);
```

L'exemple suivant crée un nouvel objet LoadVars qui ajoute un en-tête de requête intitulé `FLASH-UUID`. L'en-tête inclut une variable pouvant être vérifiée par le serveur.

```
var my_lv:LoadVars = new LoadVars();
my_lv.addRequestHeader("FLASH-UUID", "41472");
my_lv.name = "Mort";
my_lv.age = 26;
my_lv.send("http://flash-mx.com/mm/cgivars.cfm", "_blank", "POST");
```

### Voir également

addRequestHeader (méthode XML.addRequestHeader)

## contentType (propriété LoadVars.contentType)

```
public contentType : String
```

Le type MIME qui est envoyé au serveur lorsque vous appelez `LoadVars.send()` ou `LoadVars.sendAndLoad()`. Le format par défaut est *application/x-www-form-urlencoded*.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

### Exemple

L'exemple suivant crée un objet LoadVars et affiche le type de contenu par défaut des données envoyées au serveur.

```
var my_lv:LoadVars = new LoadVars();
trace(my_lv.contentType); // output: application/x-www-form-urlencoded
```

### Voir également

send (méthode LoadVars.send), sendAndLoad (méthode LoadVars.sendAndLoad)

## decode (méthode LoadVars.decode)

```
public decode(queryString:String) : Void
```

Convertit la chaîne de variable en propriétés de l'objet LoadVars spécifié.

Cette méthode est utilisée en interne par le gestionnaire d'événements `LoadVars.onData`. La plupart des utilisateurs n'ont pas besoin d'appeler cette méthode directement. Si vous ignorez le gestionnaire d'événements `LoadVars.onData`, vous pouvez appeler explicitement `LoadVars.decode()` pour analyser une chaîne de variables.

**Disponibilité :** ActionScript 1.0 ; Flash Player 7

### Paramètres

*`queryString`*`:String` - Une chaîne de requête codée au format URL contenant les paires nom/valeur.

### Exemple

L'exemple suivant présente les trois variables :

```
// Create a new LoadVars object
var my_lv:LoadVars = new LoadVars();
//Convert the variable string to properties
```

```
my_lv.decode("name=Mort&score=250000");
trace(my_lv.toString());
// Iterate over properties in my_lv
for (var prop in my_lv) {
  trace(prop+" -> "+my_lv[prop]);
}
```

### Voir également

onData (gestionnaire LoadVars.onData), parseXML (XML.parseXML, méthode)

## getBytesLoaded (méthode LoadVars.getBytesLoaded)

```
public getBytesLoaded() : Number
```

Renvoie le nombre d'octets téléchargés par`LoadVars.load()` ou `LoadVars.sendAndLoad()`. Cette méthode renvoie `undefined` si aucune opération de chargement n'est en cours ou si une opération de chargement n'a pas encore commencé.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

### Renvoie

`Number` - Un entier.

### Exemple

L'exemple suivant utilise une occurrence ProgressBar et un objet LoadVars pour télécharger un fichier texte. Lorsque vous testez le fichier, deux informations s'affichent dans le panneau de sortie : celles indiquant si le chargement du fichier a réussi ou échoué et celles indiquant la quantité de données ayant été chargées dans le fichier SWF. Vous devez remplacer le paramètre URL de la commande `LoadVars.load()` afin qu'il se réfère à un fichier texte valide via HTTP. Si vous tentez d'utiliser cet exemple pour charger un fichier local résidant sur votre disque dur, il ne fonctionnera pas correctement car, en mode Tester l'animation, Flash Player charge intégralement les fichiers locaux. Pour voir ce code fonctionner, ajoutez une occurrence ProgressBar intitulée `loadvars_pb` sur la scène. Ajoutez ensuite le code ActionScript suivant à l'image 1 du scénario :

```
var loadvars_pb:mx.controls.ProgressBar;
var my_lv:LoadVars = new LoadVars();
loadvars_pb.mode = "manual";
this.createEmptyMovieClip("timer_mc", 999);
timer_mc.onEnterFrame = function() {
  var lvBytesLoaded:Number = my_lv.getBytesLoaded();
  var lvBytesTotal:Number = my_lv.getBytesTotal();
  if (lvBytesTotal != undefined) {
```

```
  trace("Loaded "+lvBytesLoaded+" of "+lvBytesTotal+" bytes.");
  loadvars_pb.setProgress(lvBytesLoaded, lvBytesTotal);
  }
};
my_lv.onLoad = function(success:Boolean) {
  loadvars_pb.setProgress(my_lv.getBytesLoaded(), my_lv.getBytesTotal());
  delete timer_mc.onEnterFrame;
  if (success) {
  trace("LoadVars loaded successfully.");
  } else {
  trace("An error occurred while loading variables.");
  }
};
my_lv.load("[place a valid URL pointing to a text file here]");
```

### Voir également

`load (méthode LoadVars.load)`, `sendAndLoad (méthode LoadVars.sendAndLoad)`

## getBytesTotal (méthode LoadVars.getBytesTotal)

```
public getBytesTotal() : Number
```

Renvoie le nombre total d'octets téléchargés par `LoadVars.load()` ou `LoadVars.sendAndLoad()`. Cette méthode renvoie `undefined` si aucune opération de chargement n'est en cours ou si une opération de chargement n'a pas commencé. Cette méthode renvoie également `undefined` si le nombre total d'octets ne peut pas être déterminé (par exemple, si le téléchargement a été lancé mais le serveur n'a pas transmis de longueur de contenu HTTP).

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

### Renvoie

`Number` - Un entier.

## Exemple

L'exemple suivant utilise une occurrence ProgressBar et un objet LoadVars pour télécharger un fichier texte. Lorsque vous testez le fichier, deux informations s'affichent dans le panneau de sortie : celles indiquant si le chargement du fichier a réussi ou échoué et celles indiquant la quantité de données ayant été chargées dans le fichier SWF. Vous devez remplacer le paramètre URL de la commande `LoadVars.load()` afin qu'il se réfère à un fichier texte valide via HTTP. Si vous tentez d'utiliser cet exemple pour charger un fichier local résidant sur votre disque dur, il ne fonctionnera pas correctement car, en mode de test d'animation, Flash Player charge intégralement les fichiers locaux. Pour voir ce code fonctionner, ajoutez une occurrence ProgressBar intitulée `loadvars_pb` sur la scène. Ajoutez ensuite le code ActionScript suivant à l'image 1 du scénario :

```
var loadvars_pb:mx.controls.ProgressBar;
var my_lv:LoadVars = new LoadVars();
loadvars_pb.mode = "manual";
this.createEmptyMovieClip("timer_mc", 999);
timer_mc.onEnterFrame = function() {
  var lvBytesLoaded:Number = my_lv.getBytesLoaded();
  var lvBytesTotal:Number = my_lv.getBytesTotal();
  if (lvBytesTotal != undefined) {
  trace("Loaded "+lvBytesLoaded+" of "+lvBytesTotal+" bytes.");
  loadvars_pb.setProgress(lvBytesLoaded, lvBytesTotal);
  }
};
my_lv.onLoad = function(success:Boolean) {
  loadvars_pb.setProgress(my_lv.getBytesLoaded(), my_lv.getBytesTotal());
  delete timer_mc.onEnterFrame;
  if (success) {
  trace("LoadVars loaded successfully.");
  } else {
  trace("An error occurred while loading variables.");
  }
};
my_lv.load("[place a valid URL pointing to a text file here]");
```

## Voir également

load (méthode LoadVars.load), sendAndLoad (méthode LoadVars.sendAndLoad)

# load (méthode LoadVars.load)

```
public load(url:String) : Boolean
```

Télécharge des variables à partir de l'URL spécifiée, analyse les données de variable et place les variables obtenues dans `my_lv`. Toutes les propriétés contenues dans `my_lv` ayant les mêmes noms que les variables téléchargées sont écrasées. Toutes les propriétés contenues dans `my_lv` ayant des noms différents de ceux des variables téléchargées ne sont pas supprimées. Il s'agit d'une action asynchrone.

Les données téléchargées doivent être dans le type de contenu MIME *application/x-www-form-urlencoded*.

Il s'agit du même format que celui utilisé par `loadVariables()`.

De plus, dans les fichiers publiés pour Flash Player 7, le respect de la casse est pris en charge pour les variables externes chargées via `LoadVars.load()`.

Cette méthode est similaire à `XML.load()`.

*Remarque :* Si un fichier en cours de chargement contient des caractères non-ASCII (dans un grand nombre de langues autres que l'anglais), il est recommandé de sauvegarder le fichier avec le codage UTF-8 ou UTF-16 plutôt que le format non-Unicode tel que ASCII.

Lorque vous utilisez cette méthode, prenez en considération le modèle de sécurité de Flash Player :

Pour Flash Player 8 :

- Le chargement de données n'est pas autorisé si le fichier SWF appelant se trouve dans le Sandbow du système de fichiers local et si la ressource cible se trouve dans un Sandbox du réseau.
- Le chargement de données n'est également pas autorisé si le fichier SWF appelant provient d'un Sandbox du réseau et si la ressource cible est locale.

Pour plus d'informations, consultez le :

- Chapitre 17, « Fonctionnement de la sécurité » du guide *Formation à ActionScript 2.0 dans Flash*
- Livre blanc sur la sécurité de Flash Player 8
- Livre blanc concernant les API liées à la sécurité de Flash Player 8 à l'adresse http://www.macromedia.com/go/fp8_security_apis

Les sites Web pour Flash Player 7 et les versions ultérieures peuvent autoriser l'accès interdomaines à une ressource via un fichier de régulation interdomaines. Dans les fichiers SWF d'une version exécutée dans Flash Player 7 ou une version ultérieure, le paramètre `url` doit se trouver exactement dans le même domaine. Par exemple, un fichier SWF à l'adresse www.someDomain.com ne peut charger que des données provenant de sources se trouvant également à l'adresse store.someDomain.com,

Pour les fichiers SWF lus par une version antérieure à Flash Player 7, le paramètre `url` doit correspondre au superdomaine du fichier SWF envoyant cet appel. Le superdomaine est dérivé en supprimant le composant le plus à gauche de l'URL d'un fichier. Par exemple, un fichier SWF à l'adresse www.someDomain.com peut charger des données provenant de sources à l'adresse store.someDomain.com, étant donné que les deux fichiers sont dans le même superdomaine de someDomain.com.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

## Paramètres

*`url`*`:String` - Une chaîne ; l'URL permettant de télécharger les variables. Si le fichier SWF effectuant cet appel s'exécute dans un navigateur Web, `url` doit appartenir au même domaine que le fichier SWF.

## Renvoie

`Boolean` - `false` si aucun paramètre (null) n'est transmis ; `true` sinon. Utilisez le gestionnaire d'événements `onLoad()` pour vérifier que les données ont bien été téléchargées.

## Exemple

Le code suivant définit une fonction du gestionaire `onLoad` indiquant à quel moment les données sont renvoyées à l'application Flash à partir d'un script PHP côté serveur, puis charge les données dans passvars.php.

```
var my_lv:LoadVars = new LoadVars();
my_lv.onLoad = function(success:Boolean) {
  if (success) {
  trace(this.toString());
  } else {
  trace("Error loading/parsing LoadVars.");
  }
};
my_lv.load("http://www.helpexamples.com/flash/params.txt");
```

Vous trouverez un autre exemple dans le fichier guestbook.fla du dossier d'exemples d'ActionScript. Vous trouverez ci-dessous les chemins type de ce dossier :

- Windows : *lecteur d'amorçage*\Program Files\Macromedia\Flash 8\Samples et Tutorials\Samples\ActionScript
- Macintosh : *Macintosh HD*/Applications/Macromedia Flash 8/Samples et Tutorials/Samples/ActionScript

### Voir également

`load (XML.load, méhode)`, `loaded (propriété LoadVars.loaded)`, `onLoad (gestionnaire LoadVars.onLoad)`, `useCodepage (propriété System.useCodepage)`

## loaded (propriété LoadVars.loaded)

```
public loaded : Boolean
```

Valeur booléenne indiquant si une opération `load` ou `sendAndLoad` s'est terminée, `undefined` par défaut. Au début d'une opération `LoadVars.load()` ou `LoadVars.sendAndLoad()`, la propriété `loaded` est définie sur `false` ; lorsque l'opération se termine, la propriété `loaded` est définie sur `true`. Si l'opération n'est pas terminée ou a échoué avec une erreur, la propriété `loaded` reste définie sur `false`.

Cette propriété est similaire à la propriété `XML.loaded`.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

### Exemple

L'exemple suivant charge un fichier texte et affiche les informations dans le panneau de sortie lorsque l'opération se termine.

```
var my_lv:LoadVars = new LoadVars();
my_lv.onLoad = function(success:Boolean) {
  trace("LoadVars loaded successfully: "+this.loaded);
};
my_lv.load("http://www.helpexamples.com/flash/params.txt");
```

### Voir également

`load (méthode LoadVars.load)`, `sendAndLoad (méthode LoadVars.sendAndLoad)`, `load (XML.load, méhode)`

## constructeur LoadVars

```
public LoadVars()
```

Crée un objet LoadVars. Vous pouvez ensuite utiliser les méthodes de cet objet LoadVars pour envoyer et charger des données.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

### Exemple

L'exemple suivant crée un objet LoadVars intitulé `my_lv` :

```
var my_lv:LoadVars = new LoadVars();
```

## onData (gestionnaire LoadVars.onData)

### onData = fonction(src:String) {}

Invoqué lorsque les données ont été totalement téléchargées à partir du serveur ou lorsqu'une erreur se produit au cours du téléchargement des données à partir d'un serveur. Ce gestionnaire est invoqué avant l'analyse des données et peut être utilisé pour appeler une routine d'analyse personnalisée au lieu de celle intégrée à Flash Player. La valeur du paramètre `src` transmis à la fonction affectée à `LoadVars.onData` peut être `undefined` ou une chaîne contenant les paires nom et valeur de code URL téléchargées à partir du serveur. Si le paramètre `src` est `undefined`, une erreur s'est produite au cours du téléchargement des données à partir du serveur.

L'implémentation par défaut de `LoadVars.onData` appelle `LoadVars.onLoad`. Vous pouvez ignorer cette implémentation par défaut en affectant une fonction personnalisée à `LoadVars.onData`, mais `LoadVars.onLoad` n'est pas appelé sauf si vous l'appelez dans votre implémentation de `LoadVars.onData`.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

### Paramètres

*`src`*`:String` - Une chaîne ou `undefined` ; les données brutes (non analysées) provenant d'un appel de méthode `LoadVars.load()` ou `LoadVars.sendAndLoad()`.

### Exemple

L'exemple suivant charge un fichier texte et affiche le contenu dans une occurrence TextArea intitulée `content_ta` lorsque l'opération se termine. Si une erreur se produit, les informations s'affichent alors dans le panneau de sortie.

```
var my_lv:LoadVars = new LoadVars();
my_lv.onData = function(src:String) {
  if (src == undefined) {
    trace("Error loading content.");
    return;
  }
  content_ta.text = src;
};
my_lv.load("content.txt", my_lv, "GET");
```

### Voir également

`onLoad (gestionnaire LoadVars.onLoad)`, `onLoad (gestionnaire LoadVars.onLoad)`, `load (méthode LoadVars.load)`, `sendAndLoad (méthode LoadVars.sendAndLoad)`

## onHTTPStatus (gestionnaire LoadVars.onHTTPStatus)

onHTTPStatus = fonction(httpStatus:Number) {}

Invoqué lorsque Flash Player reçoit un code d'état HTTP du serveur. Ce gestionnaire vous permet de capturer les codes d'état HTTP et d'agir dessus.

Le gestionnaire `onHTTPStatus` est appelé avant `onData`, qui déclenche les appels vers `onLoad` avec une valeur `undefined` si le chargement échoue. Une fois que `onHTTPStatus` est déclenché, `onData` est *toujours* déclenché, que vous annuliez ou non `onHTTPStatus`. Afin de mieux utiliser le gestionnaire `onHTTPStatus`, vous devez écrire une fonction pour récupérer le résultat de l'appel `onHTTPStatus` ; vous pouvez alors utiliser le résultat dans vos gestionnaires `onData` et `onLoad`. Si `onHTTPStatus` n'est pas appelé, ceci signifie que le lecteur n'a pas essayé de lancer la requête URL. Ceci peut se produire étant donné que la requête viole les règles du Sandbox de sécurité du fichier SWF.

Si Flash Player ne peut pas obtenir une code d'état auprès du serveur, ou s'il ne peut pas communiquer avec le serveur, la valeur 0 par défaut est transmise à votre code ActionScript. Une valeur 0 peut être générée dans un lecteur quelconque (par exemple, si une URL déformée est requise), et une valeur 0 est toujours générée par le module Flash Player lorsqu'il est exécuté dans les navigateurs suivants, qui ne transmettent pas de codes d'état HTTP au lecteur : Netscape, Mozilla, Safari, Opera, et Internet Explorer pour Macintosh.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

### Paramètres

*`httpStatus`*`:Number` - Le code d'état HTTP renvoyé par le serveur. Par exemple, une valeur 404 indique que le serveur n'a pas détecté de correspondance pour l'URL requise. Les codes d'état HTTP sont répertoriées dans les sections 10.4 et 10.5 de la spécification HTTP à l'adresse ftp://ftp.isi.edu/in-notes/rfc2616.txt.

## Exemple

L'exemple suivant indique comment utiliser `onHTTPStatus()` pour faciliter le débogage. L'exemple collecte les codes d'état HTTP et affecte leur valeur et leur type à une occurrence de l'objet LoadVars. (Notez que cet exemple crée les membres d'occurrence `this.httpStatus` et `this.httpStatusType` à l'exécution.) La `onData` méthode utilise ces membres d'occurrence pour tracer les informations concernant la réponse HTTP qui peuvent faciliter le débogage.

```
var myLoadVars:LoadVars = new LoadVars();

myLoadVars.onHTTPStatus = function(httpStatus:Number) {
  this.httpStatus = httpStatus;
  if(httpStatus < 100) {
    this.httpStatusType = "flashError";
  }
  else if(httpStatus < 200) {
    this.httpStatusType = "informational";
  }
  else if(httpStatus < 300) {
    this.httpStatusType = "successful";
  }
  else if(httpStatus < 400) {
    this.httpStatusType = "redirection";
  }
  else if(httpStatus < 500) {
    this.httpStatusType = "clientError";
  }
  else if(httpStatus < 600) {
    this.httpStatusType = "serverError";
  }
}

myLoadVars.onData = function(src:String) {
  trace(">> " + this.httpStatusType + ": " + this.httpStatus);
  if(src != undefined) {
    this.decode(src);
    this.loaded = true;
    this.onLoad(true);
  }
  else {
    this.onLoad(false);
  }
}

myLoadVars.onLoad = function(success:Boolean) {
}

myLoadVars.load("http://weblogs.macromedia.com/mxna/flashservices/
  getMostRecentPosts.cfm");
```

### Voir également

onHTTPStatus (XML.onHTTPStatus, gestionnaire), load (méthode LoadVars.load), sendAndLoad (méthode LoadVars.sendAndLoad)

## onLoad (gestionnaire LoadVars.onLoad)

onLoad = fonction(success:Boolean) {}

Invoqué lorsqu'une opération `LoadVars.load()` ou `LoadVars.sendAndLoad()` s'est terminée. Si l'opération a réussi, `my_lv` est renseigné par les variables téléchargées par l'opération : ces variables sont disponibles lorsque ce gestionnaire est appelé.

La valeur par défaut de ce gestionnaire est undefined.

Ce gestionnaire d'événements est similaire à `XML.onLoad`.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

### Paramètres

*`success`*`:Boolean` - Une valeur booléenne qui indique si l'opération de chargement s'est terminée avec succès (`true`) ou a échoué (`false`).

### Exemple

L'exemple suivant ajoute une occurrence TextInput instance intitulée `name_ti`, une occurrence TextArea intitulée `result_ta`, et une occurrence Button intitulée `submit_button` sur la scène. Lorsque l'utilisateur clique sur l'occurrence de bouton Login, deux objets LoadVars sont créés : `send_lv` et `result_lv`. L'objet `send_lv` copie le nom de l'occurrence `name_ti` et envoie les données à greeting.cfm. Le résultat de ce script est chargé dans l'objet `result_lv` et la réponse du serveur est affichée dans l'occurrence TextArea (`result_ta`). Ajoutez le code ActionScript suivant à l'image 1 du scénario :

```
var submitListener:Object = new Object();
submitListener.click = function(evt:Object) {
  var result_lv:LoadVars = new LoadVars();
  result_lv.onLoad = function(success:Boolean) {
    if (success) {
      result_ta.text = result_lv.welcomeMessage;
    } else {
      result_ta.text = "Error connecting to server.";
    }
  };
  var send_lv:LoadVars = new LoadVars();
  send_lv.name = name_ti.text;
  send_lv.sendAndLoad("http://www.flash-mx.com/mm/greeting.cfm",
  result_lv, "POST");
```

```
};
submit_button.addEventListener("click", submitListener);
```

Pour afficher un exemple plus détaillé, consultez le fichier login.fla du dossier d'exemples d'ActionScript. Vous trouverez ci-dessous les chemins type de ce dossier :

- Windows : *lecteur d'amorçage*\Program Files\Macromedia\Flash 8\Samples et Tutorials\Samples\ActionScript
- Macintosh : *Macintosh HD*/Applications/Macromedia Flash 8/Samples et Tutorials/Samples/ActionScript

### Voir également

`onLoad (XML.onLoad, gestionnaire)`, `loaded (propriété LoadVars.loaded)`, `load (méthode LoadVars.load)`, `sendAndLoad (méthode LoadVars.sendAndLoad)`

## send (méthode LoadVars.send)

```
public send(url:String, target:String, [method:String]) : Boolean
```

Envoie les variables de l'objet *`my_lv`* vers l'URL spécifiée. Toutes les variables énumérables contenues dans *`my_lv`* sont concaténées dans une chaîne au format *`application/x-www-form-urlencoded`* par défaut ; cette chaîne est publiée sur l'URL à l'aide de la méthode HTTP `POST`. Il s'agit du même format que celui utilisé par `loadVariables()`. Le type de contenu MIME envoyé dans les en-têtes de requête HTTP est la valeur `my_lv.contentType` ou la valeur par défaut *`application/x-www-form-urlencoded`*. La méthode `POST` est utilisée sauf si `GET` est spécifié.

Vous devez spécifier le paramètre `target` pour vous assurer que le script ou l'application sera exécuté(e) à l'URL spécifiée. Si vous omettez le paramètre `target`, la fonction renvoie `true`, mais le script ou l'application ne sera pas exécuté(e).

La méthode `send()` est utile si vous souhaitez que la réponse du serveur :

- remplace le contenu SWF (utilisez `"_self"` en tant que paramètre `target`) ;
- s'affiche dans une nouvelle fenêtre (utilisez `"_blank"` en tant que paramètre `target`) ;
- s'affiche dans le parent de l'image ou dans l'image de plus haut niveau (utilisez `"_parent"` ou `"_top"`en tant que paramètre `target`) ;
- s'affiche dans une image nommée (utilisez le nom de l'image en tant que chaîne pour le paramètre `target`).

Si l'appel de la méthode `send()` a réussi, elle ouvre toujours une nouvelle fenêtre de navigateur ou remplace le contenu dans une fenêtre ou image existante. Si vous préférez envoyer des informations à un serveur et continuer à lire votre fichier SWF sans ouvrir de nouvelle fenêtre ou remplacer le contenu dans une fenêtre ou une image, vous devez alors utiliser `LoadVars.sendAndLoad()`.

Cette méthode est similaire à `XML.send()`.

L'environnement de test Flash utilise toujours la méthode GET. Pour effectuer le test à l'aide de la méthode POST, veillez à tenter de l'utiliser à partir d'un navigateur.

Lorque vous utilisez cette méthode, prenez en considération le modèle de sécurité de Flash Player :

- Pour Flash Player 8, la méthode n'est pas autorisée si le fichier SWF appelant se trouve dans un sandbox local non sécurisé.
- Pour Flash Player 7 et les versions ultérieures, la méthode n'est pas autorisée si le fichier SWF appelant se trouve dans un fichier local.

Pour plus d'informations, consultez le :

- Chapitre 17, « Fonctionnement de la sécurité » du guide *Formation à ActionScript 2.0 dans Flash*
- Livre blanc sur la sécurité de Flash Player 8
- Livre blanc concernant les API liées à la sécurité de Flash Player 8 à l'adresse http://www.macromedia.com/go/fp8_security_apis

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

## Paramètres

*`url`*`:String` - Une chaîne ; l'URL vers laquelle les variables doivent être transférées.

*`target`*`:String` - Une chaîne ; la fenêtre de navigateur ou l'image dans laquelle les réponses s'affichent. Vous pouvez entrer le nom d'une fenêtre spécifique ou le sélectionner à partir des noms cibles réservés suivants :

- `"_self"` spécifie le cadre actif de la fenêtre en cours d'utilisation.
- `"_blank"` crée une fenêtre.
- `"_parent"` appelle le parent du cadre actif.
- `"_top"` sélectionne le cadre de plus haut niveau de la fenêtre active.

*`method`*`:String` [facultatif] - Une chaîne ; la méthode `GET` ou `POST` du protocole HTTP. La valeur par défaut est `POST`.

### Renvoie

`Boolean` - Une valeur booléenne ; `false` si aucun paramètre n'est spécifié, `true` sinon.

### Exemple

L'exemple suivant copie deux valeurs à partir de champs de texte et envoie les données à un script CFM, utilisé pour traiter les informations. Par exemple, le script peut vérifier si l'utilisateur a obtenu un meilleur score, puis insérer ces données dans une table de base de données.

```
var my_lv:LoadVars = new LoadVars();
my_lv.playerName = playerName_txt.text;
my_lv.playerScore = playerScore_txt.text;
my_lv.send("setscore.cfm", "_blank", "POST");
```

### Voir également

`sendAndLoad (méthode LoadVars.sendAndLoad)`, `send (XML.send, méthode)`

## sendAndLoad (méthode LoadVars.sendAndLoad)

```
public sendAndLoad(url:String, target:Object, [method:String]) : Boolean
```

Publie les variables de l'objet `my_lv` vers l'URL spécifiée. La réponse du serveur est téléchargée, analysée en tant que données de variable, et les variables obtenues sont placées dans l'objet `target`.

Les variables sont publiées de la même manière que `LoadVars.send()`. Les variables sont téléchargées dans la `target` de la même manière que `LoadVars.load()`.

Lorque vous utilisez cette méthode, prenez en considération le modèle de sécurité de Flash Player :

Pour Flash Player 8 :

- Le chargement de données n'est pas autorisé si le fichier SWF appelant se trouve dans le Sandbow du système de fichiers local et si la ressource cible se trouve dans un Sandbox du réseau.
- Le chargement de données n'est également pas autorisé si le fichier SWF appelant provient d'un Sandbox du réseau et si la ressource cible est locale.

Pour plus d'informations, consultez le :

- Chapitre 17, « Fonctionnement de la sécurité » du guide *Formation à ActionScript 2.0 dans Flash*
- Livre blanc sur la sécurité de Flash Player 8

- Livre blanc concernant les API liées à la sécurité de Flash Player 8 à l'adresse http://www.macromedia.com/go/fp8_security_apis

Pour Flash Player 7 et version ultérieure :

- Les sites Web permettent l'accès interdomaines à une ressource via un fichier de régulation interdomaines.
- Dans les fichiers SWF d'une version exécutée dans Flash Player 7 ou une version ultérieure, le paramètre `url` doit se trouver exactement dans le même domaine. Par exemple, un fichier SWF à l'adresse www.someDomain.com ne peut charger que des données provenant de sources se trouvant également à l'adresse store.someDomain.com,

Pour les fichiers SWF lus par une version antérieure à Flash Player 7, le paramètre `url` doit correspondre au superdomaine du fichier SWF envoyant cet appel. Le superdomaine est dérivé en supprimant le composant le plus à gauche de l'URL d'un fichier. Par exemple, un fichier SWF à l'adresse www.someDomain.com peut charger des données provenant de sources à l'adresse store.someDomain.com, étant donné que les deux fichiers sont dans le même superdomaine intitulé someDomain.com.

Cette méthode est similaire à `XML.sendAndLoad()`.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

## Paramètres

*`url`*`:String` - Une chaîne ; l'URL vers laquelle les variables doivent être transférées. Si le fichier SWF effectuant cet appel s'exécute dans un navigateur Web, `url` doit appartenir au même domaine que le fichier SWF.

*`target`*`:Object` - L'objet LoadVars ou XML qui reçoit les variables téléchargées.

*`method`*`:String` [facultatif] - Une chaîne ; la méthode `GET` ou `POST` du protocole HTTP. La valeur par défaut est `POST`.

## Renvoie

`Boolean` - Une valeur booléenne.

## Exemple

Pour l'exemple suivant, ajoutez une occurrence TextInput instance intitulée `name_ti`, une occurrence TextArea intitulée `result_ta`, et une occurrence Button intitulée `submit_button` sur la scène. Lorsque l'utilisateur clique sur l'occurrence de bouton Login dans l'exemple suivant, deux objets LoadVars sont créés : `send_lv` et `result_lv`. L'objet `send_lv` copie le nom de l'occurrence `name_ti` et envoie les données à greeting.cfm. Le résultat de ce script est chargé dans l'objet `result_lv` et la réponse du serveur s'affiche dans l'occurrence TextArea (`result_ta`). Ajoutez le code ActionScript suivant à l'image 1 du scénario :

```
var submitListener:Object = new Object();
submitListener.click = function(evt:Object) {
  var result_lv:LoadVars = new LoadVars();
  result_lv.onLoad = function(success:Boolean) {
  if (success) {
    result_ta.text = result_lv.welcomeMessage;
  } else {
    result_ta.text = "Error connecting to server.";
  }
  };
  var send_lv:LoadVars = new LoadVars();
  send_lv.name = name_ti.text;
  send_lv.sendAndLoad("http://www.flash-mx.com/mm/greeting.cfm",
  result_lv, "POST");
};
submit_button.addEventListener("click", submitListener);
```

Pour afficher un exemple plus détaillé, consultez le fichier login.fla du dossier d'exemples d'ActionScript. Les chemins type du dossier d'exemples d'ActionScript sont :

- Windows : *lecteur d'amorçage*\Program Files\Macromedia\Flash 8\Samples et Tutorials\Samples\ActionScript
- Macintosh : *Macintosh HD*/Applications/Macromedia Flash 8/Samples et Tutorials/Samples/ActionScript

### Voir également

`send (méthode LoadVars.send)`, `load (méthode LoadVars.load)`, `sendAndLoad (XML.sendAndLoad, méthode)`

## toString (méthode LoadVars.toString)

```
public toString() : String
```

Renvoie une chaîne contenant toutes les variables énumérables dans *`my_lv`*, au format de codage du contenu MIME *application/x-www-form-urlencoded*.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

### Renvoie

`String` - Une chaîne.

### Exemple

L'exemple suivant instancie un nouvel objet `LoadVars()`, crée deux propriétés et utilise `toString()` pour renvoyer une chaîne contenant les deux propriétés au format de code URL :

```
var my_lv:LoadVars = new LoadVars();
```

```
my_lv.name = "Gary";
my_lv.age = 26;
trace (my_lv.toString()); //output: age=26&name=Gary
```

# LocalConnection

```
Object
  |
  +-LocalConnection
```

`public dynamic class` ***LocalConnection***
`extends Object`

La classe LocalConnection vous permet de développer des fichiers SWF qui peuvent échanger des instructions entre eux sans utiliser `fscommand()` ou JavaScript. Les objets LocalConnection peuvent communiquer uniquement avec les fichiers SWF s'exécutant sur le même ordinateur client, mais peuvent s'exécuter dans diverses applications, par exemple un fichier SWF s'exécutant dans un navigateur et un fichier SWF s'exécutant dans une projection. Vous pouvez utiliser les objets LocalConnection pour envoyer et recevoir des données dans un fichier SWF unique, mais il ne s'agit pas de l'implémentation standard ; tous les exemples de cette section illustrent la communication entre différents fichiers SWF.

Les principales méthodes utilisées pour envoyer et recevoir des données sont les méthodes `LocalConnection.send()` et `LocalConnection.connect()`. Sous sa forme la plus basique, votre code implémente les commandes suivantes ; notez que les commandes `LocalConnection.send()` et `LocalConnection.connect()` spécifient le même nom de connexion, `lc_name` :

```
// Code in the receiving SWF file
this.createTextField("result_txt", 1, 10, 10, 100, 22);
result_txt.border = true;
var receiving_lc:LocalConnection = new LocalConnection();
receiving_lc.methodToExecute = function(param1:Number, param2:Number) {
result_txt.text = param1+param2;
};
receiving_lc.connect("lc_name");

// Code in the sending SWF file
var sending_lc:LocalConnection = new LocalConnection();
sending_lc.send("lc_name", "methodToExecute", 5, 7);
```

La manière la plus simple d'utiliser un objet LocalConnection est d'autoriser la communication uniquement entre les objets LocalConnection appartenant au même domaine, ce qui vous évitera tout problème de sécurité. Toutefois, si vous devez autoriser la communication entre les domaines, vous pouvez procéder de différentes façons pour implémenter vos mesures de sécurité. Pour plus d'informations, consultez la rubrique consacrée au paramètre `connectionName` dans `LocalConnection.send()` ainsi que les entrées `LocalConnection.allowDomain` et `LocalConnection.domain()`.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

## Résumé des propriétés

*Propriétés héritées de la classe Object*

| |
|---|
| `constructeur (propriété Object.constructor), __proto__ (Object.__proto__ property), prototype (Object.prototype, propriété), __resolve (Object.__resolve, propriété)` |

## Résumé des événements

| Evénement | Description |
|---|---|
| `allowDomain = function([sendingDomain:String]) {}` | Invoqué à chaque fois que receiving_lc reçoit une requête pour appeler une méthode à partir d'un objet LocalConnection d'envoi. |
| `allowInsecureDomain = function([sendingDomain:String]) {}` | Invoqué à chaque fois que receiving_lc, qui se trouve dans un fichier SWF hébergé sur un domaine utilisant un protocole sécurisé (HTTPS), reçoit une requête pour appeler une méthode à partir d'un objet LocalConnection d'envoi qui se trouve dans un fichier SWF hébergé à l'aide d'un protocole non sécurisé. |
| `onStatus = function(infoObject:Object) {}` | Invoqué une fois qu'un objet LocalConnection d'envoi a tenté d'envoyer une commande à un objet LocalConnection de réception. |

## Résumé des constructeurs

| Signature | Description |
|---|---|
| `LocalConnection()` | Crée un objet LocalConnection. |

Résumé de la méthode

| Modificateurs | Signature | Description |
|---|---|---|
| | `close() : Void` | Ferme (déconnecte) un objet LocalConnection. |
| | `connect(connectionName:String) : Boolean` | Prépare un objet LocalConnection à recevoir des commandes à partir d'une commande `LocalConnection.send()` (appelée l'*objet LocalConnection d'envoi*). |
| | `domain() : String` | Renvoie une chaîne représentant le domaine de l'emplacement du fichier SWF actuel. |
| | `send(connectionName:String, methodName:String, [args:Object]) : Boolean` | Invoque la méthode nommée `method` sur une connexion établie à l'aide de la commande `LocalConnection.connect( connectionName )` (l'objet LocalConnection de réception). |

*Méthodes héritées de la classe Object*

`addProperty (méthode Object.addProperty), hasOwnProperty (méthode Object.hasOwnProperty), isPropertyEnumerable (méthode Object.isPropertyEnumerable), isPrototypeOf (méthode Object.isPrototypeOf), registerClass (méthode Object.registerClass), toString (méthode Object.toString), unwatch (méthode Object.unwatch), valueOf (méthode Object.valueOf), watch (méthode Object.watch)`

# allowDomain (gestionnaire LocalConnection.allowDomain)

allowDomain = fonction([sendingDomain:String]) {}

Invoqué à chaque fois que receiving_lc reçoit une requête pour appeler une méthode à partir d'un objet LocalConnection d'envoi. Flash s'attend à ce que le code que vous implémentez dans ce gestionnaire renvoie une valeur booléenne true ou false. Si le gestionnaire ne renvoie pas de valeur true, la requête émanant de l'objet d'envoi est ignorée, et la méthode n'est pas appelée.

Lorsque ce gestionnaire d'événements est absent, Flash Player applique une stratégie de sécurité par défaut, équivalente au code suivant :

```
my_lc.allowDomain = function (sendingDomain)
{
  return (sendingDomain == this.domain());
}
```

Utilisez `LocalConnection.allowDomain` pour permettre de façon explicite aux objets LocalConnection issus de domaines spécifiés, ou d'un domaine quelconque, d'exécuter les méthodes de l'objet LocalConnection de réception. Si vous ne déclarez pas le paramètre sendingDomain, vous souhaitez probablement accepter les commandes émanant de tous les domaines : le code de votre gestionnaire renvoie alors simplement la valeur true. Si vous déclarez sendingDomain, vous souhaitez probablement comparer la valeur de sendingDomain aux domaines à partir desquels vous voulez accepter les commandes. Les exemples suivants illustrent les deux implémentations.

Dans les fichiers créés pour Flash Player 6, le paramètre `sendingDomain` contient le superdomaine de l'appelant. Dans les fichiers créés pour Flash Player 7 ou version ultérieure, le paramètre `sendingDomain`contient le domaine exact de l'appelant. Dans ce cas, pour autoriser l'accès aux fichiers SWF hébergés à l'adresse www.domain.com ou store.domain.com, vous devez autoriser l'accès de façon explicite à partir des deux domaines.

```
// For Flash Player 6
receiving_lc.allowDomain = function(sendingDomain) {
  return(sendingDomain=="domain.com");
}
// For Flash Player 7 or later
receiving_lc.allowDomain = function(sendingDomain) {
  return(sendingDomain=="www.domain.com" ||
    sendingDomain=="store.domain.com");
}
```

De plus, pour les fichiers créés pour Flash Player 7 ou version ultérieure, vous ne pouvez pas utiliser cette méthode pour permettre aux fichiers SWF hébergés via un protocole sécurisé (HTTPS) d'autoriser l'accès à partir de fichiers SWF hébergés à l'aide de protocoles non sécurisés ; vous devez utiliser le gestionnaire d'événements `LocalConnection.allowInsecureDomain` à la place.

La situation suivante peut parfois se produire. Supposons que vous chargiez un fichier SWF enfant à partir d'un domaine différent. Vous souhaitez implémenter cette méthode de manière à ce que le fichier SWF enfant puisse effectuer des appels LocalConnection vers le fichier SWF parent, mais vous ne connaissez pas le domaine final à partir duquel est issu le fichier SWF enfant. Cela peut se produire, par exemple, lorsque vous utilisez des redirections d'équilibrage de charge ou des serveurs tiers.

Dans ce cas, vous pouvez utiliser la propriété `MovieClip._url` pour implémenter cette méthode. Par exemple, si vous chargez un fichier SWF dans my_mc, vous pouvez ensuite implémenter cette méthode en vérifiant si l'argument du domaine correspond au domaine de `my_mc._url`. (Vous devez analyser le domaine à partir de l'adresse URL complète contenue dans `my_mc._url`.)

Si vous procédez ainsi, veuillez patienter jusqu'à la fin du chargement du fichier SWF dans `my_mc` car la propriété _url ne dispose pas de sa valeur correcte et finale tant que le fichier n'est pas entièrement chargé. La meilleure façon de déterminer si le chargement d'un fichier SWF enfant est terminé est d'utiliser `MovieClipLoader.onLoadComplete`.

La situation opposée peut également se produire : Vous pouvez créer un fichier SWF enfant qui souhaite accepter les appels LocalConnection émanant de son parent, mais qui ignore le domaine de ce dernier. Dans ce cas, implémentez cette méthode en vérifiant si l'argument du domaine correspond au domaine de _parent._url. Encore une fois, vous devez analyser le domaine à partir de l'adresse URL complète de _parent._url. Dans ce cas, il n'est pas nécessaire d'attendre la fin du chargement du fichier SWF parent ; le parent sera déjà chargé lorsque celui de l'enfant commencera.

**Disponibilité :** ActionScript 1.0 ; Flash Player 7

## Paramètres

*`sendingDomain`*`:String` [facultatif] - Une chaîne qui spécifie le domaine du fichier SWF contenant l'objet LocalConnection d'envoi.

## Exemple

L'exemple suivant illustre la façon dont un objet LocalConnection d'un fichier SWF de réception peut permettre aux fichiers SWF d'un domaine quelconque d'invoquer ses méthodes. Comparez cet exemple à celui de la méthode `LocalConnection.connect()`, dans lequel seuls les fichiers SWF appartenant au même domaine peuvent appeler la méthode `trace()` dans le fichier SWF de réception. Pour plus d'informations concernant l'utilisation du trait de soulignement (_) dans le nom de la connexion, consultez `LocalConnection.send()`.

```
this.createTextField("welcome_txt", this.getNextHighestDepth(), 10, 10,
  100, 20);
var my_lc:LocalConnection = new LocalConnection();
my_lc.allowDomain = function(sendingDomain:String) {
  domain_txt.text = sendingDomain;
  return true;
};
my_lc.allowInsecureDomain = function(sendingDomain:String) {
  return (sendingDomain == this.domain());
}
my_lc.sayHello = function(name:String) {
  welcome_txt.text = "Hello, "+name;
};
my_lc.connect("_mylc");
```

L'exemple suivant envoie une chaîne au fichier SWF précédent et affiche un message d'état indiquant si la connexion locale a réussi, ou non, à se connecter au fichier. Un composant TextInput intitulé `name_ti`, une occurrence TextArea intitulée `status_ta` et une occurrence Button intitulée `send_button` sont utilisés pour afficher le contenu.

```
var sending_lc:LocalConnection;
var sendListener:Object = new Object();
sendListener.click = function(evt:Object) {
  sending_lc = new LocalConnection();
  sending_lc.onStatus = function(infoObject:Object) {
  switch (infoObject.level) {
  case 'status' :
    status_ta.text = "LocalConnection connected successfully.";
    break;
  case 'error' :
    status_ta.text = "LocalConnection encountered an error.";
    break;
  }
  };
  sending_lc.send("_mylc", "sayHello", name_ti.text);
};
send_button.addEventListener("click", sendListener);
```

Si votre fichier SWF inclut un composant de la version 2, utilisez la classe DepthManager des composants de la version 2 à la place de la méthode `MovieClip.getNextHighestDepth()` utilisée dans l'exemple précédent.

Dans l'exemple suivant, le fichier SWF de réception, qui réside sur `thisDomain.com`, accepte uniquement les commandes issues de fichiers SWF situés dans `thisDomain.com` ou `thatDomain.com` :

```
var aLocalConn:LocalConnection = new LocalConnection();
aLocalConn.Trace = function(aString) {
  aTextField += aString+newline;
};
aLocalConn.allowDomain = function(sendingDomain) {
  return (sendingDomain == this.domain() || sendingDomain ==
  "www.macromedia.com");
};
aLocalConn.connect("_mylc");
```

Lorsque vous publiez des fichiers pour Flash Player 7 ou une version ultérieure, la correspondance exacte des domaines est utilisée. Cela signifie que l'exemple échoue si les fichiers SWF sont situés à l'adresse www.thatDomain.com ; en revanche, il fonctionne si les fichiers sont situés à l'adresse thatDomain.com.

### Voir également

connect (méthode LocalConnection.connect), domain (méthode LocalConnection.domain), send (méthode LocalConnection.send), _url (propriété MovieClip._url), onLoadComplete (écouteur d'événement MovieClipLoader.onLoadComplete), _parent, propriété

## allowInsecureDomain (gestionnaire LocalConnection.allowInsecureDomain)

allowInsecureDomain = fonction([sendingDomain:String]) {}

Invoqué à chaque fois que receiving_lc, qui se trouve dans un fichier SWF hébergé sur un domaine utilisant un protocole sécurisé (HTTPS), reçoit une requête pour appeler une méthode à partir d'un objet LocalConnection d'envoi qui se trouve dans un fichier SWF hébergé à l'aide d'un protocole non sécurisé. Flash s'attend à ce que le code que vous implémentez dans ce gestionnaire renvoie une valeur booléenne `true` ou `false`. Si le gestionnaire ne renvoie pas de valeur true, la requête émanant de l'objet d'envoi est ignorée, et la méthode n'est pas appelée.

Par défaut, les fichiers SWF hébergés via le protocole HTTPS sont accessibles uniquement aux autres fichiers SWF hébergés par l'intermédiaire du protocole HTTPS. Cette implémentation conserve l'intégrité fournie par le protocole HTTPS.

Il n'est pas recommandé d'utiliser cette méthode pour annuler le comportement par défaut car elle compromet la sécurité HTTPS. Cependant, vous devrez peut-être l'utiliser, par exemple, si vous devez autoriser l'accès aux fichiers HTTPS publiés pour Flash Player 7 ou version ultérieure à partir de fichiers HTTP publiés pour Flash Player 6.

Un fichier SWF publié pour Flash Player 6 peut utiliser le gestionnaire d'événements `LocalConnection.allowDomain` afin d'autoriser l'accès HTTPS à partir de HTTP. Toutefois, étant donné que la sécurité est implémentée différemment dans Flash Player 7, vous devez utiliser la méthode `LocalConnection.allowInsecureDomain()` pour permettre un tel accès dans les fichiers SWF publiés pour Flash Player 7 ou version ultérieure.

**Disponibilité :** ActionScript 1.0 ; Flash Player 7

### Paramètres

*`sendingDomain`*`:String` [facultatif] - Une chaîne qui spécifie le domaine du fichier SWF contenant l'objet LocalConnection d'envoi.

### Exemple

L'exemple suivant autorise les connexions à partir du domaine actuel ou à partir de l'adresse www.macromedia.com ; sinon, il autorise les connexions non sécurisées uniquement à partir du domaine actuel.

```
this.createTextField("welcome_txt", this.getNextHighestDepth(), 10, 10,
  100, 20);
var my_lc:LocalConnection = new LocalConnection();
my_lc.allowDomain = function(sendingDomain:String) {
  domain_txt.text = sendingDomain;
  return (sendingDomain == this.domain() || sendingDomain ==
  "www.macromedia.com");
};
my_lc.allowInsecureDomain = function(sendingDomain:String) {
  return (sendingDomain == this.domain());
}
my_lc.sayHello = function(name:String) {
  welcome_txt.text = "Hello, "+name;
};
my_lc.connect("lc_name");
```

Si votre fichier SWF inclut un composant de la version 2, utilisez la classe DepthManager des composants de la version 2 à la place de la méthode `MovieClip.getNextHighestDepth()` utilisée dans cet exemple.

### Voir également

`allowDomain (gestionnaire LocalConnection.allowDomain)`, `connect (méthode LocalConnection.connect)`

## close (méthode LocalConnection.close)

```
public close() : Void
```

Ferme (déconnecte) un objet LocalConnection. Appelez cette commande lorsque vous ne souhaitez plus que l'objet accepte de commandes, par exemple, lorsque vous souhaitez exécuter une commande `LocalConnection.connect()` utilisant le même paramètre `connectionName` dans un autre fichier SWF.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

### Exemple

L'exemple suivant ferme une connexion intitulée `receiving_lc` lorsque vous cliquez sur une occurrence de composant Button intitulée `close_button` :

```
this.createTextField("welcome_txt", this.getNextHighestDepth(), 10, 10,
  100, 22);
```

```
this.createTextField("status_txt", this.getNextHighestDepth(), 10, 42,
  100,44);

var receiving_lc:LocalConnection = new LocalConnection();
receiving_lc.sayHello = function(name:String) {
  welcome_txt.text = "Hello, "+name;
};
receiving_lc.connect("lc_name");
var closeListener:Object = new Object();
closeListener.click = function(evt:Object) {
  receiving_lc.close();
  status_txt.text = "connection closed";
};
close_button.addEventListener("click", closeListener);
```

La méthode `MovieClip.getNextHighestDepth()` utilisée dans cet exemple nécessite Flash Player 7 ou une version ultérieure. Si votre fichier SWF inclut un composant de la version 2, utilisez la classe DepthManager des composants de la version 2 à la place de la méthode `MovieClip.getNextHighestDepth()`.

### Voir également

`connect (méthode LocalConnection.connect)`

## connect (méthode LocalConnection.connect)

```
public connect(connectionName:String) : Boolean
```

Prépare un objet LocalConnection à recevoir des commandes à partir d'une commande `LocalConnection.send()` (appelée l'*objet LocalConnection d'envoi*). L'objet utilisé avec cette commande est appelé l'*objet LocalConnection de réception*. Les objets de réception et d'envoi doivent s'exécuter sur le même ordinateur client.

Assurez-vous de définir les méthodes associées à *`receiving_lc`* avant d'appeler cette méthode, comme indiqué dans tous les exemples de cette section.

Par défaut, Flash Player renvoie `connectionName` à la valeur `" superdomain :connectionName"`, où *`superdomain`* est le superdomaine du fichier SWF contenant la commande `LocalConnection.connect()`. Par exemple, si le fichier SWF contenant l'objet LocalConnection de réception se trouve à l'adresse www.someDomain.com, `connectionName` renvoie à `"someDomain.com:connectionName"`. (Si un fichier SWF se trouve sur l'ordinateur client, la valeur affectée au `superdomain` est `"localhost"`.)

En outre, par défaut, Flash Player permet à l'objet LocalConnection de réception d'accepter uniquement les commandes émanant des objets LocalConnection d'envoi dont le nom de connexion renvoie également à la valeur `" superdomain :connectionName"`. Ainsi, Flash facilite la communication entre les fichiers SWF situés dans le même domaine.

Si vous implémentez une communication uniquement entre les fichiers SWF appartenant au même domaine, spécifiez pour `connectionName` une chaîne qui ne commence pas par un trait de soulignement (_) et qui ne spécifie pas un nom de domaine (par exemple, `"myDomain:connectionName"`). Utilisez la même chaîne dans la commande `LocalConnection.connect( connectionName )`.

Si vous implémentez une communication entre des fichiers SWF appartenant à différents domaines, en spécifiant pour *`connectionName`* une chaîne qui commence par un trait de soulignement (_), le fichier SWF associé à l'objet LocalConnection de réception devient plus portable entre les domaines. Voici deux cas de figures possibles :

- Si la chaîne dédiée à *`connectionName`* ne commence pas par un trait de soulignement (_), Flash Player ajoute un préfixe au superdomaine et deux points (par exemple, `"myDomain:connectionName"`). Bien que cela permette de garantir que votre connexion n'entre pas en conflit avec les connexions de même nom dans d'autres domaines, tous les objets LocalConnection d'envoi doivent spécifier ce superdomaine (par exemple, `"myDomain:connectionName"`). Si le fichier SWF associé à l'objet LocalConnection de réception est déplacé vers un autre domaine, le lecteur modifie le préfixe afin qu'il reflète le nouveau superdomaine (par exemple, `"anotherDomain:connectionName"`). Tous les objets LocalConnection d'envoi doivent être modifiés manuellement pour pointer vers le nouveau superdomaine.
- Si la chaîne dédiée à *`connectionName`* commence par un trait de soulignement (par exemple, `"_connectionName"`), Flash Player n'ajoute pas de préfixe à la chaîne. Cela signifie que les objets LocalConnection de réception et d'envoi utilisent des chaînes identiques pour `connectionName`. Si l'objet de réception utilise `LocalConnection.allowDomain` pour spécifier que les connexions à partir de tous les domaines seront acceptées, le fichier SWF associé à l'objet LocalConnection de réception peut être déplacé vers un autre domaine, sans modifier les objets LocalConnection d'envoi.

Pour plus d'informations, consultez la rubrique consacrée au paramètre `connectionName` dans `LocalConnection.send()` ainsi que les entrées `LocalConnection.allowDomain` et `LocalConnection.domain()`.

*Remarque :* Les deux-points sont utilisés en tant que caractères spéciaux pour séparer le superdomaine de la chaîne `connectionName`. Une chaîne dédiée à `connectionName` contenant deux-points n'est pas valide.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

## Paramètres

*connectionName*:String - Une chaîne correspondant au nom de connexion spécifié dans la commande LocalConnection.send() qui souhaite communiquer avec *receiving_lc*.

## Renvoie

Boolean - Une valeur booléenne : true si aucun autre processus en cours d'exécution sur le même ordinateur client n'a déjà appelé cette commande utilisant la même valeur pour le paramètre *connectionName* ; false sinon.

## Exemple

L'exemple suivant indique comment un fichier SWF d'un domaine spécifique peut appeler une méthode intitulée printOut dans un fichier SWF de réception appartenant au même domaine.

Tout d'abord, créez un fichier SWF avec le code suivant :

```
this.createTextField("tf", this.getNextHighestDepth(), 10, 10, 300, 100);
var aLocalConnection:LocalConnection = new LocalConnection();
aLocalConnection.connect("demoConnection");
aLocalConnection.printOut = function(aString:String):Void{
  tf.text += aString;
}
```

Créez ensuite un deuxième fichier avec le code suivant :

```
var sending_lc:LocalConnection = new LocalConnection();
sending_lc.send("demoConnection", "printOut", "This is a message from file
  B. Hello.");
```

Pour tester cet exemple, exécutez le premier fichier SWF, puis le second.

Voici un autre exemple. Le fichier SWF 1 contient le code suivant permettant de créer un nouvel objet Sound qui lit un fichier MP3 lors de l'exécution. Une barre de progression intitulée playback_pb affiche la progression de la lecture du fichier MP3. Une occurrence du composant Label intitulée song_lbl affiche le nom du fichier MP3. Les boutons inclus dans les différents fichiers SWF seront utilisés pour contrôler la lecture à l'aide d'un objet LocalConnection.

```
var playback_pb:mx.controls.ProgressBar;
var my_sound:Sound;
playback_pb.setStyle("themeColor", "haloBlue");
this.createEmptyMovieClip("timer_mc", this.getNextHighestDepth());
var receiving_lc:LocalConnection = new LocalConnection();
receiving_lc.playMP3 = function(mp3Path:String, mp3Name:String) {
  song_lbl.text = mp3Name;
  playback_pb.indeterminate = true;
  my_sound = new Sound();
  my_sound.onLoad = function(success:Boolean) {
```

```
  playback_pb.indeterminate = false;
  };
  my_sound.onSoundComplete = function() {
  delete timer_mc.onEnterFrame;
  };
  timer_mc.onEnterFrame = function() {
  playback_pb.setProgress(my_sound.position, my_sound.duration);
  };
  my_sound.loadSound(mp3Path, true);
};
receiving_lc.connect("lc_name");
```

Le fichier SWF 2 contient un bouton intitulé `play_btn`. Lorsque vous cliquez sur ce bouton, il établit la connexion au fichier SWF 1 et transmet deux variables. La première variable contient le fichier MP3 à diffuser en flux continu ; la deuxième variable est le nom du fichier affiché dans l'occurrence du composant Label du fichier SWF 1.

```
play_btn.onRelease = function() {
  var sending_lc:LocalConnection = new LocalConnection();
  sending_lc.send("lc_name", "playMP3", "song1.mp3", "Album - 01 - Song");
};
```

Le fichier SWF 3 contient un bouton intitulé `play_btn`. Lorsque vous cliquez sur ce bouton, il établit la connexion au fichier SWF 1 et transmet deux variables. La première variable contient le fichier MP3 à diffuser en flux continu ; la deuxième variable est le nom du fichier affiché dans l'occurrence du composant Label du fichier SWF 1.

```
play_btn.onRelease = function() {
  var sending_lc:LocalConnection = new LocalConnection();
  sending_lc.send("lc_name", "playMP3", "song2.mp3", "Album - 02 - Another
  Song");
};
```

La méthode `MovieClip.getNextHighestDepth()` utilisée dans ces exemples nécessite Flash Player 7 ou une version ultérieure. Si votre fichier SWF inclut un composant de la version 2, utilisez la classe DepthManager des composants de la version 2 à la place de la méthode `MovieClip.getNextHighestDepth()`.

### Voir également

`send (méthode LocalConnection.send)`, `allowDomain (gestionnaire LocalConnection.allowDomain)`, `domain (méthode LocalConnection.domain)`

## domain (méthode LocalConnection.domain)

```
public domain() : String
```

Renvoie une chaîne représentant le domaine de l'emplacement du fichier SWF actuel.

Dans les fichiers SWF publiés pour Flash Player 6, la chaîne renvoyée est le superdomaine du fichier SWF actuel. Par exemple, si le fichier SWF se trouve à l'adresse www.macromedia.com, cette commande renvoie `"macromedia.com"`.

Dans les fichiers SWF publiés pour Flash Player 7 ou version ultérieure, la chaîne renvoyée est le domaine exact du fichier SWF actuel. Par exemple, si le fichier SWF se trouve à l'adresse www.macromedia.com, cette commande renvoie `"www.macromedia.com"`.

Si le fichier SWF actuel est un fichier local résidant sur l'ordinateur client, cette commande renvoie `"localhost"`.

L'emploi le plus courant de cette commande consiste à inclure le nom de domaine de l'objet LocalConnection d'envoi en tant que paramètre de la méthode que vous comptez invoquer dans l'objet LocalConnection de réception ou avec `LocalConnection.allowDomain` pour accepter les commandes issues d'un domaine spécifié. Si vous autorisez uniquement la communication entre les objets LocalConnection appartenant au même domaine, vous n'aurez probablement pas besoin d'utiliser cette commande.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

## Renvoie

`String` - Une chaîne représentant le domaine de l'emplacement du fichier SWF actuel ; pour plus d'informations, consultez la section Description.

## Exemple

Dans l'exemple suivant, un fichier SWF de réception accepte uniquement les commandes issues des fichiers SWF situés dans le même domaine ou à l'adresse macromedia.com :

```
// If both the sending and receiving SWF files are Flash Player 6,
// then use the superdomain
var my_lc:LocalConnection = new LocalConnection();
my_lc.allowDomain = function(sendingDomain):String{
  return (sendingDomain==this.domain() || sendingDomain=="macromedia.com");
}

// If either the sending or receiving SWF file is Flash Player 7 or later,
// then use the exact domain. In this case, commands from a SWF file posted
// at www.macromedia.com will be accepted, but those from one posted at
// a different subdomain, e.g. livedocs.macromedia.com, will not.
var my_lc:LocalConnection = new LocalConnection();
my_lc.allowDomain = function(sendingDomain):String{
  return (sendingDomain==this.domain() ||
  sendingDomain=="www.macromedia.com");
}
```

Dans l'exemple suivant, un fichier SWF d'envoi situé à l'adresse www.yourdomain.com invoque une méthode dans un fichier SWF de réception situé à l'adresse www.mydomain.com. Le fichier SWF d'envoi inclut son nom de domaine en tant que paramètre de la méthode qu'il invoque : le fichier SWF de réception peut ainsi renvoyer une valeur de réponse à un objet LocalConnection situé dans le domaine approprié. Le fichier SWF d'envoi spécifie également qu'il accepte uniquement les commandes issues de fichiers SWF à l'adresse mydomain.com.

Les numéros de ligne sont inclus à titre de référence. La séquence des événements est décrite dans la liste suivante :

- Le fichier SWF de réception se prépare à recevoir des commandes sur une connexion intitulée `"sum"` (ligne 11). Flash Player résout le nom de cette connexion en renvoyant `"mydomain.com:sum"` (consultez `LocalConnection.connect()`).
- Le fichier SWF d'envoi se prépare à recevoir une réponse sur l'objet LocalConnection intitulé `"result"` (ligne 67). Il spécifie également qu'il accepte uniquement les commandes issues de fichiers SWF à l'adresse mydomain.com (lignes 51 à 53).
- Le fichier SWF d'envoi invoque la méthode `aSum` d'une connexion intitulée `"mydomain.com:sum"` (ligne 68) et transmet les paramètres suivants : son superdomaine, le nom de la connexion permettant de recevoir la réponse (`"result"`) et les valeurs à utiliser par la méthode `aSum` (123 et 456).
- La méthode `aSum` (ligne 6) est invoquée avec les valeurs suivantes : expéditeur = `"mydomain.com:result"`, `replyMethod` = `"aResult"`, `n1` = 123, et `n2` = 456. Il exécute alors la ligne de code suivante :

```
this.send("mydomain.com:result", "aResult", (123 + 456));
```

- La méthode `aResult` (ligne 54) affiche la valeur renvoyée par `aSum` (579).

```
// The receiving SWF at http://www.mydomain.com/folder/movie.swf
// contains the following code

1 var aLocalConnection:LocalConnection = new LocalConnection();
2 aLocalConnection.allowDomain = function()
3 {
  // Allow connections from any domain
4 return true;
5 }
6 aLocalConnection.aSum = function(sender, replyMethod, n1, n2)
7 {
8 this.send(sender, replyMethod, (n1 + n2));
9 }
10
11 aLocalConnection.connect("sum");

// The sending SWF at http://www.yourdomain.com/folder/movie.swf
// contains the following code
```

```
  50 var lc:LocalConnection = new LocalConnection();
  51 lc.allowDomain = function(aDomain) {
       // Allow connections only from mydomain.com
  52 return (aDomain == "mydomain.com");
  53 }
  54 lc.aResult = function(aParam) {
  55 trace("The sum is " + aParam);
  56 }
     // determine our domain and see if we need to truncate it
  57 var channelDomain:String = lc.domain();
  58 if (getVersion() >= 7 && this.getSWFVersion() >= 7)
  59 {
       // split domain name into elements
  60 var domainArray:Array = channelDomain.split(".");

       // if more than two elements are found,
       // chop off first element to create superdomain
  61 if (domainArray.length > 2)
  62 {
  63 domainArray.shift();
  64 channelDomain = domainArray.join(".");
  65 }
  66 }

  67 lc.connect("result");
  68 lc.send("mydomain.com:sum", "aSum", channelDomain + ':' + "result",
"aResult", 123, 456);
```

### Voir également

allowDomain (gestionnaire LocalConnection.allowDomain), connect (méthode LocalConnection.connect)

## constructeur LocalConnection

```
public LocalConnection()
```

Crée un objet LocalConnection.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

### Exemple

L'exemple suivant illustre la manière dont la réception et l'envoi de fichiers SWF permettent de créer des objets LocalConnnection. Les deux fichiers SWF peuvent utiliser le même nom ou des noms différents pour leurs objets LocalConnection respectifs. Dans cet exemple, ils utilisent des noms différents.

```
// Code in the receiving SWF file
this.createTextField("result_txt", 1, 10, 10, 100, 22);
result_txt.border = true;
var receiving_lc:LocalConnection = new LocalConnection();
receiving_lc.methodToExecute = function(param1:Number, param2:Number) {
  result_txt.text = param1+param2;
};
receiving_lc.connect("lc_name");
```

Le fichier SWF suivant envoie la requête au premier fichier SWF.

```
// Code in the sending SWF file
var sending_lc:LocalConnection = new LocalConnection();
sending_lc.send("lc_name", "methodToExecute", 5, 7);
```

### Voir également

`connect (méthode LocalConnection.connect)`, `send (méthode LocalConnection.send)`

## onStatus (gestionnaire LocalConnection.onStatus)

onStatus = fonction(infoObject:Object) {}

Invoqué une fois qu'un objet LocalConnection d'envoi a tenté d'envoyer une commande à un objet LocalConnection de réception. Si vous souhaitez répondre à ce gestionnaire d'événements, vous devez créer une fonction pour traiter l'objet d'informations envoyé par l'objet LocalConnection.

Si l'objet d'informations renvoyé par ce gestionnaire d'événements contient une valeur de niveau d'état, cela signifie que Flash a réussi à envoyer la commande à un objet LocalConnection de réception. Cela ne signifie pas que Flash a réussi à appeler la méthode spécifiée de l'objet LocalConnection de réception ; cela signifie seulement que Flash a pu envoyer la commande. Par exemple, la méthode n'est pas invoquée si l'objet LocalConnection de réception n'autorise pas les connexions à partir du domaine d'envoi ou si la méthode n'existe pas. La seule façon de s'assurer que la méthode a été invoquée consiste à demander à l'objet de réception d'envoyer une réponse à l'objet d'envoi.

Si l'objet d'informations renvoyé par ce gestionnaire d'événements contient une valeur de niveau d'erreur, Flash ne peut pas envoyer la commande à un objet LocalConnection de réception : cela est probablement dû au fait qu'aucun objet LocalConnection de réception dont le nom correspond à celui spécifié dans la commande `sending_lc.send()` ayant appelé ce gestionnaire n'est connecté.

Outre ce gestionnaire `onStatus`, Flash est également doté d'une « super » fonction appelée `System.onStatus`. Si `onStatus` est appelé pour un objet spécifique et si aucune fonction n'est affectée pour y répondre, Flash exécute une fonction affectée à `System.onStatus` si elle existe.

Dans la plupart des cas, vous implémentez ce gestionnaire uniquement pour répondre à des conditions d'erreur, comme indiqué dans l'exemple suivant.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

## Paramètres

*`infoObject`*`:Object` - Un paramètre défini en fonction du message d'état. Pour plus de détails sur ce paramètre, consultez la section Description.

## Exemple

L'exemple suivant affiche un message d'état indiquant si le fichier SWF se connecte, ou non, à un autre objet de connexion locale intitulé `lc_name`. Un composant TextInput intitulé `name_ti`, une occurrence TextArea intitulée `status_ta` et une occurrence Button intitulée `send_button` sont utilisés pour afficher le contenu.

```
var sending_lc:LocalConnection;
var sendListener:Object = new Object();
sendListener.click = function(evt:Object) {
  sending_lc = new LocalConnection();
  sending_lc.onStatus = function(infoObject:Object) {
    switch (infoObject.level) {
    case 'status' :
      status_ta.text = "LocalConnection connected successfully.";
        break;
    case 'error' :
      status_ta.text = "LocalConnection encountered an error.";
      break;
    }
  };
  sending_lc.send("lc_name", "sayHello", name_ti.text);
};
send_button.addEventListener("click", sendListener);
```

## Voir également

`send (méthode LocalConnection.send)`, `onStatus (gestionnaire System.onStatus)`

# send (méthode LocalConnection.send)

```
public send(connectionName:String, methodName:String, [args:Object]) :
  Boolean
```

Invoque la méthode nommée `method` sur une connexion établie à l'aide de la commande `LocalConnection.connect( connectionName )` (l'objet LocalConnection de réception). L'objet utilisé avec cette commande est appelé l'objet LocalConnection d'envoi. Les fichiers SWF qui contiennent les objets d'envoi et de réception doivent s'exécuter sur le même ordinateur client.

La quantité de données que vous pouvez transmettre en tant que paramètres à cette commande est limitée à 40 Ko. Si la commande renvoie la valeur `false` mais si votre syntaxe est correcte, essayez de répartir les requêtes `LocalConnection.send()` en plusieurs commandes, chacune comportant moins de 40 Ko de données.

Comme nous l'avons vu dans l'entrée `LocalConnection.connect()`, Flash ajoute le superdomaine actuel à `connectionName` par défaut. Si vous implémentez la communication entre différents domaines, vous devez définir `connectionName` dans les objets LocalConnection d'envoi et de réception de sorte que Flash n'ajoute pas le superdomaine actuel à `connectionName`. Pour ce faire, procédez de l'une des deux façons suivantes :

- Placez un trait de soulignement (_) au début de `connectionName` dans les objets LocalConnection d'envoi et de réception. Dans le fichier SWF contenant l'objet de réception, utilisez `LocalConnection.allowDomain` pour spécifier que les connexions à partir de tous les domaines seront acceptées. Cette implémentation vous permet de stocker vos fichiers SWF d'envoi et de réception dans n'importe quel domaine.
- Incluez le superdomaine à `connectionName` dans l'objet LocalConnection d'envoi, par exemple, `myDomain.com:myConnectionName`. Dans l'objet de réception, utilisez `LocalConnection.allowDomain` pour spécifier que les connexions à partir du superdomaine spécifié seront acceptées (dans ce cas, myDomain.com) ou que les connexions à partir de tous les domaines seront acceptées.

*Remarque :* Vous ne pouvez pas spécifier de superdomaine dans `connectionName`pour l'objet LocalConnection de réception : vous pouvez le faire uniquement dans l'objet LocalConnection d'envoi.

Lorque vous utilisez cette méthode, prenez en considération le modèle de sécurité de Flash Player. Par défaut, un objet LocalConnection est associé au Sandbox du fichier SWF qui l'a créé et les appels interdomaines vers les objets LocalConnection ne sont pas autorisés si la méthode `LocalConnection.allowDomain()` a été invoquée.

Pour plus d'informations, consultez le :

- Chapitre 17, « Fonctionnement de la sécurité » du guide *Formation à ActionScript 2.0 dans Flash*
- Livre blanc concernant la sécurité de Flash Player 8 à l'adresse http://www.macromedia.com/go/fp8_security
- Livre blanc concernant les API liées à la sécurité de Flash Player 8 à l'adresse http://www.macromedia.com/go/fp8_security_apis

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

## Paramètres

*connectionName*:String - Une chaîne correspondant au nom de connexion spécifié dans la commande LocalConnection.connect() qui souhaite communiquer avec *sending_lc*.

*methodName*:String - Une chaîne spécifiant le nom de la méthode à invoquer dans l'objet LocalConnection de réception. Les noms de méthode suivants entraînent l'échec de la commande : send, connect, close, domain, onStatus et allowDomain.

*args*:Object [facultatif] - Arguments à transmettre à la méthode spécifiée.

## Renvoie

Boolean - Une valeur booléenne : true si Flash peut exécuter la requête ; false sinon.

*Remarque :* Une valeur true renvoyée ne signifie pas nécessairement que Flash a réussi à se connecter à un objet LocalConnection de réception ; cela signifie uniquement que la commande est correcte du point de vue syntaxique. Pour déterminer si la connexion a été établie, consultez LocalConnection.onStatus.

## Exemple

Pour obtenir un exemple de communication entre les objets LocalConnection appartenant au même domaine, consultez LocalConnection.connect(). Pour obtenir un exemple de communication entre les objets LocalConnection appartenant à un domaine quelconque, consultez LocalConnection.allowDomain. Pour obtenir un exemple de communication entre les objets LocalConnection appartenant à des domaines spécifiés, consultez LocalConnection.allowDomain et LocalConnection.domain().

## Voir également

allowDomain (gestionnaire LocalConnection.allowDomain), connect (méthode LocalConnection.connect), domain (méthode LocalConnection.domain), onStatus (gestionnaire LocalConnection.onStatus)

# Locale (mx.lang.Locale)

```
Object
  |
  +-mx.lang.Locale
```

```
public class Locale
extends Object
```

La classe mx.lang.Locale vous permet de contrôler la façon dont le texte multilingue s'affiche dans un fichier SWF. Le panneau Chaînes de Flash vous permet d'utiliser des ID de chaîne au lieu de littéraux de chaîne dans les champs de texte dynamique. Vous pouvez ainsi créer un fichier SWF affichant du texte chargé à partir d'un fichier XML spécifique à une langue. Le fichier XML doit répondre à la norme XML Localization Interchange File Format (XLIFF). Vous pouvez afficher les chaînes spécifiques à une langue contenues dans les fichiers XLIFF de trois manières différentes :

- `"automatically at runtime"` - Flash Player remplace les ID de chaîne par les chaînes provenant du fichier XML correspondant au code de langue du système par défaut renvoyé par System.capabilities.language.
- `"manually using stage language"` - Les ID de chaîne sont remplacés par les chaînes au moment de la compilation et ne peuvent pas être modifiés par Flash Player.
- `"via ActionScript at runtime"` - Le remplacement des ID de chaîne est commandé à l'aide d'ActionScript à l'exécution. Cette option vous permet de contrôler la synchronisation et la langue du remplacement des ID de chaîne.

Vous pouvez utiliser les propriétés et les méthodes de cette classe lorsque vous souhaitez remplacer les ID de chaîne « via ActionScript lors de l'exécution ».

Toutes les propriétés et les méthodes disponibles sont statiques, ce qui signifie qu'elles sont accessibles via la classe mx.lang.Locale plutôt que via une occurrence de la classe.

*Remarque :* La classe Locale est différente des autres classes de la section ActionScript 2.0 Language Reference, étant donné qu'elle ne fait pas partie de Flash Player. Etant donné que cette classe se trouve dans le chemin de classe Flash Authoring, elle est automatiquement compilée dans vos fichiers SWF. L'utilisation de la classe Locale augmente légèrement la taille du fichier SWF étant donné que la classe est compilée dans le SWF.

**Disponibilité :** ActionScript 2,0 ; Flash Player 7

## Résumé des propriétés

| Modificateurs | Propriété | Description |
|---|---|---|
| static | autoReplace:Boolean | Détermine si les chaînes sont remplacées automatiquement une fois le chargement du fichier xml terminé. |
| static | languageCodeArray:Array [lecture seule] | Un tableau contenant les codes de langue des langues ayant été spécifiées ou chargées dans le fichier FLA. |
| static | stringIDArray:Array [lecture seule] | Un tableau contenant tous les ID de chaîne du fichier FLA. |

*Propriétés héritées de la classe Object*

constructeur (propriété Object.constructor), __proto__ (Object.__proto__ property), prototype (Object.prototype, propriété), __resolve (Object.__resolve, propriété)

## Résumé de la méthode

| Modificateurs | Signature | Description |
|---|---|---|
| static | addDelayedInstance(instance:Object, stringID:String) : Void | Ajoute la paire {instance, string ID} dans le tableau interne en vue d'une utilisation ultérieure. |
| static | addXMLPath(langCode:String, path:String) : Void | Ajoute la paire {languageCode et languagePath} dans le tableau interne en vue d'une utilisation ultérieure. |
| static | checkXMLStatus() : Boolean | Renvoie true si le fichier xml est chargé ; false sinon. |
| static | getDefaultLang() : String | Le code de langue par défaut tel que défini dans la boîte de dialogue Paramètres du panneau Chaînes ou en appelant la méthode setDefaultLang(). |
| static | initialize() : Void | Détermine automatiquement la langue à utiliser et charge le dossier de langue XML. |
| static | loadLanguageXML(xmlLanguageCode:String, customXmlCompleteCallback:Function) : Void | Charge le dossier de langue XML spécifié. |
| static | loadString(id:String) : String | Renvoie la valeur de chaîne associée à l'ID de chaîne spécifié dans la langue actuellement utilisée. |

| Modificateurs | Signature | Description |
|---|---|---|
| static | loadStringEx(stringID:String, languageCode:String) : String | Renvoie la valeur de chaîne associée à l'ID de chaîne et au code de langue spécifiés. |
| static | setDefaultLang(langCode:String) : Void | Définit le code de langue par défaut. |
| static | setLoadCallback(loadCallback:Function) : Void | Définit la fonction de rappel qui sera appelée une fois le fichier XML chargé. |
| static | setString(stringID:String, languageCode:String, stringValue:String) : Void | Définit la nouvelle valeur de chaîne d'un ID de chaîne et d'un code de langue spécifiés. |

*Méthodes héritées de la classe Object*

addProperty (méthode Object.addProperty), hasOwnProperty (méthode Object.hasOwnProperty), isPropertyEnumerable (méthode Object.isPropertyEnumerable), isPrototypeOf (méthode Object.isPrototypeOf), registerClass (méthode Object.registerClass), toString (méthode Object.toString), unwatch (méthode Object.unwatch), valueOf (méthode Object.valueOf), watch (méthode Object.watch)

# addDelayedInstance (méthode Locale.addDelayedInstance)

```
statique publique addDelayedInstance(instance:Object, stringID:String) :
  Void
```

Ajoute la paire {instance, string ID} dans le tableau interne en vue d'une utilisation ultérieure. Flash procède généralement ainsi lorsque la méthode de remplacement des chaînes est définie sur `"automatically at runtime"`.

**Disponibilité :** ActionScript 2,0 ; Flash Player 7

## Paramètres

*instance*:Object - Nom de l'occurrence du champ de texte à renseigner.

*stringID*:String - ID de chaîne de langue.

### Exemple

L'exemple suivant utilise la propriété `autoReplace` et la méthode `addDelayedInstance()` pour renseigner une champ de texte sur la scène avec la chaîne `IDS_GREETING` provenant du fichier de langue XML anglais.

```
import mx.lang.Locale;
greeting_txt.autoSize = "left";
Locale.autoReplace = true;
Locale.addDelayedInstance(greeting_txt, "IDS_GREETING");
Locale.loadLanguageXML("en");
```

## addXMLPath (méthode Locale.addXML)

```
statique publique addXMLPath(langCode:String, path:String) : Void
```

Ajoute la paire {languageCode et languagePath} dans le tableau interne en vue d'une utilisation ultérieure. Flash procède généralement ainsi lorsque la méthode de remplacement des chaînes est définie sur `"automatically at runtime"` ou `"via ActionScript at runtime"`.

**Disponibilité :** ActionScript 2.0 ; Flash Player 7

### Paramètres

*`langCode`*`:String` - Le code de langue.

*`path`*`:String` - Le chemin du fichier XML à ajouter.

### Exemple

L'exemple suivant utilise la méthode `setInterval()` afin de vérifier si le fichier de langue XML a été chargé avec succès.

```
import mx.lang.Locale;
Locale.setLoadCallback(localeCallback);
Locale.loadLanguageXML("en");
// create interval to check if language XML file is loaded
var locale_int:Number = setInterval(checkLocaleStatus, 10);
function checkLocaleStatus():Void {
  if (Locale.checkXMLStatus()) {
    clearInterval(locale_int);
    trace("clearing interval @ " + getTimer() + " ms");
  }
}
// callback function for Locale.setLoadCallback()
function localeCallback(success:Boolean):Void {
  greeting_txt.text = Locale.loadString("IDS_GREETING");
}
```

# autoReplace (propriété Locale.autoReplace)

`statique publique autoReplace : Boolean`

Détermine si les chaînes sont remplacées automatiquement une fois le chargement du fichier xml terminé. Si la valeur est `true`, la méthode de remplacement du texte est équivalente au paramètre `"automatically at runtime"` du panneau Chaînes. Cela signifie que Flash Player détermine la langue par défaut de l'environnement hôte et affiche automatiquement le texte dans cette langue. Si la valeur est `false`, la méthode de remplacement du texte est équivalente au paramètre `"via ActionScript at runtime"` du panneau Chaînes. Cela signifie que vous êtes responsable du chargement du fichier XML approprié pour afficher le texte.

La valeur par défaut de cette propriété reflète le paramètre sélectionné pour Remplacer les chaînes dans la boîte de dialogue du panneau Chaînes : `true` pour `"automatically at runtime"` (le paramètre par défaut) et `false` pour « via ActionScript lors de l'exécution ».

**Disponibilité :** ActionScript 2.0 ; Flash Player 8

## Exemple

L'exemple suivant utilise la propriété `Locale.autoReplace` pour remplir le champ de texte `greeting_txt` créé dynamiquement sur la scène avec le contenu de la chaîne `IDS_GREETING` du fichier XML en anglais. Dans le panneau Chaînes, cliquez sur le bouton Paramètres pour ouvrir la boîte de dialogue Paramètres. Lorsqu'elle est ouverte, vous pouvez ajouter deux langues actives : Anglais (en) et Français (fr). Définissez ensuite l'option de remplacement des chaînes sur `"via ActionScript at runtime"`, puis cliquez sur OK. Enfin, entrez l'ID de chaîne de *IDS_GREETING* dans le panneau Chaînes et ajoutez du texte pour chaque langue active.

```
import mx.lang.Locale;
this.createTextField("greeting_txt", 10, 40, 40, 200, 20);
greeting_txt.autoSize = "left";
Locale.autoReplace = true;
Locale.addDelayedInstance(greeting_txt, "IDS_GREETING");
Locale.loadLanguageXML("en");
```

# checkXMLStatus (méthode Locale.checkXMLStatus)

`statique publique checkXMLStatus() : Boolean`

Renvoie `true` si le fichier xml est chargé ; `false` sinon.

**Disponibilité :** ActionScript 2.0 ; Flash Player 7

### Renvoie

`Boolean` - Renvoie `true` si le fichier XML est chargé ; `false` sinon.

### Exemple

L'exemple suivant utilise un intervalle de 10 millisecondes afin de vérifier si le fichier de langue a été chargé avec succès. Une fois le fichier XML chargé, l'occurrence de champ de texte `greeting_txt` présente sur la scène est renseignée par la chaîne `IDS_GREETING` du fichier de langue XML.

```
import mx.lang.Locale;
Locale.setLoadCallback(localeCallback);
Locale.loadLanguageXML("en");
// create interval to check if language XML file is loaded
var locale_int:Number = setInterval(checkLocaleStatus, 10);
function checkLocaleStatus():Void {
  if (Locale.checkXMLStatus()) {
    clearInterval(locale_int);
    trace("clearing interval @ " + getTimer() + " ms");
  }
}
// callback function for Locale.setLoadCallback()
function localeCallback(success:Boolean):Void {
  greeting_txt.text = Locale.loadString("IDS_GREETING");
}
```

## getDefaultLang (méthode Locale.getDefaultLang)

`statique publique getDefaultLang() : String`

Le code de langue par défaut tel que défini dans la boîte de dialogue Paramètres du panneau Chaînes ou en appelant la méthode `setDefaultLang()`.

**Disponibilité :** ActionScript 2.0 ; Flash Player 8

### Renvoie

`String` - Renvoie le code de langue par défaut.

### Exemple

L'exemple suivant crée une variable intitulée `defLang`, utilisée pour conserver la langue par défaut initiale du document Flash. Dans le panneau Chaînes, cliquez sur le bouton Paramètres pour ouvrir la boîte de dialogue Paramètres. Ensuite, ajoutez deux langues actives : Anglais (en) et Français (fr), définissez le bouton radio de remplacement des chaînes sur `"via ActionScript at runtime"`, puis cliquez sur OK. Dans le panneau Chaînes, ajoutez l'ID de chaîne de *IDS_GREETING*, puis ajoutez du texte pour chaque langue active.

```
import mx.lang.Locale;
var defLang:String = "fr";
Locale.setDefaultLang(defLang);
Locale.setLoadCallback(localeCallback);
Locale.loadLanguageXML(Locale.getDefaultLang());
function localeCallback(success:Boolean) {
  if (success) {
    trace(Locale.stringIDArray); // IDS_GREETING
    trace(Locale.loadString("IDS_GREETING"));
  } else {
    trace("unable to load XML");
  }
}
```

### Voir également

`setDefaultLang (méthode Locale.setDefaultLang)`

## initialize (méthode Locale.initialize)

`statique publique initialize() : Void`

Détermine automatiquement la langue à utiliser et charge le dossier de langue XML. Flash procède généralement ainsi lorsque la méthode de remplacement des chaînes est définie sur `"automatically at runtime"`.

**Disponibilité :** ActionScript 2.0 ; Flash Player 7

### Exemple

L'exemple suivant indique comment utiliser la méthode `initialize()` pour renseigner automatiquement le champ de texte `greeting_txt` présent sur la scène avec la langue du système d'exploitation actuellement utilisée par l'utilisateur. Au lieu d'utiliser directement la méthode `initialize()`, utilisez la méthode de remplacement des chaînes définie sur `"automatically at runtime"`.

```
import mx.lang.Locale;
trace(System.capabilities.language);
Locale.autoReplace = true;
Locale.addDelayedInstance(greeting_txt, "IDS_GREETING");
Locale.initialize();
```

# languageCodeArray (propriété Locale.languageCodeArray)

```
statique publique languageCodeArray : Array [lecture seule]
```

Un tableau contenant les codes de langue des langues ayant été spécifiées ou chargées dans le fichier FLA. Les codes de langue ne sont pas triés par ordre alphabétique.

**Disponibilité :** ActionScript 2.0 ; Flash Player 8

## Exemple

L'exemple suivant charge un fichier de langue XML en fonction de la valeur actuelle d'un composant ComboBox. Faites glisser un composant ComboBox sur la scène et donnez-lui le nom d'occurrence `lang_cb`. A l'aide de l'outil Texte, créez un champ de texte dynamique et donnez-lui le nom d'occurrence `greeting_txt`. Dans le panneau Chaînes, ajoutez au moins deux langues actives, définissez l'option de remplacement des chaînes sur `"via ActionScript at runtime"`, puis cliquez sur OK. Ensuite, ajoutez l'ID de chaîne de *IDS_GREETING*, puis entrez du texte pour chaque langue active. Enfin, ajoutez le code ActionScript suivant à l'image 1 du scénario principal :

```
import mx.lang.Locale;
Locale.setLoadCallback(localeListener);
lang_cb.dataProvider = Locale.languageCodeArray.sort();
lang_cb.addEventListener("change", langListener);

function langListener(eventObj:Object):Void {
  Locale.loadLanguageXML(eventObj.target.value);
}
function localeListener(success:Boolean):Void {
  if (success) {
    greeting_txt.text = Locale.loadString("IDS_GREETING");
  } else {
    greeting_txt.text = "unable to load language XML file.";
  }
}
```

# loadLanguageXML (méthode Locale.loadLanguageXML)

```
statique publique loadLanguageXML(xmlLanguageCode:String,
  customXmlCompleteCallback:Function) : Void
```

Charge le dossier de langue XML spécifié.

**Disponibilité :** ActionScript 2.0 ; Flash Player 8

### Paramètres

*xmlLanguageCode*:String - Le code de langue du fichier de langue XML que vous souhaitez charger.

*customXmlCompleteCallback*:Function - La fonction de rappel personalisée à appeler lorsque le fichier de langue XML est chargé.

### Exemple

L'exemple suivant utilise la méthode `loadLanguageXML()` pour charger le fichier de langue XML Anglais (en). Une fois le fichier de langue chargé, la méthode `localeCallback()` est appelée et renseigne le champ de texte `greeting_txt` présent sur la scène avec le contenu de la chaîne `IDS_GREETING` du fichier XML.

```
import mx.lang.Locale;
Locale.setLoadCallback(localeCallback);
Locale.loadLanguageXML("en");
// create interval to check if language XML file is loaded
var locale_int:Number = setInterval(checkLocaleStatus, 10);
function checkLocaleStatus():Void {
  if (Locale.checkXMLStatus()) {
    clearInterval(locale_int);
    trace("clearing interval @ " + getTimer() + " ms");
  }
}
// callback function for Locale.setLoadCallback()
function localeCallback(success:Boolean):Void {
  greeting_txt.text = Locale.loadString("IDS_GREETING");
}
```

## loadString (méthode Locale.loadString)

```
statique publique loadString(id:String) : String
```

Renvoie la valeur de chaîne associée à l'ID de chaîne spécifié dans la langue actuellement utilisée.

**Disponibilité :** ActionScript 2.0 ; Flash Player 7

### Paramètres

*id*:String - Le numéro d'identification (ID) de la chaîne à charger.

### Renvoie

String - La valeur de chaîne associée à l'ID de chaîne spécifié dans la langue actuellement utilisée.

### Exemple

L'exemple suivant utilise un intervalle de 10 millisecondes afin de vérifier si le fichier de langue a été chargé avec succès. Une fois le fichier XML chargé, l'occurrence de champ de texte `greeting_txt` présente sur la scène est renseignée par la chaîne `IDS_GREETING` du fichier de langue XML.

```
import mx.lang.Locale;
Locale.setLoadCallback(localeCallback);
Locale.loadLanguageXML("en");
// create interval to check if language XML file is loaded
var locale_int:Number = setInterval(checkLocaleStatus, 10);
function checkLocaleStatus():Void {
  if (Locale.checkXMLStatus()) {
    clearInterval(locale_int);
    trace("clearing interval @ " + getTimer() + " ms");
  }
}
// callback function for Locale.setLoadCallback()
function localeCallback(success:Boolean):Void {
  greeting_txt.text = Locale.loadString("IDS_GREETING");
}
```

### Voir également

`loadStringEx (méthode Locale.loadStringEx)`

## loadStringEx (méthode Locale.loadStringEx)

```
statique publique loadStringEx(stringID:String, languageCode:String) :
  String
```

Renvoie la valeur de chaîne associée à l'ID de chaîne et au code de langue spécifiés. Pour éviter le chargement inattendu du fichier XML, `loadStringEx()` ne charge pas le fichier de langue XML si le fichier XML n'est pas déjà chargé. Vous devez décider d'appeler la méthode `loadLanguageXML()` au moment opportun si vous souhaitez charger un fichier de langue XML.

**Disponibilité :** ActionScript 2.0 ; Flash Player 8

### Paramètres

*`stringID`*`:String` - Le numéro d'identification (ID) de la chaîne à charger.

*`languageCode`*`:String` - Le code de langue.

### Renvoie

`String` - La valeur de chaîne associée à l'ID de chaîne spécifié dans la langue spécifiée par le paramètre `languageCode`.

### Exemple

L'exemple suivant utilise la méthode `loadStringEx()` pour analyser la valeur de la chaîne `IDS_GREETING` du fichier de langue XML Français actuellement chargé.

```
import mx.lang.Locale;
Locale.setLoadCallback(localeCallback);
Locale.loadLanguageXML("fr");
function localeCallback(success:Boolean) {
  trace(success);
  trace(Locale.stringIDArray); // IDS_GREETING
  trace(Locale.loadStringEx("IDS_GREETING", "fr")); // bonjour
}
```

### Voir également

`loadString (méthode Locale.loadString)`

## setDefaultLang (méthode Locale.setDefaultLang)

`statique publique setDefaultLang(langCode:String) : Void`

Définit le code de langue par défaut.

**Disponibilité :** ActionScript 2.0 ; Flash Player 7

### Paramètres

*`langCode`*`:String` - Une chaîne représentant un code de langue.

### Exemple

L'exemple suivant crée une variable intitulée `defLang`, utilisée pour conserver la langue par défaut initiale du document Flash. Dans le panneau Chaînes, cliquez sur le bouton Paramètres pour ouvrir la boîte de dialogue Paramètres. Ensuite, ajoutez deux langues actives : Anglais (en) et Français (fr), définissez le bouton radio de remplacement des chaînes sur `"via ActionScript at runtime"`, puis cliquez sur OK. Dans le panneau Chaînes, ajoutez l'ID de chaîne de *IDS_GREETING*, puis ajoutez du texte pour chaque langue active.

```
import mx.lang.Locale;
var defLang:String = "fr";
Locale.setDefaultLang(defLang);
Locale.setLoadCallback(localeCallback);
Locale.loadLanguageXML(Locale.getDefaultLang());
function localeCallback(success:Boolean) {
```

```
  if (success) {
    trace(Locale.stringIDArray); // IDS_GREETING
    trace(Locale.loadString("IDS_GREETING"));
  } else {
    trace("unable to load XML");
  }
}
```

### Voir également

getDefaultLang (méthode Locale.getDefaultLang)

## setLoadCallback (méthode Locale.setLoadCallback)

`statique publique setLoadCallback(loadCallback:Function) : Void`

Définit la fonction de rappel qui sera appelée une fois le fichier XML chargé.

**Disponibilité :** ActionScript 2.0 ; Flash Player 7

### Paramètres

*`loadCallback`*`:Function` - La fonction à appeler lorsque le fichier de langue XML est chargé.

### Exemple

L'exemple suivant utilise un intervalle de 10 millisecondes afin de vérifier si le fichier de langue a été chargé avec succès. Une fois le fichier XML chargé, l'occurrence de champ de texte `greeting_txt` présente sur la scène est renseignée par la chaîne `IDS_GREETING` du fichier de langue XML.

```
import mx.lang.Locale;
Locale.setLoadCallback(localeCallback);
Locale.loadLanguageXML("en");
// create interval to check if language XML file is loaded
var locale_int:Number = setInterval(checkLocaleStatus, 10);
function checkLocaleStatus():Void {
  if (Locale.checkXMLStatus()) {
    clearInterval(locale_int);
    trace("clearing interval @ " + getTimer() + " ms");
  }
}
// callback function for Locale.setLoadCallback()
function localeCallback(success:Boolean):Void {
  greeting_txt.text = Locale.loadString("IDS_GREETING");
}
```

# setString (méthode Locale.setString)

```
statique publique setString(stringID:String, languageCode:String,
  stringValue:String) : Void
```

Définit la nouvelle valeur de chaîne d'un ID de chaîne et d'un code de langue spécifiés.

**Disponibilité :** ActionScript 2.0 ; Flash Player 8

## Paramètres

*stringID*:String - Le numéro d'identification (ID) de la chaîne à définir.

*languageCode*:String - Le code de langue.

*stringValue*:String - Une valeur de chaîne.

## Exemple

L'exemple suivant utilise la méthode `setString()` pour définir la chaîne `IDS_WELCOME` des langues Anglais (en) et Français (fr).

```
import mx.lang.Locale;
Locale.setString("IDS_WELCOME", "en", "hello");
Locale.setString("IDS_WELCOME", "fr", "bonjour");
trace(Locale.loadStringEx("IDS_WELCOME", "en")); // hello
```

# stringIDArray (propriété Locale.stringIDArray)

```
statique publique stringIDArray : Array [lecture seule]
```

Un tableau contenant tous les ID de chaîne du fichier FLA. Les ID de chaîne ne sont pas triés par ordre alphabétique.

**Disponibilité :** ActionScript 2.0 ; Flash Player 8

## Exemple

L'exemple suivant présente la propriété `Locale.stringIDArray` du fichier de langue XML actuellement chargé. Dans le panneau Chaînes, cliquez sur le bouton Paramètres pour ouvrir la boîte de dialogue Paramètres. Ensuite, ajoutez deux langues actives : Anglais (en) et Français (fr), définissez le bouton radio de remplacement des chaînes sur `"via ActionScript at runtime"`, puis cliquez sur OK. Dans le panneau Chaînes, ajoutez l'ID de chaîne de *IDS_GREETING*, puis ajoutez du texte pour chaque langue active.

```
import mx.lang.Locale;
Locale.setLoadCallback(localeCallback);
Locale.loadLanguageXML("fr");
function localeCallback(success:Boolean) {
  trace(success);
  trace(Locale.stringIDArray); // IDS_GREETING
```

```
  trace(Locale.loadStringEx("IDS_GREETING", "fr")); // bonjour
}
```

# Math

```
Object
  |
  +-Math
```

```
public class Math
extends Object
```

La classe Math est une classe de niveau supérieur dont vous pouvez utiliser les méthodes et les propriétés sans l'aide d'un constructeur.

Utilisez les méthodes et les propriétés de cette classe pour accéder aux constantes et fonctions mathématiques et les manipuler. Toutes les propriétés et les méthodes de la classe Math sont statiques et doivent être appelées à l'aide de la syntaxe `Math.method( parameter )` ou `Math.constant`. Dans ActionScript, les constantes sont définies selon la précision maximale des nombres à virgule flottante comportant deux décimales conformément à IEEE-754.

Plusieurs méthodes de la classe Math utilisent la mesure d'un angle en radians en tant que paramètre. Vous pouvez utiliser l'équation suivante pour calculer les valeurs radian avant d'appeler la méthode, puis exprimer la valeur calculée en tant que paramètre. Vous pouvez également utiliser toutes les valeurs situées à droite de l'équation (avec la mesure de l'angle exprimée en radians, à la place des `degrees`) en tant que paramètre radian.

Pour calculer une valeur radian, utilisez la formule suivante :

```
radians = degrees * Math.PI/180
```

Dans l'exemple suivant, l'équation est utilisée en tant que paramètre pour calculer le sinus d'un angle de 45 ° :

`Math.sin(45 * Math.PI/180)` est identique à `Math.sin(.7854)`

**Disponibilité :** ActionScript 1,0 ; Flash Player 5

## Résumé des propriétés

| Modificateurs | Propriété | Description |
|---|---|---|
| `static` | `E:Number` | Constante mathématique pour la base des logarithmes népériens, exprimée en *e*. |
| `static` | `LN10:Number` | Constante mathématique pour le logarithme népérien de 10, exprimée sous la forme de loge10, d'une valeur approximative de 2,302585092994046. |
| `static` | `LN2:Number` | Constante mathématique pour le logarithme népérien de 2, exprimée sous la forme de loge2, d'une valeur approximative de 0,6931471805599453. |
| `static` | `LOG10E:Number` | Constante mathématique pour le logarithme en base 10 de la constante e (`Math.E`), exprimée sous la forme de log10e, d'une valeur approximative de 0,4342944819032518. |
| `static` | `LOG2E:Number` | Constante mathématique pour le logarithme en base 2 de la constante e (`Math.E`), exprimée sous la forme de log2e, d'une valeur approximative de 1,442695040888963387. |
| `static` | `PI:Number` | Constante mathématique pour le ratio de la circonférence d'un cercle par rapport à son diamètre, exprimée sous la forme de pi, d'une valeur de 3,141592653589793. |
| `static` | `SQRT1_2:Number` | Constante mathématique pour la racine carrée de un demi, d'une valeur approximative de 0,7071067811865476. |
| `static` | `SQRT2:Number` | Constante mathématique pour la racine carrée de 2, d'une valeur approximative de 1,4142135623730951. |

*Propriétés héritées de la classe Object*

```
constructeur (propriété Object.constructor), __proto__ (Object.__proto__
property), prototype (Object.prototype, propriété), __resolve
(Object.__resolve, propriété)
```

## Résumé de la méthode

| Modificateurs | Signature | Description |
|---|---|---|
| static | abs(x:Number) : Number | Calcule et renvoie une valeur absolue pour le nombre spécifié par le paramètre x. |
| static | acos(x:Number) : Number | Calcule et renvoie l'arc cosinus du nombre spécifié dans le paramètre x, en radians. |
| static | asin(x:Number) : Number | Calcule et renvoie l'arc sinus du nombre spécifié dans le paramètre x, en radians. |
| static | atan(tangent:Number) : Number | Calcule et renvoie la valeur, en radians, de l'angle dont la tangente est spécifiée dans le paramètre tangent. |
| static | atan2(y:Number, x:Number) : Number | Calcule et renvoie l'angle du point y/x en radians, lorsqu'il est mesuré dans le sens inverse des aiguilles d'une montre à partir de l'axe *x* d'un cercle (où 0,0 représente le centre du cercle). |
| static | ceil(x:Number) : Number | Renvoie la valeur maximale du nombre ou de l'expression spécifié(e). |
| static | cos(x:Number) : Number | Calcule et renvoie le cosinus de l'angle spécifié en radians. |
| static | exp(x:Number) : Number | Renvoie la valeur de la base du logarithme népérien (*e*), à la puissance de l'exposant spécifié dans le paramètre x. |
| static | floor(x:Number) : Number | Renvoie la valeur minimale du nombre ou de l'expression spécifié(e) dans le paramètre x. |
| static | log(x:Number) : Number | Renvoie le logarithme népérien du paramètre x. |
| static | max(x:Number, y:Number) : Number | Evalue x et y, puis renvoie la valeur la plus élevée. |
| static | min(x:Number, y:Number) : Number | Evalue x et y, puis renvoie la valeur la plus faible. |
| static | pow(x:Number, y:Number) : Number | Calcule et renvoie x à la puissance de y. |
| static | random() : Number | Renvoie un nombre pseudo-aléatoire n, où 0 <= n < 1. |
| static | round(x:Number) : Number | Arrondit la valeur du paramètre x à l'entier immédiatement supérieur ou inférieur et renvoie la valeur. |
| static | sin(x:Number) : Number | Calcule et renvoie le sinus de l'angle spécifié en radians. |

| Modificateurs | Signature | Description |
|---|---|---|
| static | sqrt(x:Number) : Number | Calcule et renvoie la racine carrée du nombre spécifié. |
| static | tan(x:Number) : Number | Calcule et renvoie la tangente de l'angle spécifié. |

*Méthodes héritées de la classe Object*

addProperty (méthode Object.addProperty), hasOwnProperty (méthode Object.hasOwnProperty), isPropertyEnumerable (méthode Object.isPropertyEnumerable), isPrototypeOf (méthode Object.isPrototypeOf), registerClass (méthode Object.registerClass), toString (méthode Object.toString), unwatch (méthode Object.unwatch), valueOf (méthode Object.valueOf), watch (méthode Object.watch)

# abs (méthode Math.abs)

statique publique abs(x:Number) : Number

Calcule et renvoie une valeur absolue pour le nombre spécifié par le paramètre x.

**Disponibilité :** ActionScript 1,0 ; Flash Player 5

## Paramètres

*x*:Number - Un nombre.

## Renvoie

Number - Un nombre.

## Exemple

L'exemple suivant illustre la façon dont la méthode Math.abs() renvoie la valeur absolue d'un nombre sans affecter la valeur du paramètre x (intitulé num dans cet exemple) :

```
var num:Number = -12;
var numAbsolute:Number = Math.abs(num);
trace(num); // output: -12
trace(numAbsolute); // output: 12
```

# acos (méthode Math.acos)

statique publique acos(x:Number) : Number

Calcule et renvoie l'arc cosinus du nombre spécifié dans le paramètre x, en radians.

**Disponibilité :** ActionScript 1,0 ; Flash Player 5

### Paramètres

*x*:Number - Un nombre compris entre -1,0 et 1,0.

### Renvoie

Number - Un nombre ; l'arc cosinus du paramètre x.

### Exemple

L'exemple suivant affiche l'arc cosinus pour plusieurs valeurs.

```
trace(Math.acos(-1)); // output: 3.14159265358979
trace(Math.acos(0)); // output: 1.5707963267949
trace(Math.acos(1)); // output: 0
```

### Voir également

asin (méthode Math.asin), atan (méthode Math.atan), atan2 (méthode Math.atan2), cos (méthode Math.cos), sin (méthode Math.sin), tan (méthode Math.tan)

## asin (méthode Math.asin)

```
statique publique asin(x:Number) : Number
```

Calcule et renvoie l'arc sinus du nombre spécifié dans le paramètre x, en radians.

**Disponibilité :** ActionScript 1.0 ; Flash Player 5

### Paramètres

*x*:Number - Un nombre compris entre -1,0 et 1,0.

### Renvoie

Number - Un nombre entre pi négatif divisé par 2 et pi positif divisé par 2.

### Exemple

L'exemple suivant affiche l'arc sinus pour plusieurs valeurs.

```
trace(Math.asin(-1)); // output: -1.5707963267949
trace(Math.asin(0)); // output: 0
trace(Math.asin(1)); // output: 1.5707963267949
```

### Voir également

acos (méthode Math.acos), atan (méthode Math.atan), atan2 (méthode Math.atan2), cos (méthode Math.cos), sin (méthode Math.sin), tan (méthode Math.tan)

## atan (méthode Math.atan)

```
statique publique atan(tangent:Number) : Number
```

Calcule et renvoie la valeur, en radians, de l'angle dont la tangente est spécifiée dans le paramètre `tangent`. La valeur renvoyée est comprise entre pi négatif divisé par 2 et pi positif divisé par 2.

**Disponibilité :** ActionScript 1.0 ; Flash Player 5

### Paramètres

*`tangent`*`:Number` - Un nombre représentant la tangente d'un angle.

### Renvoie

`Number` - Un nombre entre pi négatif divisé par 2 et pi positif divisé par 2.

### Exemple

L'exemple suivant affiche la valeur d'angle de plusieurs tangentes.

```
trace(Math.atan(-1)); // output: -0.785398163397448
trace(Math.atan(0)); // output: 0
trace(Math.atan(1)); // output: 0.785398163397448
```

### Voir également

acos (méthode Math.acos), asin (méthode Math.asin), atan2 (méthode Math.atan2), cos (méthode Math.cos), sin (méthode Math.sin), tan (méthode Math.tan)

## atan2 (méthode Math.atan2)

```
statique publique atan2(y:Number, x:Number) : Number
```

Calcule et renvoie l'angle du point `y/x` en radians, lorsqu'il est mesuré dans le sens inverse des aiguilles d'une montre à partir de l'axe *x* d'un cercle (où 0,0 représente le centre du cercle). La valeur renvoyée est comprise entre pi positif et pi négatif.

**Disponibilité :** ActionScript 1.0 ; Flash Player 5

### Paramètres

*y*:Number - Un nombre spécifiant la coordonnée *y* du point.

*x*:Number - Un nombre spécifiant la coordonnée *x* du point.

### Renvoie

Number - Un nombre.

### Exemple

L'exemple suivant renvoie l'angle, en radians, du point spécifié par les coordonnées (0, 10), telles que x = 0 et y = 10. Notez que le premier paramètre attribué à atan2 est toujours la coordonnée y.

```
trace(Math.atan2(10, 0)); // output: 1.5707963267949
```

### Voir également

acos (méthode Math.acos), asin (méthode Math.asin), atan (méthode Math.atan), cos (méthode Math.cos), sin (méthode Math.sin), tan (méthode Math.tan)

## ceil (méthode Math.ceil)

```
statique publique ceil(x:Number) : Number
```

Renvoie la valeur maximale du nombre ou de l'expression spécifié(e). La valeur maximale d'un nombre est l'entier le plus proche supérieur ou égal au nombre.

**Disponibilité :** ActionScript 1.0 ; Flash Player 5

### Paramètres

*x*:Number - Un nombre ou une expression.

### Renvoie

Number - Un entier le plus proche et supérieur ou égal au paramètre x.

### Exemple

Le code suivant renvoie une valeur de 13 :

```
Math.ceil(12.5);
```

### Voir également

floor (méthode Math.floor), round (méthode Math.round)

# cos (méthode Math.cos)

```
statique publique cos(x:Number) : Number
```

Calcule et renvoie le cosinus de l'angle spécifié en radians. Pour calculer un radian, consultez la description de l'entrée de la classe Math.

**Disponibilité :** ActionScript 1.0 ; Flash Player 5

## Paramètres

*x*:Number - Un nombre représentant un angle mesuré en radians.

## Renvoie

Number - Un nombre compris entre -1,0 et 1,0.

## Exemple

L'exemple suivant affiche le cosinus de plusieurs angles différents.

```
trace (Math.cos(0)); // 0 degree angle. Output: 1
trace (Math.cos(Math.PI/2)); // 90 degree angle. Output: 6.12303176911189e-
  17
trace (Math.cos(Math.PI)); // 180 degree angle. Output: -1
trace (Math.cos(Math.PI*2)); // 360 degree angle. Output: 1
```

*Remarque :* Le cosinus d'un angle à 90 degrés est zéro, mais en raison de l'inexactitude inhérente des calculs décimaux intégrant des nombres binaires, Flash Player renvoie un nombre le plus proche de zéro, mais pas égal à zéro.

## Voir également

acos (méthode Math.acos), asin (méthode Math.asin), atan (méthode Math.atan), atan2 (méthode Math.atan2), sin (méthode Math.sin), tan (méthode Math.tan)

# E (propriété Math.E)

```
statique publique E : Number
```

Constante mathématique pour la base des logarithmes népériens, exprimée en *e*. La valeur approximative de *e* est 2,71828182845905.

**Disponibilité :** ActionScript 1.0 ; Flash Player 5

## Exemple

Cet exemple illustre l'utilisation de Math.E pour calculer les intérêts composés de façon continue d'un cas simple portant sur un intérêt à 100 % sur un an.

```
var principal:Number = 100;
```

```
var simpleInterest:Number = 100;
var continuouslyCompoundedInterest:Number = (100 * Math.E) - principal;

trace ("Beginning principal: $" + principal);
trace ("Simple interest after one year: $" + simpleInterest);
trace ("Continuously compounded interest after one year: $" +
  continuouslyCompoundedInterest);

//
Output:
Beginning principal: $100
Simple interest after one year: $100
Continuously compounded interest after one year: $171.828182845905
```

# exp (méthode Math.exp)

```
statique publique exp(x:Number) : Number
```

Renvoie la valeur de la base du logarithme népérien (*e*), à la puissance de l'exposant spécifié dans le paramètre `x`. La constante `Math.E` peut renvoyer la valeur *e*.

**Disponibilité :** ActionScript 1.0 ; Flash Player 5

## Paramètres

*x*`:Number` - L'exposant ; un nombre ou une expression.

## Renvoie

`Number` - Un nombre.

## Exemple

L'exemple suivant affiche le logarithme de deux valeurs décimales.

```
trace(Math.exp(1)); // output: 2.71828182845905
trace(Math.exp(2)); // output: 7.38905609893065
```

## Voir également

`E (propriété Math.E)`

# floor (méthode Math.floor)

```
statique publique floor(x:Number) : Number
```

Renvoie la valeur minimale du nombre ou de l'expression spécifié(e) dans le paramètre `x`. La valeur minimale est l'entier le plus proche inférieur ou égal au nombre ou à l'expression spécifié(e).

**Disponibilité :** ActionScript 1.0 ; Flash Player 5

### Paramètres

*x*:Number - Un nombre ou une expression.

### Renvoie

Number - L'entier le plus proche et inférieur ou égal au paramètre x.

### Exemple

Le code suivant renvoie une valeur de 12 :

```
Math.floor(12.5);
```

Le code suivant renvoie une valeur de -7 :

```
Math.floor(-6.5);
```

## LN10 (propriété Math.LN10)

```
statique publique LN10 : Number
```

Constante mathématique pour le logarithme népérien de 10, exprimée sous la forme de loge10, d'une valeur approximative de 2,302585092994046.

**Disponibilité :** ActionScript 1.0 ; Flash Player 5

### Exemple

Cet exemple présente la valeur de Math.LN10.

```
trace(Math.LN10);
// output: 2.30258509299405
```

## LN2 (propriété Math.LN2)

```
statique publique LN2 : Number
```

Constante mathématique pour le logarithme népérien de 2, exprimée sous la forme de loge2, d'une valeur approximative de 0,6931471805599453.

**Disponibilité :** ActionScript 1.0 ; Flash Player 5

## log (méthode Math.log)

```
statique publique log(x:Number) : Number
```

Renvoie le logarithme népérien du paramètre x.

**Disponibilité :** ActionScript 1.0 ; Flash Player 5

### Paramètres

*x*:Number - Un nombre ou une expression d'une valeur supérieure à 0.

### Renvoie

Number - Le logarithme népérien du paramètre x.

### Exemple

L'exemple suivant affiche le logarithme de trois valeurs numériques.

```
trace(Math.log(0)); // output: -Infinity
trace(Math.log(1)); // output: 0
trace(Math.log(2)); // output: 0.693147180559945
trace(Math.log(Math.E)); // output: 1
```

## LOG10E (propriété Math.LOG10E)

```
statique publique LOG10E : Number
```

Constante mathématique pour le logarithme en base 10 de la constante e (Math.E), exprimée sous la forme de log10e, d'une valeur approximative de 0,4342944819032518.

La méthode Math.log() calcule le logarithme népérien d'un nombre. Multipliez le résultat de Math.log() par Math.LOG10E pour obtenir le logarithme en base 10.

**Disponibilité :** ActionScript 1.0 ; Flash Player 5

### Exemple

Cet exemple indique comment obtenir le logarithme en base 10 d'un nombre :

```
trace(Math.log(1000) * Math.LOG10E);
// Output: 3
```

## LOG2E (propriété Math.LOG2E)

```
statique publique LOG2E : Number
```

Constante mathématique pour le logarithme en base 2 de la constante e (Math.E), exprimée sous la forme de log2e, d'une valeur approximative de 1,442695040888963387.

La méthode Math.log calcule le logarithme népérien d'un nombre. Multipliez le résultat de Math.log() par Math.LOG2E pour obtenir le logarithme en base 2.

**Disponibilité :** ActionScript 1.0 ; Flash Player 5

### Exemple

Cet exemple indique comment obtenir le logarithme en base 2 d'un nombre :

```
trace(Math.log(16) * Math.LOG2E);
// Output: 4
```

## max (méthode Math.max)

```
statique publique max(x:Number, y:Number) : Number
```

Evalue `x` et `y`, puis renvoie la valeur la plus élevée.

**Disponibilité :** ActionScript 1.0 ; Flash Player 5

### Paramètres

*x*`:Number` - Un nombre ou une expression.

*y*`:Number` - Un nombre ou une expression.

### Renvoie

`Number` - Un nombre.

### Exemple

L'exemple suivant affiche `Thu Dec 30 00:00:00 GMT-0700 2004`, soit l'expression la plus élevée parmi celles évaluées.

```
var date1:Date = new Date(2004, 11, 25);
var date2:Date = new Date(2004, 11, 30);
var maxDate:Number = Math.max(date1.getTime(), date2.getTime());
trace(new Date(maxDate).toString());
```

### Voir également

`min (méthode Math.min)`, `Date`

## min (méthode Math.min)

```
statique publique min(x:Number, y:Number) : Number
```

Evalue `x` et `y`, puis renvoie la valeur la plus faible.

**Disponibilité :** ActionScript 1.0 ; Flash Player 5

### Paramètres

*x*`:Number` - Un nombre ou une expression.

*y*`:Number` - Un nombre ou une expression.

### Renvoie

`Number` - Un nombre.

### Exemple

L'exemple suivant affiche `Sat Dec 25 00:00:00 GMT-0700 2004`, soit l'expression la plus faible parmi celles évaluées.

```
var date1:Date = new Date(2004, 11, 25);
var date2:Date = new Date(2004, 11, 30);
var minDate:Number = Math.min(date1.getTime(), date2.getTime());
trace(new Date(minDate).toString());
```

### Voir également

`max (méthode Math.max)`

## PI (propriété Math.PI)

```
statique publique PI : Number
```

Constante mathématique pour le ratio de la circonférence d'un cercle par rapport à son diamètre, exprimée sous la forme de pi, d'une valeur de 3,141592653589793.

**Disponibilité :** ActionScript 1.0 ; Flash Player 5

### Exemple

L'exemple suivant trace un cercle à l'aide de la constante mathématique pi et de l'API de dessin.

```
drawCircle(this, 100, 100, 50);
//
function drawCircle(mc:MovieClip, x:Number, y:Number, r:Number):Void {
  mc.lineStyle(2, 0xFF0000, 100);
  mc.moveTo(x+r, y);
  mc.curveTo(r+x, Math.tan(Math.PI/8)*r+y, Math.sin(Math.PI/4)*r+x,
  Math.sin(Math.PI/4)*r+y);
  mc.curveTo(Math.tan(Math.PI/8)*r+x, r+y, x, r+y);
  mc.curveTo(-Math.tan(Math.PI/8)*r+x, r+y, -Math.sin(Math.PI/4)*r+x,
  Math.sin(Math.PI/4)*r+y);
  mc.curveTo(-r+x, Math.tan(Math.PI/8)*r+y, -r+x, y);
  mc.curveTo(-r+x, -Math.tan(Math.PI/8)*r+y, -Math.sin(Math.PI/4)*r+x, -
  Math.sin(Math.PI/4)*r+y);
  mc.curveTo(-Math.tan(Math.PI/8)*r+x, -r+y, x, -r+y);
  mc.curveTo(Math.tan(Math.PI/8)*r+x, -r+y, Math.sin(Math.PI/4)*r+x, -
  Math.sin(Math.PI/4)*r+y);
  mc.curveTo(r+x, -Math.tan(Math.PI/8)*r+y, r+x, y);
}
```

# pow (méthode Math.pow)

```
statique publique pow(x:Number, y:Number) : Number
```

Calcule et renvoie `x` à la puissance de `y`.

**Disponibilité :** ActionScript 1.0 ; Flash Player 5

## Paramètres

*`x`*`:Number` - Un nombre à élever à une puissance.

*`y`*`:Number` - Un nombre spécifiant la puissance à laquelle le paramètre `x` est élevé.

## Renvoie

`Number` - Un nombre.

## Exemple

L'exemple suivant utilise `Math.pow` et `Math.sqrt` pour calculer la longueur d'une ligne.

```
this.createEmptyMovieClip("canvas_mc", this.getNextHighestDepth());
var mouseListener:Object = new Object();
mouseListener.onMouseDown = function() {
  this.origX = _xmouse;
  this.origY = _ymouse;
};
mouseListener.onMouseUp = function() {
  this.newX = _xmouse;
  this.newY = _ymouse;
  var minY = Math.min(this.origY, this.newY);
  var nextDepth:Number = canvas_mc.getNextHighestDepth();
  var line_mc:MovieClip =
  canvas_mc.createEmptyMovieClip("line"+nextDepth+"_mc", nextDepth);
  line_mc.moveTo(this.origX, this.origY);
  line_mc.lineStyle(2, 0x000000, 100);
  line_mc.lineTo(this.newX, this.newY);
  var hypLen:Number = Math.sqrt(Math.pow(line_mc._width,
  2)+Math.pow(line_mc._height, 2));
  line_mc.createTextField("length"+nextDepth+"_txt",
  canvas_mc.getNextHighestDepth(), this.origX, this.origY-22, 100, 22);
  line_mc['length'+nextDepth+'_txt'].text = Math.round(hypLen) +" pixels";
};
Mouse.addListener(mouseListener);
```

La méthode `MovieClip.getNextHighestDepth()` utilisée dans cet exemple nécessite Flash Player 7 ou une version ultérieure. Si votre fichier SWF inclut un composant de la version 2, utilisez la classe DepthManager des composants de la version 2 à la place de la méthode `MovieClip.getNextHighestDepth()`.

# random (méthode Math.random)

```
statique publique random() : Number
```

Renvoie un nombre pseudo-aléatoire n, où 0 <= n < 1. Le nombre renvoyé est un nombre pseudo-aléatoire car il n'est pas généré par un phénomène naturel parfaitement aléatoire telle qu'une désintégration radioactive.

**Disponibilité :** ActionScript 1.0 ; Flash Player 5

## Renvoie

`Number` - Un nombre.

## Exemple

L'exemple suivant renvoie 100 entiers aléatoires compris entre 4 et 11 (inclus) :

```
function randRange(min:Number, max:Number):Number {
  var randomNum:Number = Math.floor(Math.random() * (max - min + 1)) + min;
  return randomNum;
}
for (var i = 0; i < 100; i++) {
  var n:Number = randRange(4, 11)
  trace(n);
}
```

# round (méthode Math.round)

```
statique publique round(x:Number) : Number
```

Arrondit la valeur du paramètre `x` à l'entier immédiatement supérieur ou inférieur et renvoie la valeur. Si le paramètre `x` est équidistant de ses deux entiers les plus proches (si le nombre se termine par ,5), la valeur est arrondie à l'entier immédiatement supérieur.

**Disponibilité :** ActionScript 1.0 ; Flash Player 5

## Paramètres

*`x`*`:Number` - Un nombre.

## Renvoie

`Number` - Un nombre ; un entier.

## Exemple

L'exemple suivant renvoie un nombre aléatoire compris entre deux entiers spécifiés.

```
function randRange(min:Number, max:Number):Number {
```

```
  var randomNum:Number = Math.round(Math.random() * (max-min+1) + (min-
  .5));
  return randomNum;
}
for (var i = 0; i<25; i++) {
  trace(randRange(4, 11));
}
```

### Voir également

ceil (méthode Math.ceil), floor (méthode Math.floor)

## sin (méthode Math.sin)

```
statique publique sin(x:Number) : Number
```

Calcule et renvoie le sinus de l'angle spécifié en radians. Pour calculer un radian, consultez la description de l'entrée de la classe Math.

**Disponibilité :** ActionScript 1.0 ; Flash Player 5

### Paramètres

*x*:Number - Un nombre représentant un angle mesuré en radians.

### Renvoie

Number - Un nombre ; le sinus de l'angle spécifié (entre -1,0 et 1,0).

### Exemple

L'exemple suivant trace un cercle à l'aide de la constante mathématique pi, du sinus d'un angle et de l'API de dessin.

```
drawCircle(this, 100, 100, 50);
//
function drawCircle(mc:MovieClip, x:Number, y:Number, r:Number):Void {
  mc.lineStyle(2, 0xFF0000, 100);
  mc.moveTo(x+r, y);
  mc.curveTo(r+x, Math.tan(Math.PI/8)*r+y, Math.sin(Math.PI/4)*r+x,
  Math.sin(Math.PI/4)*r+y);
  mc.curveTo(Math.tan(Math.PI/8)*r+x, r+y, x, r+y);
  mc.curveTo(-Math.tan(Math.PI/8)*r+x, r+y, -Math.sin(Math.PI/4)*r+x,
  Math.sin(Math.PI/4)*r+y);
  mc.curveTo(-r+x, Math.tan(Math.PI/8)*r+y, -r+x, y);
  mc.curveTo(-r+x, -Math.tan(Math.PI/8)*r+y, -Math.sin(Math.PI/4)*r+x, -
  Math.sin(Math.PI/4)*r+y);
  mc.curveTo(-Math.tan(Math.PI/8)*r+x, -r+y, x, -r+y);
  mc.curveTo(Math.tan(Math.PI/8)*r+x, -r+y, Math.sin(Math.PI/4)*r+x, -
  Math.sin(Math.PI/4)*r+y);
```

```
  mc.curveTo(r+x, -Math.tan(Math.PI/8)*r+y, r+x, y);
}
```

### Voir également

acos (méthode Math.acos), asin (méthode Math.asin), atan (méthode Math.atan), atan2 (méthode Math.atan2), cos (méthode Math.cos), tan (méthode Math.tan)

## sqrt (méthode Math.sqrt)

```
statique publique sqrt(x:Number) : Number
```

Calcule et renvoie la racine carrée du nombre spécifié.

**Disponibilité :** ActionScript 1.0 ; Flash Player 5

### Paramètres

*x*:Number - Un nombre ou une expression supérieur(e) ou égal(e) à 0.

### Renvoie

Number - Un nombre si le paramètre *x* est supérieur ou égal à zéro ; NaN (pas un nombre) sinon.

### Exemple

L'exemple suivant utilise Math.pow et Math.sqrt pour calculer la longueur d'une ligne.

```
this.createEmptyMovieClip("canvas_mc", this.getNextHighestDepth());
var mouseListener:Object = new Object();
mouseListener.onMouseDown = function() {
  this.origX = _xmouse;
  this.origY = _ymouse;
};
mouseListener.onMouseUp = function() {
  this.newX = _xmouse;
  this.newY = _ymouse;
  var minY = Math.min(this.origY, this.newY);
  var nextDepth:Number = canvas_mc.getNextHighestDepth();
  var line_mc:MovieClip =
  canvas_mc.createEmptyMovieClip("line"+nextDepth+"_mc", nextDepth);
  line_mc.moveTo(this.origX, this.origY);
  line_mc.lineStyle(2, 0x000000, 100);
  line_mc.lineTo(this.newX, this.newY);
  var hypLen:Number = Math.sqrt(Math.pow(line_mc._width,
  2)+Math.pow(line_mc._height, 2));
  line_mc.createTextField("length"+nextDepth+"_txt",
  canvas_mc.getNextHighestDepth(), this.origX, this.origY-22, 100, 22);
  line_mc['length'+nextDepth+'_txt'].text = Math.round(hypLen) +" pixels";
```

```
};
Mouse.addListener(mouseListener);
```

# SQRT1_2 (propriété Math.SQRT1_2)

```
statique publique SQRT1_2 : Number
```

Constante mathématique pour la racine carrée de un demi, d'une valeur approximative de 0,7071067811865476.

**Disponibilité :** ActionScript 1.0 ; Flash Player 5

## Exemple

Cet exemple présente la valeur de `Math.SQRT1_2`.

```
trace(Math.SQRT1_2);
// Output: 0.707106781186548
```

# SQRT2 (propriété Math.SQRT2)

```
statique publique SQRT2 : Number
```

Constante mathématique pour la racine carrée de 2, d'une valeur approximative de 1,4142135623730951.

**Disponibilité :** ActionScript 1.0 ; Flash Player 5

## Exemple

Cet exemple présente la valeur de `Math.SQRT2`.

```
trace(Math.SQRT2);
// Output: 1.4142135623731
```

# tan (méthode Math.tan)

```
statique publique tan(x:Number) : Number
```

Calcule et renvoie la tangente de l'angle spécifié. Pour calculer un radian, suivez les informations qui figurent dans l'introduction à la classe Math.

**Disponibilité :** ActionScript 1.0 ; Flash Player 5

## Paramètres

*x*`:Number` - Un nombre représentant un angle mesuré en radians.

## Renvoie

`Number` - Un nombre ; la tangente du paramètre `x`.

### Exemple

L'exemple suivant trace un cercle à l'aide de la constante mathématique pi, de la tangente d'un angle et de l'API de dessin.

```
drawCircle(this, 100, 100, 50);
//
function drawCircle(mc:MovieClip, x:Number, y:Number, r:Number):Void {
  mc.lineStyle(2, 0xFF0000, 100);
  mc.moveTo(x+r, y);
  mc.curveTo(r+x, Math.tan(Math.PI/8)*r+y, Math.sin(Math.PI/4)*r+x,
  Math.sin(Math.PI/4)*r+y);
  mc.curveTo(Math.tan(Math.PI/8)*r+x, r+y, x, r+y);
  mc.curveTo(-Math.tan(Math.PI/8)*r+x, r+y, -Math.sin(Math.PI/4)*r+x,
  Math.sin(Math.PI/4)*r+y);
  mc.curveTo(-r+x, Math.tan(Math.PI/8)*r+y, -r+x, y);
  mc.curveTo(-r+x, -Math.tan(Math.PI/8)*r+y, -Math.sin(Math.PI/4)*r+x, -
  Math.sin(Math.PI/4)*r+y);
  mc.curveTo(-Math.tan(Math.PI/8)*r+x, -r+y, x, -r+y);
  mc.curveTo(Math.tan(Math.PI/8)*r+x, -r+y, Math.sin(Math.PI/4)*r+x, -
  Math.sin(Math.PI/4)*r+y);
  mc.curveTo(r+x, -Math.tan(Math.PI/8)*r+y, r+x, y);
}
```

### Voir également

acos (méthode Math.acos), asin (méthode Math.asin), atan (méthode Math.atan), atan2 (méthode Math.atan2), cos (méthode Math.cos), sin (méthode Math.sin)

# Matrix (flash.geom.Matrix)

```
Object
  |
  +-flash.geom.Matrix
```

```
public class Matrix
extends Object
```

La classe flash.geom.Matrix représente une matrice de transformation qui détermine la façon de mapper des points d'un espace de coordonnées à l'autre. Pour effectuer diverses transformations graphiques d'un objet, il vous suffit de définir les propriétés d'un objet Matrix et de l'appliquer à un objet MovieClip ou BitmapData. Ces fonctions de transformation incluent la translation (repositionnement de *x* et *y*), la rotation, le redimensionnement et l'inclinaison.

Associés, ces types de transformation sont connus sous le nom de *transformations affines*. Les transformations affines préservent la rectitude des lignes au cours de la transformation ; en outre, les lignes parallèles restent parallèles.

Pour appliquer une matrice de transformation à un clip, il vous suffit de créer un objet flash.geom.Transform et de définir sa propriété Matrix sur la matrice de transformation. Les objets Matrix sont également utilisés en tant que paramètres de certaines méthodes, telle que la méthode `draw()` de la classe flash.display.BitmapData.

Un objet de matrice de transformation est considéré comme étant une matrice 3 x 3 incluant le contenu suivant :

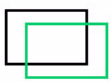
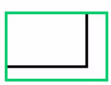
$$\begin{bmatrix} a & b & t_x \end{bmatrix}$$

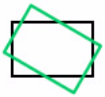
Dans le cas des matrices de transformation classiques, les propriétés `u`, `v` et `w` sont dotées de fonctionnalités supplémentaires. La classe Matrix fonctionne uniquement dans un espace bidimensionnel ; ainsi, elle suppose toujours que les valeurs des propriétés `u` et `v` sont 0,0, et que la valeur de la propriété `w` est 1,0. En d'autres termes, les valeurs effectives de la matrice sont les suivantes :

$$\begin{bmatrix} a & b & t_x \end{bmatrix}$$

Vous pouvez obtenir et définir les valeurs des six propriétés suivantes dans un objet Matrix : `a`, `b`, `c`, `d`, `tx` et `ty`.

La classe Matrix prend en charge les quatre principaux types de fonctions de transformation : la translation, le redimensionnement, la rotation et l'inclinaison. Trois de ces fonctions font appel à des méthodes spécialisées, tel que décrit dans le tableau ci-dessous.

| Transformation | Méthode | Valeurs de matrice | Résultat affiché | Description |
|---|---|---|---|---|
| Translation (déplacement) | `translate(tx, ty)` | $\begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix}$ |  | Déplace les pixels `tx` de l'image vers la droite et les pixels `ty` vers le bas. |
| Redimensionnement | `scale(sx, sy)` | $\begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix}$ |  | Redimensionne l'image, en multipliant l'emplacement de chaque pixel par `sx` sur l'axe *x* et par `sy` sur l'axe *y*. |

| Transformation | Méthode | Valeurs de matrice | Résultat affiché | Description |
|---|---|---|---|---|
| Rotation | `rotate(q)` | $\begin{bmatrix} \cos(q) & \sin(q) & 0 \\ -\sin(q) & \cos(q) & 0 \\ 0 & 0 & 1 \end{bmatrix}$ |  | Fait pivoter l'image selon un angle `q`, mesuré en radians |
| Inclinaison ou cisaillement | Aucune ; doit définir les propriétés `b` et `c`. | $\begin{bmatrix} 0 & sk_y & 0 \\ sk_x & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$ |  | Fait glisser l'image progressivement dans une direction parallèle à l'axe *x* ou *y*. La valeur `skx` fait office de multiplicateur contrôlant la distance de glissement le long de l'axe *x* ; la valeur `sky` contrôle la distance de glissement le long de l'axe *y*. |

Chaque fonction de transformation modifie les propriétés de matrice actuelles, ce qui vous permet d'associer effectivement plusieurs transformations. Pour ce faire, il vous suffit d'appeler plusieurs fonctions de transformation avant d'appliquer la matrice à son clip ou bitmap cible.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

## Voir également

`transform (propriété MovieClip.transform)`, `Transform (flash.geom.Transform)`, `draw (méthode BitmapData.draw)`, `a (propriété Matrix.a)`, `b (propriété Matrix.b)`, `c (propriété Matrix.c)`, `d (propriété Matrix.d)`, `tx (propriété Matrix.tx)`, `ty (propriété Matrix.ty)`, `translate (méthode Matrix.translate)`, `scale (méthode Matrix.scale)`, `rotate (méthode Matrix.rotate)`

## Résumé des propriétés

| Modificateurs | Propriété | Description |
|---|---|---|
| | `a:Number` | Dans la première ligne et la première colonne de l'objet Matrix, la valeur affectant le positionnement des pixels sur l'axe *x* lors du redimensionnement ou de la rotation d'une image. |
| | `b:Number` | Dans la première ligne et la deuxième colonne de l'objet Matrix, la valeur affectant le positionnement des pixels sur l'axe *y* lors de la rotation ou de l'inclinaison d'une image. |
| | `c:Number` | Dans la deuxième ligne et la première colonne de l'objet Matrix, la valeur affectant le positionnement des pixels sur l'axe *x* lors de la rotation ou de l'inclinaison d'une image. |
| | `d:Number` | Dans la deuxième ligne et la deuxième colonne de l'objet Matrix, la valeur affectant le positionnement des pixels sur l'axe *y* lors du redimensionnement ou de la rotation d'une image. |
| | `tx:Number` | La distance de translation de chaque point sur l'axe *x*. |
| | `ty:Number` | La distance de translation de chaque point sur l'axe *y*. |

*Propriétés héritées de la classe Object*

```
constructeur (propriété Object.constructor), __proto__ (Object.__proto__
property), prototype (Object.prototype, propriété), __resolve
(Object.__resolve, propriété)
```

## Résumé des constructeurs

| Signature | Description |
|---|---|
| `Matrix([a:Number], [b:Number], [c:Number], [d:Number], [tx:Number], [ty:Number])` | Crée un nouvel objet Matrix avec les paramètres spécifiés. |

Résumé de la méthode

| Modificateurs | Signature | Description |
|---|---|---|
| | `clone() : Matrix` | Renvoie un nouvel objet Matrix, clone de cette matrice, avec une copie exacte de l'objet contenu. |
| | `concat(m:Matrix) : Void` | Concatène une matrice avec la matrice actuelle, ce qui a pour effet de combiner les effets géométriques des deux matrices. |
| | `createBox(scaleX:Number, scaleY:Number, [rotation:Number], [tx:Number], [ty:Number]) : Void` | Inclut les paramètres de redimensionnement, de rotation et de translation. |
| | `createGradientBox(width:Number, height:Number, [rotation:Number], [tx:Number], [ty:Number]) : Void` | Crée le style de matrice attendu par la méthode `MovieClip.beginGradientFill()`. |
| | `deltaTransformPoint(pt:Point) : Point` | En partant d'un point dans l'espace de coordonnées de prétransformation, cette méthode renvoie les coordonnées de ce point suite à la transformation. |
| | `identity() : Void` | Définit chaque propriété de matrice sur une valeur qui rend un clip ou une construction géométrique transformé identique à l'original. |
| | `invert() : Void` | Effectue la transformation opposée de la matrice d'origine. |
| | `rotate(angle:Number) : Void` | Définit les valeurs dans la matrice actuelle de façon à ce qu'elle puisse être utilisée pour appliquer une transformation de rotation. |
| | `scale(sx:Number, sy:Number) : Void` | Modifie une matrice de façon à ce qu'elle redimensionne l'image à laquelle elle est appliquée. |
| | `toString() : String` | Renvoie une valeur de texte donnant la liste des propriétés de l'objet Matrix. |
| | `transformPoint(pt:Point) : Point` | Applique la transformation géométrique représentée par l'objet Matrix au point spécifié. |
| | `translate(tx:Number, ty:Number) : Void` | Modifie un objet Matrix de façon à ce que l'effet de sa transformation soit de déplacer un objet sur les axes *x* et *y*. |

addProperty (méthode Object.addProperty), hasOwnProperty (méthode Object.hasOwnProperty), isPropertyEnumerable (méthode Object.isPropertyEnumerable), isPrototypeOf (méthode Object.isPrototypeOf), registerClass (méthode Object.registerClass), toString (méthode Object.toString), unwatch (méthode Object.unwatch), valueOf (méthode Object.valueOf), watch (méthode Object.watch)

# a (propriété Matrix.a)

```
public a : Number
```

Dans la première ligne et la première colonne de l'objet Matrix, la valeur affectant le positionnement des pixels sur l'axe *x* lors du redimensionnement ou de la rotation d'une image.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

## Exemple

L'exemple suivant crée l'objet Matrix `myMatrix` et définit sa valeur `a`.

```
import flash.geom.Matrix;

var myMatrix:Matrix = new Matrix();
trace(myMatrix.a); // 1

myMatrix.a = 2;
trace(myMatrix.a); // 2
```

# b (propriété Matrix.b)

```
public b : Number
```

Dans la première ligne et la deuxième colonne de l'objet Matrix, la valeur affectant le positionnement des pixels sur l'axe *y* lors de la rotation ou de l'inclinaison d'une image.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

## Exemple

L'exemple suivant crée l'objet Matrix `myMatrix` et définit sa valeur `b`.

```
import flash.geom.Matrix;

var myMatrix:Matrix = new Matrix();
trace(myMatrix.b); // 0
```

```
var degrees:Number = 45;
var radians:Number = (degrees/180) Math.PI;
myMatrix.b = radians;
trace(myMatrix.b); // 0.785398163397448
```

# c (propriété Matrix.c)

```
public c : Number
```

Dans la deuxième ligne et la première colonne de l'objet Matrix, la valeur affectant le positionnement des pixels sur l'axe *x* lors de la rotation ou de l'inclinaison d'une image.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

## Exemple

L'exemple suivant crée l'objet Matrix `myMatrix` et définit sa valeur `c`.

```
import flash.geom.Matrix;

var myMatrix:Matrix = new Matrix();
trace(myMatrix.c); // 0

var degrees:Number = 45;
var radians:Number = (degrees/180) Math.PI;
myMatrix.c = radians;
trace(myMatrix.c); // 0.785398163397448
```

# clone (méthode Matrix.clone)

```
public clone() : Matrix
```

Renvoie un nouvel objet Matrix, clone de cette matrice, avec une copie exacte de l'objet contenu.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

## Renvoie

`flash.geom.Matrix` - Un objet Matrix.

## Exemple

L'exemple suivant crée la variable `clonedMatrix` à partir de la variable `myMatrix`. La classe Matrix ne dispose pas de méthode `equals` ; par conséquent, l'exemple suivant utilise une fonction écrite personnalisée pour tester l'égalité de deux matrices.

```
import flash.geom.Matrix;

var myMatrix:Matrix = new Matrix(2, 0, 0, 2, 0, 0);
```

```
var clonedMatrix:Matrix = new Matrix();

trace(myMatrix); // (a=2, b=0, c=0, d=2, tx=0, ty=0)
trace(clonedMatrix); // (a=1, b=0, c=0, d=1, tx=0, ty=0)
trace(equals(myMatrix, clonedMatrix)); // false

clonedMatrix = myMatrix.clone();

trace(myMatrix); // (a=2, b=0, c=0, d=2, tx=0, ty=0)
trace(clonedMatrix); // (a=2, b=0, c=0, d=2, tx=0, ty=0)
trace(equals(myMatrix, clonedMatrix)); // true

function equals(m1:Matrix, m2:Matrix):Boolean {
  return m1.toString() == m2.toString();
}
```

## concat (méthode Matrix.concat)

```
public concat(m:Matrix) : Void
```

Concatène une matrice avec la matrice actuelle, ce qui a pour effet de combiner les effets géométriques des deux matrices. En termes mathématiques, la concaténation de deux matrices revient à les combiner par l'intermédiaire de la multiplication de matrices.

Par exemple, si la matrice `m1` redimensionne un objet en le multipliant par 4 et si la matrice `m2` fait pivoter un objet de 1,5707963267949 radians (`Math.PI/2`), alors `m1.concat(m2)` transforme `m1` en matrice qui redimensionne un objet en le multipliant par 4 et le fait pivoter de `Math.PI/2` radians.

Cette méthode permet de remplacer la matrice source par la matrice concaténée. Si vous souhaitez concaténer deux matrices sans modifier l'une des deux matrices source, vous pouvez d'abord copier la matrice source via la méthode clone(), comme indiqué dans la section relative aux exemples.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

### Paramètres

*m*:`flash.geom.Matrix` - La matrice à concaténer avec la matrice source.

## Exemple

L'exemple suivant crée trois matrices définissant des transformations pour trois rectangles de clips. Les deux premières matrices `rotate45Matrix` et `doubleScaleMatrix` sont appliquées aux deux rectangles `rectangleMc_1` et `rectangleMc_2`. Ensuite, la troisième matrice est créée à l'aide de la méthode `concat()` sur `rotate45Matrix` et `doubleScaleMatrix` pour obtenir `scaleAndRotateMatrix`. Cette matrice est ensuite appliquée à `rectangleMc_3` pour le redimensionner et le faire pivoter.

```
import flash.geom.Matrix;
import flash.geom.Transform;

var rectangleMc_0:MovieClip = createRectangle(20, 80, 0x000000);
var rectangleMc_1:MovieClip = createRectangle(20, 80, 0xFF0000);
var rectangleMc_2:MovieClip = createRectangle(20, 80, 0x00FF00);
var rectangleMc_3:MovieClip = createRectangle(20, 80, 0x0000FF);

var rectangleTrans_1:Transform = new Transform(rectangleMc_1);
var rectangleTrans_2:Transform = new Transform(rectangleMc_2);
var rectangleTrans_3:Transform = new Transform(rectangleMc_3);

var rotate45Matrix:Matrix = new Matrix();
rotate45Matrix.rotate(Math.PI/4);
rectangleTrans_1.matrix = rotate45Matrix;
rectangleMc_1._x = 100;
trace(rotate45Matrix.toString()); // (a=0.707106781186548,
  b=0.707106781186547, c=-0.707106781186547, d=0.707106781186548, tx=0,
  ty=0)

var doubleScaleMatrix:Matrix = new Matrix();
doubleScaleMatrix.scale(2, 2);
rectangleTrans_2.matrix = doubleScaleMatrix;
rectangleMc_2._x = 200;
trace(doubleScaleMatrix.toString()); // (a=2, b=0, c=0, d=2, tx=0, ty=0)

var scaleAndRotateMatrix:Matrix = doubleScaleMatrix.clone();
scaleAndRotateMatrix.concat(rotate45Matrix);
rectangleTrans_3.matrix = scaleAndRotateMatrix;
rectangleMc_3._x = 300;
trace(scaleAndRotateMatrix.toString()); // (a=1.4142135623731,
  b=1.41421356237309, c=-1.41421356237309, d=1.4142135623731, tx=0, ty=0)

function createRectangle(width:Number, height:Number,
  color:Number):MovieClip {
  var depth:Number = this.getNextHighestDepth();
  var mc:MovieClip = this.createEmptyMovieClip("mc_" + depth, depth);
  mc.beginFill(color);
  mc.lineTo(0, height);
  mc.lineTo(width, height);
  mc.lineTo(width, 0);
```

```
  mc.lineTo(0, 0);
  return mc;
}
```

# createBox (méthode Matrix.createBox)

```
public createBox(scaleX:Number, scaleY:Number, [rotation:Number],
  [tx:Number], [ty:Number]) : Void
```

Inclut les paramètres de redimensionnement, de rotation et de translation. Lorsqu'elle est appliquée à une matrice, elle définit ses valeurs en fonction de ces paramètres.

L'utilisation de la méthode `createBox()` vous permet d'obtenir la même matrice que celle que vous obtiendriez si vous deviez appliquer les méthodes `identity()`, `rotate()`, `scale()` et `translate()` de manière successive. Par exemple, `mat1.createBox(2,2,Math.PI/5, 100, 100)` permet d'obtenir le résultat suivant :

```
import flash.geom.Matrix;

var mat1:Matrix = new Matrix();
mat1.identity();
mat1.rotate(Math.PI/4);
mat1.scale(2,2);
mat1.translate(10,20);
```

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

## Paramètres

*`scaleX`*`:Number` - Le facteur à appliquer au redimensionnement horizontal.

*`scaleY`*`:Number` - Le facteur à appliquer au redimensionnement vertical.

*`rotation`*`:Number` [facultatif] - La valeur de rotation, en radians. La valeur par défaut est 0.

*`tx`*`:Number` [facultatif] - Le nombre de pixels à translater (déplacer) vers la droite sur l'axe *x*. La valeur par défaut est 0.

*`ty`*`:Number` [facultatif] - Le nombre de pixels à translater (déplacer) vers le bas sur l'axe *y*. La valeur par défaut est 0.

## Exemple

L'exemple suivant définit le redimensionnement `scaleX`, `scaleY`, la rotation, l'emplacement *x* et *y* de `myMatrix` en appelant sa méthode `createBox()`.

```
import flash.geom.Matrix;
import flash.geom.Transform;
```

```
var myMatrix:Matrix = new Matrix();
trace(myMatrix.toString()); // (a=1, b=0, c=0, d=1, tx=0, ty=0)

myMatrix.createBox(1, 2, Math.PI/4, 100, 200);
trace(myMatrix.toString()); // (a=0.707106781186548, b=1.41421356237309,
  c=-0.707106781186547, d=1.4142135623731, tx=100, ty=200)

var rectangleMc:MovieClip = createRectangle(20, 80, 0xFF0000);
var rectangleTrans:Transform = new Transform(rectangleMc);
rectangleTrans.matrix = myMatrix;
```

### Voir également

## createGradientBox (méthode Matrix.createGradientBox)

```
public createGradientBox(width:Number, height:Number, [rotation:Number],
  [tx:Number], [ty:Number]) : Void
```

Crée le style de matrice attendu par la méthode `MovieClip.beginGradientFill()`. La largeur et la hauteur sont redimensionnées selon une paire `scaleX`/`scaleY` et les valeurs `tx`/`ty` sont décalées de la moitié de la largeur et de la hauteur.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

### Paramètres

*`width`*`:Number` - La largeur de la zone de dégradés.

*`height`*`:Number` - La hauteur de la zone de dégradés.

*`rotation`*`:Number` [facultatif] - La valeur de rotation, en radians. La valeur par défaut est 0.

*`tx`*`:Number` [facultatif] - La distance en pixels à translater vers la droite sur l'axe *x*. Cette valeur sera décalée de la moitié du paramètre width. La valeur par défaut est 0.

*`ty`*`:Number` [facultatif] - La distance en pixels à translater vers le bas sur l'axe *y*. Cette valeur sera décalée de la moitié du paramètre height. La valeur par défaut est 0.

### Exemple

L'exemple suivant utilise `myMatrix` en tant que paramètre pour la méthode `beginGradientFill()` de l'objet MovieClip.

```
import flash.geom.Matrix;

var myMatrix:Matrix = new Matrix();
trace(myMatrix.toString()); // (a=1, b=0, c=0, d=1, tx=0, ty=0)

myMatrix.createGradientBox(200, 200, 0, 50, 50);
```

```
trace(myMatrix.toString()); // (a=0.1220703125, b=0, c=0, d=0.1220703125,
  tx=150, ty=150)

var depth:Number = this.getNextHighestDepth();
var mc:MovieClip = this.createEmptyMovieClip("mc_" + depth, depth);
var colors:Array = [0xFF0000, 0x0000FF];
var alphas:Array = [100, 100];
var ratios:Array = [0, 0xFF];
mc.beginGradientFill("linear", colors, alphas, ratios, myMatrix);
mc.lineTo(0, 300);
mc.lineTo(300, 300);
mc.lineTo(300, 0);
mc.lineTo(0, 0);
```

### Voir également

beginGradientFill (méthode MovieClip.beginGradientFill)

## d (propriété Matrix.d)

```
public d : Number
```

Dans la deuxième ligne et la deuxième colonne de l'objet Matrix, la valeur affectant le positionnement des pixels sur l'axe *y* lors du redimensionnement ou de la rotation d'une image.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

### Exemple

L'exemple suivant crée l'objet Matrix `myMatrix` et définit sa valeur `d`.

```
import flash.geom.Matrix;

var myMatrix:Matrix = new Matrix();
trace(myMatrix.d); // 1

myMatrix.d = 2;
trace(myMatrix.d); // 2
```

# deltaTransformPoint (méthode Matrix.deltaTransformPoint)

`public deltaTransformPoint(pt:Point) : Point`

En partant d'un point dans l'espace de coordonnées de prétransformation, cette méthode renvoie les coordonnées de ce point suite à la transformation. Contrairement à la transformation standard appliquée via la méthode transformPoint(), la transformation de la méthode deltaTransformPoint() ne prend pas en considération les paramètres de translation `tx` et `ty`.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

## Paramètres

*`pt`*`:flash.geom.Point` - Un objet Point.

## Renvoie

`flash.geom.Point` - Le nouvel objet Point.

## Exemple

L'exemple suivant utilise la méthode `deltaTransformPoint()` pour créer `deltaTransformedPoint` à partir de `myPoint`. Dans cet exemple, la méthode `translate()` ne modifie pas la position du point intitulé `deltaTransformedPoint`. Cependant, la méthode `scale()` affecte la position du point. Elle multiplie la valeur `x` du point par trois. Celle-ci passe donc de 50 à 150.

```
import flash.geom.Matrix;
import flash.geom.Point;

var myMatrix:Matrix = new Matrix();
trace(myMatrix); // (a=1, b=0, c=0, d=1, tx=0, ty=0)

myMatrix.translate(100, 0);
trace(myMatrix); // (a=1, b=0, c=0, d=1, tx=100, ty=0)

myMatrix.scale(3, 3);
trace(myMatrix); // (a=3, b=0, c=0, d=3, tx=300, ty=0)

var myPoint:Point = new Point(50,0);
trace(myPoint); // (50, 0)

var deltaTransformedPoint:Point = myMatrix.deltaTransformPoint(myPoint);
trace(deltaTransformedPoint); // (150, 0)

var pointMc_0:MovieClip = createRectangle(10, 10, 0xFF0000);
```

```
pointMc_0._x = myPoint.x;

var pointMc_1:MovieClip = createRectangle(10, 10, 0x00FF00);
pointMc_1._x = deltaTransformedPoint.x;

function createRectangle(width:Number, height:Number,
  color:Number):MovieClip {
  var depth:Number = this.getNextHighestDepth();
  var mc:MovieClip = this.createEmptyMovieClip("mc_" + depth, depth);
  mc.beginFill(color);
  mc.lineTo(0, height);
  mc.lineTo(width, height);
  mc.lineTo(width, 0);
  mc.lineTo(0, 0);
  return mc;
}
```

# identity (méthode Matrix.identity)

```
public identity() : Void
```

Définit chaque propriété de matrice sur une valeur qui rend un clip ou une construction géométrique transformé identique à l'original.

Après avoir appelé la méthode `identity()`, la matrice obtenue est dotée des propriétés suivantes : `a=1, b=0, c=0, d=1, tx=0, ty=0`.

Dans la notation des matrices, la matrice d'identité a l'aspect suivant :

$$\begin{bmatrix} 1 & 0 & 0 \end{bmatrix}$$

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

## Exemple

L'exemple suivant démontre que l'appel de la méthode `identity()` convertit l'appel de l'objet Matrix en objet Matrix d'identité. Le nombre et les types de transformation appliqués à l'objet Matrix d'origine auparavant sont inapplicables. Si la méthode `identity()` est appelée, les valeurs de la matrice sont alors converties aux valeurs (a=1, b=0, c=0, d=1, tx=0, ty=0).

```
import flash.geom.Matrix;

var myMatrix:Matrix = new Matrix(2, 0, 0, 2, 0 ,0);
trace(myMatrix.toString()); // (a=2, b=0, c=0, d=2, tx=0, ty=0)

myMatrix.rotate(Math.atan(3/4));
trace(myMatrix.toString()); // (a=1.6, b=1.2, c=-1.2, d=1.6, tx=0, ty=0)

myMatrix.translate(100,200);
```

```
trace(myMatrix.toString()); // (a=1.6, b=1.2, c=-1.2, d=1.6, tx=100,
  ty=200)

myMatrix.scale(2, 2);
trace(myMatrix.toString()); // (a=3.2, b=2.4, c=-2.4, d=3.2, tx=200,
  ty=400)

myMatrix.identity();
trace(myMatrix.toString()); // (a=1, b=0, c=0, d=1, tx=0, ty=0)
```

# invert (méthode Matrix.invert)

```
public invert() : Void
```

Effectue la transformation opposée de la matrice d'origine. Vous pouvez appliquer une matrice inversée à un objet pour annuler la transformation effectuée lors de l'application de la matrice d'origine.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

## Exemple

L'exemple suivant crée `halfScaleMatrix` en appelant la méthode `invert()` de `doubleScaleMatrix`, puis démontre que les deux matrices ont été inversées l'une par rapport à l'autre, annulant ainsi les transformations effectuées par chacune d'entre elles. L'exemple met en évidence cette inversion en créant une matrice `originalAndInverseMatrix`, équivalente à la matrice `noScaleMatrix`.

```
import flash.geom.Matrix;
import flash.geom.Transform;

var rectangleMc_0:MovieClip = createRectangle(20, 80, 0xFF0000);
var rectangleMc_1:MovieClip = createRectangle(20, 80, 0x00FF00);
var rectangleMc_2:MovieClip = createRectangle(20, 80, 0x0000FF);
var rectangleMc_3:MovieClip = createRectangle(20, 80, 0x000000);

var rectangleTrans_0:Transform = new Transform(rectangleMc_0);
var rectangleTrans_1:Transform = new Transform(rectangleMc_1);
var rectangleTrans_2:Transform = new Transform(rectangleMc_2);
var rectangleTrans_3:Transform = new Transform(rectangleMc_3);

var doubleScaleMatrix:Matrix = new Matrix(2, 0, 0, 2, 0, 0);
rectangleTrans_0.matrix = doubleScaleMatrix;
trace(doubleScaleMatrix.toString()); // (a=2, b=0, c=0, d=2, tx=0, ty=0)

var noScaleMatrix:Matrix = new Matrix(1, 0, 0, 1, 0, 0);
rectangleTrans_1.matrix = noScaleMatrix;
rectangleMc_1._x = 100;
trace(noScaleMatrix.toString()); // (a=1, b=0, c=0, d=1, tx=0, ty=0)
```

```
var halfScaleMatrix:Matrix = doubleScaleMatrix.clone();
halfScaleMatrix.invert();
rectangleTrans_2.matrix = halfScaleMatrix;
rectangleMc_2._x = 200;
trace(halfScaleMatrix.toString()); // (a=0.5, b=0, c=0, d=0.5, tx=0, ty=0)

var originalAndInverseMatrix:Matrix = doubleScaleMatrix.clone();
originalAndInverseMatrix.concat(halfScaleMatrix);
rectangleTrans_3.matrix = originalAndInverseMatrix;
rectangleMc_3._x = 300;
trace(originalAndInverseMatrix.toString()); // (a=1, b=0, c=0, d=1, tx=0,
  ty=0)

function createRectangle(width:Number, height:Number,
  color:Number):MovieClip {
  var depth:Number = this.getNextHighestDepth();
  var mc:MovieClip = this.createEmptyMovieClip("mc_" + depth, depth);
  mc.beginFill(color);
  mc.lineTo(0, height);
  mc.lineTo(width, height);
  mc.lineTo(width, 0);
  mc.lineTo(0, 0);
  return mc;
}
```

# Constructeur Matrix

```
public Matrix([a:Number], [b:Number], [c:Number], [d:Number], [tx:Number],
  [ty:Number])
```

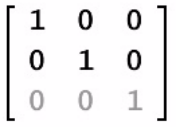
Crée un nouvel objet Matrix avec les paramètres spécifiés. Dans la notation des matrices, les propriétés sont organisées comme suit :

$$\begin{bmatrix} a & b & t_x \\ c & d & t_y \\ 0 & 0 & 1 \end{bmatrix}$$

Si vous ne transmettez aucun paramètre au nouveau constructeur `Matrix()`, celui-ci crée une « matrice d'identité » avec les valeurs suivantes :

| | |
|---|---|
| a = 1 | b = 0 |
| c = 0 | d = 1 |
| tx = 0 | ty = 0 |

Dans la notation des matrices, la matrice d'identité a l'aspect suivant :



**Disponibilité :** ActionScript 1.0 ; Flash Player 8

### Paramètres

*a*:Number [facultatif] - La valeur dans la première ligne et la première colonne du nouvel objet Matrix.

*b*:Number [facultatif] - La valeur dans la première ligne et la deuxième colonne du nouvel objet Matrix.

*c*:Number [facultatif] - La valeur dans la deuxième ligne et la première colonne du nouvel objet Matrix.

*d*:Number [facultatif] - La valeur dans la deuxième ligne et la deuxième colonne du nouvel objet Matrix.

*tx*:Number [facultatif] - La valeur dans la troisième ligne et la première colonne du nouvel objet Matrix.

*ty*:Number [facultatif] - La valeur dans la troisième ligne et la deuxième colonne du nouvel objet Matrix.

### Exemple

L'exemple suivant crée matrix_1 sans transmettre de paramètre au constructeur Matrix et matrix_2 en lui transmettant des paramètres. L'objet Matrix matrix_1, créé sans paramètre, est une matrice d'identité incluant les valeurs (a=1, b=0, c=0, d=1, tx=0, ty=0).

```
import flash.geom.Matrix;

var matrix_1:Matrix = new Matrix();
trace(matrix_1); // (a=1, b=0, c=0, d=1, tx=0, ty=0)

var matrix_2:Matrix = new Matrix(1, 2, 3, 4, 5, 6);
trace(matrix_2); // (a=1, b=2, c=3, d=4, tx=5, ty=6)
```

## rotate (méthode Matrix.rotate)

```
public rotate(angle:Number) : Void
```

Définit les valeurs dans la matrice actuelle de façon à ce qu'elle puisse être utilisée pour appliquer une transformation de rotation.

La méthode `rotate()` modifie les propriétés `a` et `d` de l'objet Matrix. Dans la notation des matrices, ceci est illustré comme suit :

$$\begin{bmatrix} \cos(q) & \sin(q) & 0 \\ -\sin(q) & \cos(q) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

## Paramètres

*`angle`*`:Number` - L'angle de rotation en radians.

## Exemple

L'exemple suivant indique comment la méthode `rotate()` fait pivoter `rectangleMc` de 30 degrés vers la droite. L'application de `myMatrix` à `rectangleMc` redéfinit sa valeur *_x*, ce qui vous oblige à la redéfinir sur 100 manuallement.

```
import flash.geom.Matrix;
import flash.geom.Transform;

var myMatrix:Matrix = new Matrix();
trace(myMatrix.toString()); // (a=1, b=0, c=0, d=1, tx=0, ty=0)

var degrees:Number = 30;
var radians:Number = (degrees/180) Math.PI;
myMatrix.rotate(radians);
trace(myMatrix.toString()); // (a=0.866025403784439, b=0.5, c=-0.5,
  d=0.866025403784439, tx=0, ty=0)

var rectangleMc:MovieClip = createRectangle(20, 80, 0xFF0000);
trace(rectangleMc._x); // 0
rectangleMc._x = 100;
trace(rectangleMc._x); // 100

var rectangleTrans:Transform = new Transform(rectangleMc);
rectangleTrans.matrix = myMatrix;
trace(rectangleMc._x); // 0
rectangleMc._x = 100;
trace(rectangleMc._x); // 100

function createRectangle(width:Number, height:Number,
  color:Number):MovieClip {
  var depth:Number = this.getNextHighestDepth();
  var mc:MovieClip = this.createEmptyMovieClip("mc_" + depth, depth);
  mc.beginFill(color);
  mc.lineTo(0, height);
```

```
  mc.lineTo(width, height);
  mc.lineTo(width, 0);
  mc.lineTo(0, 0);
  return mc;
}
```

L'exemple précédent utilise la propriété `_x` de l'objet MovieClip pour positionner `rectangleMc`. En général, lorsque vous positionnez l'objet Matrix, le recours à plusieurs techniques de positionnement de manière simultanée est considéré comme étant incorrect. L'exemple précédent, écrit selon une syntaxe correcte, permet de concaténer une matrice de translation en `myMatrix` de manière à modifier la coordonnée horizontale de `rectangleMc`. L'exemple ci-dessous l'illustre parfaitement.

```
import flash.geom.Matrix;
import flash.geom.Transform;

var myMatrix:Matrix = new Matrix();
trace(myMatrix.toString()); // (a=1, b=0, c=0, d=1, tx=0, ty=0)

var degrees:Number = 30;
var radians:Number = (degrees/180) Math.PI;
myMatrix.rotate(radians);
trace(myMatrix.toString()); // (a=0.866025403784439, b=0.5, c=-0.5,
  d=0.866025403784439, tx=0, ty=0)

var translateMatrix:Matrix = new Matrix();
translateMatrix.translate(100, 0);
myMatrix.concat(translateMatrix);
trace(myMatrix.toString()); // (a=0.866025403784439, b=0.5, c=-0.5,
  d=0.866025403784439, tx=100, ty=0)

var rectangleMc:MovieClip = createRectangle(20, 80, 0xFF0000);
trace(rectangleMc._x); // 0
rectangleMc._x = 100;
trace(rectangleMc._x); // 100

var rectangleTrans:Transform = new Transform(rectangleMc);
rectangleTrans.matrix = myMatrix;
trace(rectangleMc._x); // 100

function createRectangle(width:Number, height:Number,
  color:Number):MovieClip {
  var depth:Number = this.getNextHighestDepth();
  var mc:MovieClip = this.createEmptyMovieClip("mc_" + depth, depth);
  mc.beginFill(color);
  mc.lineTo(0, height);
  mc.lineTo(width, height);
  mc.lineTo(width, 0);
  mc.lineTo(0, 0);
  return mc;
```

```
}
```

## scale (méthode Matrix.scale)

```
public scale(sx:Number, sy:Number) : Void
```

Modifie une matrice de façon à ce qu'elle redimensionne l'image à laquelle elle est appliquée. Sur l'image redimensionnée, l'emplacement de chaque pixel sur l'axe *x* est multiplié par `sx` ; sur l'axe *y*, il est multiplié par `sy`.

La méthode `scale()` modifie les propriétés `a` et `d` de l'objet Matrix. Dans la notation des matrices, ceci est illustré comme suit :

$$\begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

### Paramètres

*sx*`:Number` - Un multiplicateur utilisé pour redimensionner l'objet sur l'axe *x*.

*sy*`:Number` - Un multiplicateur utilisé pour redimensionner l'objet sur l'axe *y*.

### Exemple

L'exemple suivant utilise la méthode `scale()` pour redimensionner `myMatrix` en appliquant un facteur de 3 à l'horizontale et un facteur de 4 à la verticale.

```
import flash.geom.Matrix;

var myMatrix:Matrix = new Matrix(2, 0, 0, 2, 100, 100);
trace(myMatrix.toString()); // (a=2, b=0, c=0, d=2, tx=100, ty=100)

myMatrix.scale(3, 4);
trace(myMatrix.toString()); // (a=6, b=0, c=0, d=8, tx=300, ty=400)
```

## toString (méthode Matrix.toString)

```
public toString() : String
```

Renvoie une valeur de texte donnant la liste des propriétés de l'objet Matrix.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

### Renvoie

`String` - Une chaîne contenant les valeurs des propriétés de l'objet Matrix : `a`, `b`, `c`, `d`, `tx` et `ty`.

### Exemple

L'exemple suivant crée `myMatrix` et convertit ses valeurs en chaîne au format (a=A, b=B, c=C, d=D, tx=TX, ty=TY).

```
import flash.geom.Matrix;

var myMatrix:Matrix = new Matrix();
trace("myMatrix: " + myMatrix.toString()); // (a=1, b=0, c=0, d=1, tx=0,
  ty=0)
```

## transformPoint (méthode Matrix.transformPoint)

```
public transformPoint(pt:Point) : Point
```

Applique la transformation géométrique représentée par l'objet Matrix au point spécifié.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

### Paramètres

*`pt`*`:flash.geom.Point` - Le point (*x*,*y*) à transformer.

### Renvoie

`flash.geom.Point` - Le nouvel objet Point.

### Exemple

L'exemple suivant utilise la méthode `transformPoint()` pour créer `transformedPoint` à partir de `myPoint`. La méthode `translate()` affecte la position de `transformedPoint`. Dans cet exemple, la méthode `scale()` multiplie la valeur `x` d'origine par trois, passant donc de 50 à 150 ; la méthode `translate()` augmente la valeur `x` à 300, ce qui permet d'obtenir une valeur totale de 450.

```
import flash.geom.Matrix;
import flash.geom.Point;

var myMatrix:Matrix = new Matrix();
trace(myMatrix); // (a=1, b=0, c=0, d=1, tx=0, ty=0)

myMatrix.translate(100, 0);
trace(myMatrix); // (a=1, b=0, c=0, d=1, tx=100, ty=0)

myMatrix.scale(3, 3);
```

```
trace(myMatrix); // (a=3, b=0, c=0, d=3, tx=300, ty=0)

var myPoint:Point = new Point(50,0);
trace(myPoint); // (50, 0)

var transformedPoint:Point = myMatrix.transformPoint(myPoint);
trace(transformedPoint); // (450, 0)

var pointMc_0:MovieClip = createRectangle(10, 10, 0xFF0000);
pointMc_0._x = myPoint.x;

var pointMc_1:MovieClip = createRectangle(10, 10, 0x00FF00);
pointMc_1._x = transformedPoint.x;

function createRectangle(width:Number, height:Number,
  color:Number):MovieClip {
  var depth:Number = this.getNextHighestDepth();
  var mc:MovieClip = this.createEmptyMovieClip("mc_" + depth, depth);
  mc.beginFill(color);
  mc.lineTo(0, height);
  mc.lineTo(width, height);
  mc.lineTo(width, 0);
  mc.lineTo(0, 0);
  return mc;
}
```

# translate (méthode Matrix.translate)

```
public translate(tx:Number, ty:Number) : Void
```

Modifie un objet Matrix de façon à ce que l'effet de sa transformation soit de déplacer un objet sur les axes *x* et *y*.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

## Paramètres

*tx*:Number - La quantité de mouvement sur l'axe *x* vers la droite, en pixels.

*ty*:Number - La quantité de mouvement vers le bas sur l'axe *y*, en pixels.

## Exemple

L'exemple suivant utilise la méthode `translate()` pour positionner `rectangleMc` x:100 et y:50. La méthode `translate()` affecte les propriétés de translation `tx` et `ty`, mais n'affecte pas les propriétés `a`, `b`, `c` ou `d`.

```
import flash.geom.Matrix;

var myMatrix:Matrix = new Matrix(2, 0, 0, 2, 100, 100);
```

```
trace(myMatrix.toString()); // (a=2, b=0, c=0, d=2, tx=100, ty=100)

myMatrix.translate(100, 50);
trace(myMatrix.toString()); // (a=2, b=0, c=0, d=2, tx=200, ty=150)
```

# tx (propriété Matrix.tx)

```
public tx : Number
```

La distance de translation de chaque point sur l'axe *x*. Elle représente la valeur dans la troisième ligne et la première colonne de l'objet Matrix.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

## Exemple

L'exemple suivant crée l'objet Matrix `myMatrix` et définit sa valeur `tx`.

```
import flash.geom.Matrix;

var myMatrix:Matrix = new Matrix();
trace(myMatrix.tx); // 0

myMatrix.tx = 50; // 50
trace(myMatrix.tx);
```

# ty (propriété Matrix.ty)

```
public ty : Number
```

La distance de translation de chaque point sur l'axe *y*. Elle représente la valeur dans la troisième ligne et la deuxième colonne de l'objet Matrix.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

## Exemple

L'exemple suivant crée l'objet Matrix `myMatrix` et définit sa valeur `ty`.

```
import flash.geom.Matrix;

var myMatrix:Matrix = new Matrix();
trace(myMatrix.ty); // 0

myMatrix.ty = 50;
trace(myMatrix.ty); // 50
```

# Microphone

```
Object
  |
  +-Microphone
```

```
public class Microphone
extends Object
```

La classe Microphone vous permet de capturer des données audio à partir d'un microphone relié à l'ordinateur qui exécute Flash Player.

La classe Microphone est principalement dédiée à Flash Communication Server, mais peut être utilisée de façon restreinte sans le serveur, par exemple pour transmettre le son à partir de votre microphone via les haut-parleurs de votre système local.

*Attention :* Flash Player affiche une boîte de dialogue Confidentialité permettant à l'utilisateur d'autoriser ou de refuser l'accès au microphone. Assurez-vous que la taille de votre scène est d'au moins 215 x 138 pixels ; il s'agit de la taille minimale requise par Flash pour afficher la boîte de dialogue.

Les utilisateurs et les administrateurs peuvent également désactiver l'accès au microphone au niveau de chaque site ou de manière globale.

Pour créer ou référencer un objet Microphone, utilisez la méthode `Microphone.get()`.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

## Résumé des propriétés

| Modificateurs | Propriété | Description |
|---|---|---|
| | `activityLevel:Number` [lecture seule] | Une valeur numérique spécifiant le volume sonore détecté par le microphone. |
| | `gain:Number` [lecture seule] | Le volume à partir duquel le microphone accroît le signal. |
| | `index:Number` [lecture seule] | Un entier de base zéro spécifiant l'indice du microphone, tel qu'indiqué dans le tableau renvoyé par `Microphone.names`. |
| | `muted:Boolean` [lecture seule] | Valeur booléenne spécifiant si l'utilisateur a refusé l'accès au microphone (`true`) ou autorisé l'accès (`false`). |
| | `name:String` [lecture seule] | Chaîne spécifiant le nom du périphérique de capture de son actuel, tel que renvoyé par le matériel de capture de son. |
| `static` | `names:Array` [lecture seule] | Récupère un tableau de chaînes reflétant les noms de tous les périphériques de capture de son disponibles sans afficher le panneau Paramètres de contrôle de Flash Player. |
| | `rate:Number` [lecture seule] | Cadence à laquelle le microphone capture le son, en kHz. |
| | `silenceLevel:Number` [lecture seule] | Entier spécifiant le volume sonore requis pour activer le microphone et invoquer `Microphone.onActivity(true)`. |
| | `silenceTimeOut:Number` [lecture seule] | Valeur numérique représentant le nombre de millisecondes entre le moment où le microphone arrête la détection du son et le moment où `Microphone.onActivity(false)` est appelé. |
| | `useEchoSuppression:Boolean` | Propriété (lecture seule) ; valeur booléenne `true` si la suppression de l'écho est activée, `false` sinon. |

*Propriétés héritées de la classe Object*

`constructeur (propriété Object.constructor), __proto__ (Object.__proto__ property), prototype (Object.prototype, propriété), __resolve (Object.__resolve, propriété)`

## Résumé des événements

| Evénement | Description |
|---|---|
| `onActivity = function(active:Boolean) {}` | Invoqué lorsque le microphone commence ou arrête la détection du son. |
| `onStatus = function(infoObject:Object) {}` | Invoqué lorsque l'utilisateur autorise ou refuse l'accès au microphone. |

## Résumé de la méthode

| Modificateurs | Signature | Description |
|---|---|---|
| `static` | `get([index:Number]) : Microphone` | Renvoie une référence à un objet Microphone pour capturer des données audio. |
| | `setGain(gain:Number) : Void` | Définit le gain du microphone, c'est-à-dire la valeur par laquelle le microphone doit multiplier le signal avant de le transmettre. |
| | `setRate(rate:Number) : Void` | Définit le taux de capture du son par le microphone, en kHz. |
| | `setSilenceLevel(silenceLevel:Number, [timeOut:Number]) : Void` | Définit le niveau d'entrée minimal devant être considéré comme du son et (éventuellement) la durée de silence indiquant le début du silence. |
| | `setUseEchoSuppression(useEchoSuppression:Boolean) : Void` | Spécifie s'il convient d'utiliser la fonctionnalité de suppression de l'écho du codec audio. |

*Méthodes héritées de la classe Object*

`addProperty (méthode Object.addProperty), hasOwnProperty (méthode Object.hasOwnProperty), isPropertyEnumerable (méthode Object.isPropertyEnumerable), isPrototypeOf (méthode Object.isPrototypeOf), registerClass (méthode Object.registerClass), toString (méthode Object.toString), unwatch (méthode Object.unwatch), valueOf (méthode Object.valueOf), watch (méthode Object.watch)`

# activityLevel (propriété Microphone.activityLevel)

`public activityLevel : Number [lecture seule]`

Une valeur numérique spécifiant le volume sonore détecté par le microphone. Les valeurs sont comprises entre 0 (aucun son n'est détecté) et 100 (un son de grande intensité est détecté). La valeur de cette propriété peut vous aider à déterminer une valeur appropriée à transmettre à la méthode `Microphone.setSilenceLevel()`.

Si le microphone est disponible mais n'est pas encore utilisé car `Microphone.get()` n'a pas été appelé, cette propriété est définie sur -1.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

## Exemple

L'exemple suivant affiche le niveau d'activité du microphone actuel dans une occurrence ProgressBar intitulée `activityLevel_pb`.

```
var activityLevel_pb:mx.controls.ProgressBar;
activityLevel_pb.mode = "manual";
activityLevel_pb.label = "Activity Level: %3%%";
activityLevel_pb.setStyle("themeColor", "0xFF0000");
this.createEmptyMovieClip("sound_mc", this.getNextHighestDepth());
var active_mic:Microphone = Microphone.get();
sound_mc.attachAudio(active_mic);
this.onEnterFrame = function() {
  activityLevel_pb.setProgress(active_mic.activityLevel, 100);
};
active_mic.onActivity = function(active:Boolean) {
  if (active) {
  var haloTheme_str:String = "haloGreen";
  } else {
  var haloTheme_str:String = "0xFF0000";
  }
  activityLevel_pb.setStyle("themeColor", haloTheme_str);
};
```

La méthode `MovieClip.getNextHighestDepth()` utilisée dans cet exemple nécessite Flash Player 7 ou une version ultérieure. Si votre fichier SWF inclut un composant de la version 2, utilisez la classe DepthManager des composants de la version 2 à la place de la méthode `MovieClip.getNextHighestDepth()`.

## Voir également

`obtenir (méthode Microphone.get)`, `setSilenceLevel (méthode Microphone.setSilenceLevel)`, `setGain (méthode Microphone.setGain)`

# gain (propriété Microphone.gain)

```
public gain : Number [lecture seule]
```

Quantité d'augmentation que le microphone applique au signal pour le renforcer. Les valeurs valides sont comprises entre 0 et 100. La valeur par défaut est 50.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

## Exemple

L'exemple suivant utilise une occurrence ProgressBar intitulée `gain_pb` pour afficher la valeur de gain du microphone et une occurrence NumericStepper intitulée `gain_nstep` pour la définir.

```
this.createEmptyMovieClip("sound_mc", this.getNextHighestDepth());
var active_mic:Microphone = Microphone.get();
sound_mc.attachAudio(active_mic);

gain_pb.label = "Gain: %3";
gain_pb.mode = "manual";
gain_pb.setProgress(active_mic.gain, 100);
gain_nstep.value = active_mic.gain;

function changeGain() {
  active_mic.setGain(gain_nstep.value);
  gain_pb.setProgress(active_mic.gain, 100);
}
gain_nstep.addEventListener("change", changeGain);
```

La méthode `MovieClip.getNextHighestDepth()` utilisée dans cet exemple nécessite Flash Player 7 ou une version ultérieure. Si votre fichier SWF comprend un composant de la version 2, utilisez les composants de la version 2 de la classe DepthManager au lieu de la méthode `MovieClip.getNextHighestDepth()`.

## Voir également

`setGain (méthode Microphone.setGain)`

# obtenir (méthode Microphone.get)

```
public static get([index:Number]) : Microphone
```

Renvoie une référence à un objet Microphone pour capturer des données audio. Pour commencer la capture de l'audio, vous devez relier l'objet Microphone à un objet MovieClip (voir `MovieClip.attachAudio()`).

Contrairement aux objets que vous pouvez créer à l'aide du constructeur `new`, plusieurs appels de `Microphone.get()` font référence au même microphone. Ainsi, si votre script contient les lignes `mic1 = Microphone.get()` et `mic2 = Microphone.get()`, les lignes `mic1` et `mic2` font référence au même microphone (par défaut).

En général, évitez de transmettre une valeur pour *`index`* ; contentez-vous d'utiliser la méthode `Microphone.get()` pour renvoyer une référence au microphone par défaut. Via le panneau Paramètres du microphone (comme indiqué plus bas dans cette section), l'utilisateur peut spécifier le microphone que Flash doit utiliser par défaut. Si vous transmettez une valeur pour *`index`*, vous pouvez essayer de référencer un microphone différent de celui que l'utilisateur préfère utiliser. Vous pouvez utiliser *`index`* en de rares occasions, par exemple si votre application capture des données audio à partir de deux microphones simultanément.

Lorsqu'un fichier SWF tente d'accéder au microphone renvoyé par la méthode `Microphone.get()`, par exemple lorsque vous exécutez `MovieClip.attachAudio()`, Flash Player affiche une boîte de dialogue Confidentialité permettant à l'utilisateur d'autoriser ou de refuser l'accès au microphone. (Assurez-vous que la taille de votre scène est d'au moins 215 x 138 pixels ; il s'agit de la taille minimale requise par Flash pour afficher la boîte de dialogue.)

Lorsque l'utilisateur répond à cette boîte de dialogue, le gestionnaire d'événements `Microphone.onStatus` renvoie un objet d'informations qui indique la réponse de l'utilisateur. Pour déterminer si l'utilisateur a refusé ou autorisé l'accès à la caméra sans traiter ce gestionnaire d'événements, utilisez `Microphone.muted`.

L'utilisateur peut également spécifier des paramètres de confidentialité permanents pour un domaine spécifique. Pour ce faire, il lui suffit de cliquer avec le bouton droit (Windows) ou d'appuyer sur la touche Contrôle (Macintosh) lors de la lecture d'un fichier SWF, de pointer sur Paramètres, d'ouvrir le panneau Confidentialité, puis de sélectionner Mémoriser.

Vous ne pouvez pas utiliser ActionScript pour définir la valeur Autoriser ou Refuser d'un utilisateur, mais vous pouvez afficher le panneau Confidentialité pour l'utilisateur via `System.showSettings(0)`. Si l'utilisateur sélectionne Mémoriser, Flash Player n'affiche plus la boîte de dialogue Confidentialité pour les fichiers SWF de ce domaine.

Si `Microphone.get()` renvoie la valeur `null`, cela signifie que le microphone est utilisé par une autre application ou qu'aucun microphone n'est installé sur le système. Pour déterminer si un microphone est installé, utilisez `Microphones.names.length`. Pour afficher le panneau Paramètres du microphone de Flash Player, qui permet à l'utilisateur de choisir le microphone référencé par `Microphone.get()`, utilisez `System.showSettings(2)`.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

### Paramètres

*index*:Number [facultatif] - Entier de base zéro spécifiant le microphone à sélectionner, comme déterminé dans le tableau inclus dans Microphone.names. Pour obtenir le microphone par défaut (ce qui est recommandé pour la plupart des applications), omettez ce paramètre.

### Valeur renvoyée

Microphone -

- Si *index* n'est pas spécifié, cette méthode renvoie une référence au microphone par défaut ou, s'il n'est pas disponible, au premier microphone disponible. Si aucun microphone n'est disponible ou installé, la méthode renvoie null.
- Si *index* est spécifié, cette méthode renvoie une référence au microphone demandé ou null s'il n'est pas disponible.

### Exemple

L'exemple suivant permet à l'utilisateur de spécifier le microphone par défaut, puis capture des données audio et les lit en local. Pour éviter de recevoir des commentaires, il peut s'avérer judicieux d'utiliser un casque pour tester ce code.

```
this.createEmptyMovieClip("sound_mc", this.getNextHighestDepth());
System.showSettings(2);
var active_mic:Microphone = Microphone.get();
sound_mc.attachAudio(active_mic);
```

La méthode MovieClip.getNextHighestDepth() utilisée dans cet exemple nécessite Flash Player 7 ou une version ultérieure. Si votre fichier SWF comprend un composant de la version 2, utilisez les composants de la version 2 de la classe DepthManager au lieu de la méthode MovieClip.getNextHighestDepth().

### Voir également

obtenir (méthode Microphone.get), index (propriété Microphone.index), muted (propriété microphone.muted), names (propriété Microphone.names), onStatus (gestionnaire Microphone.onStatus), attachAudio (méthode MovieClip.attachAudio), showSettings (méthode System.showSettings)

## index (propriété Microphone.index)

public index : Number [lecture seule]

Entier de base zéro spécifiant l'index du microphone, tel qu'indiqué dans le tableau renvoyé par Microphone.names.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

### Exemple

L'exemple suivant affiche le nom des périphériques de capture de son disponibles sur votre ordinateur dans une occurrence ComboBox intitulée `mic_cb`. Une occurrence du composant Label, intitulée `mic_lbl`, affiche l'index du microphone. Vous pouvez utiliser le composant ComboBox pour passer d'un périphérique à l'autre.

```
var mic_lbl:mx.controls.Label;
var mic_cb:mx.controls.ComboBox;

this.createEmptyMovieClip("sound_mc", this.getNextHighestDepth());
var active_mic:Microphone = Microphone.get();
sound_mc.attachAudio(active_mic);
mic_lbl.text = "["+active_mic.index+"] "+active_mic.name;
mic_cb.dataProvider = Microphone.names;
mic_cb.selectedIndex = active_mic.index;

var cbListener:Object = new Object();
cbListener.change = function(evt:Object) {
  active_mic = Microphone.get(evt.target.selectedIndex);
  sound_mc.attachAudio(active_mic);
  mic_lbl.text = "["+active_mic.index+"] "+active_mic.name;
};
mic_cb.addEventListener("change", cbListener);
```

La méthode `MovieClip.getNextHighestDepth()` utilisée dans cet exemple nécessite Flash Player 7 ou une version ultérieure. Si votre fichier SWF comprend un composant de la version 2, utilisez les composants de la version 2 de la classe DepthManager au lieu de la méthode `MovieClip.getNextHighestDepth()`.

### Voir également

`obtenir (méthode Microphone.get)`, `names (propriété Microphone.names)`

## muted (propriété microphone.muted)

```
public muted : Boolean [lecture seule]
```

Valeur booléenne spécifiant si l'utilisateur a refusé (`true`) ou autorisé (`false`) l'accès au microphone. Lorsque cette valeur change, `Microphone.onStatus` est appelé. Pour plus d'informations, consultez `Microphone.get()`.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

### Exemple

Cet exemple sélectionne le microphone par défaut et vérifie si le son est coupé ou non.

```
var active_mic:Microphone = Microphone.get();
trace(active_mic.muted);
```

### Voir également

obtenir (méthode Microphone.get), onStatus (gestionnaire Microphone.onStatus)

## name (propriété Microphone.name)

```
public name : String [lecture seule]
```

Chaîne spécifiant le nom du périphérique de capture de son actuel, tel que renvoyé par le matériel de capture de son.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

### Exemple

L'exemple suivant affiche des informations sur le(s) périphérique(s) de capture du son présent(s) sur votre système informatique, y compris un tableau de noms et le périphérique par défaut.

```
var status_ta:mx.controls.TextArea;
status_ta.html = false;
status_ta.setStyle("fontSize", 9);
var microphone_array:Array = Microphone.names;
var active_mic:Microphone = Microphone.get();
status_ta.text = "The default device is: "+active_mic.name+newline+newline;
status_ta.text += "You have "+microphone_array.length+" device(s)
  installed."+newline+newline;
for (var i = 0; i<microphone_array.length; i++) {
  status_ta.text += "["+i+"] "+microphone_array[i]+newline;
}
```

### Voir également

obtenir (méthode Microphone.get), names (propriété Microphone.names)

## names (propriété Microphone.names)

```
public static names : Array [lecture seule]
```

Récupère un tableau de chaînes reflétant les noms de tous les périphériques de capture de son disponibles sans afficher le panneau Paramètres de contrôle de Flash Player. Ce tableau se comporte de la même manière que tout autre tableau ActionScript, fournissant de façon implicite l'index basé sur zéro de chaque périphérique de capture de son et le nombre de périphériques de capture de son présents sur le système (via `Microphone.names.length`). Pour plus d'informations, consultez l'entrée de classe Array de Microphone.names.

L'appel de la propriété `Microphone.names` nécessite un examen minutieux du matériel et plusieurs secondes peuvent être nécessaires pour créer le tableau. Dans la plupart des cas, vous pouvez utiliser le microphone par défaut.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

### Exemple

L'exemple suivant affiche des informations sur le(s) périphérique(s) de capture du son présent(s) sur votre système informatique, y compris un tableau de noms et le périphérique par défaut.

```
var status_ta:mx.controls.TextArea;
status_ta.html = false;
status_ta.setStyle("fontSize", 9);
var microphone_array:Array = Microphone.names;
var active_mic:Microphone = Microphone.get();
status_ta.text = "The default device is: "+active_mic.name+newline+newline;
status_ta.text += "You have "+microphone_array.length+" device(s)
  installed."+newline+newline;
for (var i = 0; i<microphone_array.length; i++) {
  status_ta.text += "["+i+"] "+microphone_array[i]+newline;
}
```

Par exemple, vous pouvez afficher les informations suivantes :

```
The default device is: Logitech USB Headset
You have 2 device(s) installed.
[0] Logitech USB Headset
[1] YAMAHA AC-XG WDM Audio
```

### Voir également

`name (propriété Microphone.name)`, `obtenir (méthode Microphone.get)`

## onActivity (gestionnaire microphone.onActivity)

```
onActivity = function(active:Boolean) {}
```

Invoqué lorsque le microphone commence ou arrête la détection du son. Si vous souhaitez répondre à ce gestionnaire d'événements, vous devez créer une fonction pour traiter sa valeur d'activité.

Pour spécifier le volume sonore requis pour appeler `Microphone.onActivity(true)` et la durée qui doit s'écouler sans son avant d'appeler `Microphone.onActivity(false)`, utilisez `Microphone.setSilenceLevel()`.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

## Paramètres

*`active`*`:Boolean` - Valeur booléenne définie sur true lorsque le microphone commence la détection du son et sur false lorsqu'il cesse.

## Exemple

L'exemple suivant affiche le niveau d'activité dans une occurrence ProgressBar intitulée `activityLevel_pb`. Lorsque le microphone détecte le son, il appelle la fonction `onActivity` qui modifie l'occurrence ProgressBar.

```
var activityLevel_pb:mx.controls.ProgressBar;
activityLevel_pb.mode = "manual";
activityLevel_pb.label = "Activity Level: %3%%";
this.createEmptyMovieClip("sound_mc", this.getNextHighestDepth());
var active_mic:Microphone = Microphone.get();
sound_mc.attachAudio(active_mic);
active_mic.onActivity = function(active:Boolean) {
  if (active) {
    activityLevel_pb.indeterminate = false;
    activityLevel_pb.label = "Activity Level: %3%%";
  } else {
    activityLevel_pb.indeterminate = true;
    activityLevel_pb.label = "Activity Level: (inactive)";
  }
};
this.onEnterFrame = function() {
  activityLevel_pb.setProgress(active_mic.activityLevel, 100);
};
```

La méthode `MovieClip.getNextHighestDepth()` utilisée dans cet exemple nécessite Flash Player 7 ou une version ultérieure. Si votre fichier SWF comprend un composant de la version 2, utilisez les composants de la version 2 de la classe DepthManager au lieu de la méthode `MovieClip.getNextHighestDepth()`.

## Voir également

`setSilenceLevel (méthode Microphone.setSilenceLevel)`

# onStatus (gestionnaire Microphone.onStatus)

```
onStatus = function(infoObject:Object) {}
```

Invoqué lorsque l'utilisateur autorise ou refuse l'accès au microphone. Si vous souhaitez répondre à ce gestionnaire d'événements, vous devez créer une fonction pour traiter l'objet d'informations généré par le microphone.

Lorsqu'un fichier SWF tente d'accéder au microphone, Flash Player affiche une boîte de dialogue Confidentialité permettant à l'utilisateur d'autoriser ou de refuser l'accès.

- Si l'utilisateur autorise l'accès, la propriété Microphone.muted est définie sur false, et ce gestionnaire d'événements est appelé avec un objet d'informations dont la propriété de code est « Microphone.Unmuted » et la propriété de niveau « Status ».
- Si l'utilisateur refuse l'accès, la propriété Microphone.muted est définie sur true, et ce gestionnaire d'événements est appelé avec un objet d'informations dont la propriété de code est « Microphone.Muted » et la propriété de niveau « Status ».

Pour déterminer si l'utilisateur a refusé ou autorisé l'accès au microphone sans traiter ce gestionnaire d'événements, utilisez la propriété `Microphone.muted`.

Remarque : Si l'utilisateur choisit d'autoriser ou de refuser définitivement l'accès à tous les fichiers SWF d'un domaine spécifié, cette méthode n'est pas appelée pour les fichiers SWF de ce domaine sauf si l'utilisateur modifie ultérieurement le paramètre de confidentialité.

Pour plus d'informations, consultez `Microphone.get()`.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

## Paramètres

*`infoObject`*`:Object` - Paramètre défini selon le message d'état.

## Exemple

L'exemple suivant affiche la boîte de dialogue Confidentialité permettant à l'utilisateur d'autoriser ou de refuser l'accès au microphone lorsqu'il clique sur un hyperlien. Si l'utilisateur choisit de refuser l'accès, *muet* s'affiche en texte rouge de grande taille. Si l'accès au microphone est autorisé, ce texte n'apparaît pas.

```
this.createTextField("muted_txt", this.getNextHighestDepth(), 10, 10, 100,
  22);
muted_txt.autoSize = true;
muted_txt.html = true;
muted_txt.selectable = false;
muted_txt.htmlText = "<a href=\"asfunction:System.showSettings\"><u>Click
  Here</u></a> to Allow/Deny access.";
this.createEmptyMovieClip("sound_mc", this.getNextHighestDepth());
var active_mic:Microphone = Microphone.get();
```

```
sound_mc.attachAudio(active_mic);
active_mic.onStatus = function(infoObj:Object) {
  status_txt._visible = active_mic.muted;
  muted_txt.htmlText = "Status: <a
  href=\"asfunction:System.showSettings\"><u>"+infoObj.code+"</u></a>";
};
this.createTextField("status_txt", this.getNextHighestDepth(), 0, 0, 100,
  22);
status_txt.html = true;
status_txt.autoSize = true;
status_txt.htmlText = "<font size='72' color='#FF0000'>muted</font>";
status_txt._x = (Stage.width-status_txt._width)/2;
status_txt._y = (Stage.height-status_txt._height)/2;
status_txt._visible = active_mic.muted;
```

La méthode `MovieClip.getNextHighestDepth()` utilisée dans cet exemple nécessite Flash Player 7 ou une version ultérieure. Si votre fichier SWF comprend un composant de la version 2, utilisez les composants de la version 2 de la classe DepthManager au lieu de la méthode `MovieClip.getNextHighestDepth()`.

### Voir également

obtenir (méthode Microphone.get), muted (propriété microphone.muted), showSettings (méthode System.showSettings), onStatus (gestionnaire System.onStatus)

## rate (propriété Microphone.rate)

```
public rate : Number [lecture seule]
```

Cadence à laquelle le microphone capture le son, en kHz. La valeur par défaut est 8 kHz si votre périphérique de capture de son prend en charge cette valeur. Dans le cas contraire, la valeur par défaut est le niveau de capture supérieur à 8 kHz immédiatement disponible pouvant être pris en charge par votre périphérique de capture de son, généralement 11 kHz.

Pour configurer cette valeur, utilisez `Microphone.setRate()`.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

### Exemple

Le code suivant vous permet d'utiliser une occurrence ComboBox, intitulée `rate_cb`, pour modifier la cadence à laquelle votre microphone capture le son. La cadence actuelle s'affiche dans une occurrence d'étiquette intitulée `rate_lbl`.

```
this.createEmptyMovieClip("sound_mc", this.getNextHighestDepth());
var active_mic:Microphone = Microphone.get();
sound_mc.attachAudio(active_mic);
var rate_array:Array = new Array(5, 8, 11, 22, 44);
```

```
rate_cb.dataProvider = rate_array;
rate_cb.labelFunction = function(item:Object) {
  return (item+" kHz");
};
for (var i = 0; i<rate_array.length; i++) {
  if (rate_cb.getItemAt(i) == active_mic.rate) {
  rate_cb.selectedIndex = i;
  break;
  }
}
function changeRate() {
  active_mic.setRate(rate_cb.selectedItem);
  rate_lbl.text = "Current rate: "+active_mic.rate+" kHz";
}
rate_cb.addEventListener("change", changeRate);
rate_lbl.text = "Current rate: "+active_mic.rate+" kHz";
```

La méthode `MovieClip.getNextHighestDepth()` utilisée dans cet exemple nécessite Flash Player 7 ou une version ultérieure. Si votre fichier SWF comprend un composant de la version 2, utilisez les composants de la version 2 de la classe DepthManager au lieu de la méthode `MovieClip.getNextHighestDepth()`.

### Voir également

`setRate (méthode Microphone.setRate)`

## setGain (méthode Microphone.setGain)

```
public setGain(gain:Number) : Void
```

Définit le gain du microphone, c'est-à-dire la valeur par laquelle le microphone doit multiplier le signal avant de le transmettre. Une valeur 0 indique à Flash de multiplier par 0 ; autrement dit, le microphone ne transmet aucun son.

Vous pouvez comparer ce paramètre au bouton volume d'une stéréo : 0 correspond à l'absence de volume et 50 correspond au volume normal ; les nombres inférieurs à 50 indiquent un volume inférieur au volume normal, tandis que les nombres supérieurs à 50 indiquent un volume supérieur au volume normal.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

### Paramètres

*`gain`*`:Number` - Entier spécifiant la quantité dont le microphone doit accroître le volume du signal. Les valeurs valides sont comprises entre 0 et 100. La valeur par défaut est 50 ; toutefois, l'utilisateur peut modifier cette valeur dans le panneau Paramètres du microphone de Flash Player.

### Exemple

L'exemple suivant utilise une occurrence ProgressBar intitulée `gain_pb` pour afficher la valeur de gain du microphone et une occurrence NumericStepper intitulée `gain_nstep` pour la définir.

```
this.createEmptyMovieClip("sound_mc", this.getNextHighestDepth());
var active_mic:Microphone = Microphone.get();
sound_mc.attachAudio(active_mic);

gain_pb.label = "Gain: %3";
gain_pb.mode = "manual";
gain_pb.setProgress(active_mic.gain, 100);
gain_nstep.value = active_mic.gain;

function changeGain() {
  active_mic.setGain(gain_nstep.value);
  gain_pb.setProgress(active_mic.gain, 100);
}
gain_nstep.addEventListener("change", changeGain);
```

La méthode `MovieClip.getNextHighestDepth()` utilisée dans cet exemple nécessite Flash Player 7 ou une version ultérieure. Si votre fichier SWF comprend un composant de la version 2, utilisez les composants de la version 2 de la classe DepthManager au lieu de la méthode `MovieClip.getNextHighestDepth()`.

### Voir également

`gain (propriété Microphone.gain)`, `setUseEchoSuppression (méthode microphone.setUseEchoSuppression)`

## setRate (méthode Microphone.setRate)

```
public setRate(rate:Number) : Void
```

Définit le taux de capture du son par le microphone, en kHz.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

### Paramètres

*`rate`*`:Number` - Cadence à laquelle le microphone doit capturer le son, en kHz. Les valeurs acceptables sont 5, 8, 11, 22 et 44. La valeur par défaut est 8 kHz si votre périphérique de capture de son prend en charge cette valeur. Dans le cas contraire, la valeur par défaut est le niveau de capture supérieur à 8 kHz immédiatement disponible pouvant être pris en charge par votre périphérique de capture de son, généralement 11 kHz.

## Exemple

L'exemple suivant définit la cadence du microphone selon les préférences de l'utilisateur (que vous avez affectées à la variable `userRate`) s'il s'agit de l'une des valeurs suivantes : 5, 8, 11, 22 ou 44. Si ce n'est pas le cas, la valeur est arrondie à la valeur acceptable la plus proche prise en charge par le périphérique de capture de son.

```
active_mic.setRate(userRate);
```

L'exemple suivant vous permet d'utiliser une occurrence ComboBox, intitulée `rate_cb`, pour modifier la cadence à laquelle votre microphone capture le son. La cadence actuelle s'affiche dans une occurrence d'étiquette intitulée `rate_lbl`.

```
this.createEmptyMovieClip("sound_mc", this.getNextHighestDepth());
var active_mic:Microphone = Microphone.get();
sound_mc.attachAudio(active_mic);
var rate_array:Array = new Array(5, 8, 11, 22, 44);
rate_cb.dataProvider = rate_array;
rate_cb.labelFunction = function(item:Object) {
  return (item+" kHz");
};
for (var i = 0; i<rate_array.length; i++) {
  if (rate_cb.getItemAt(i) == active_mic.rate) {
  rate_cb.selectedIndex = i;
  break;
  }
}
function changeRate() {
  active_mic.setRate(rate_cb.selectedItem);
  rate_lbl.text = "Current rate: "+active_mic.rate+" kHz";
}
rate_cb.addEventListener("change", changeRate);
rate_lbl.text = "Current rate: "+active_mic.rate+" kHz";
```

La méthode `MovieClip.getNextHighestDepth()` utilisée dans cet exemple nécessite Flash Player 7 ou une version ultérieure. Si votre fichier SWF comprend un composant de la version 2, utilisez les composants de la version 2 de la classe DepthManager au lieu de la méthode `MovieClip.getNextHighestDepth()`.

## Voir également

`rate (propriété Microphone.rate)`

# setSilenceLevel (méthode Microphone.setSilenceLevel)

```
public setSilenceLevel(silenceLevel:Number, [timeOut:Number]) : Void
```

Définit le niveau d'entrée minimal devant être considéré comme du son et (éventuellement) la durée de silence indiquant le début du silence.

- Pour empêcher le microphone de détecter le son, transmettez une valeur de 100 pour le paramètre *level* ; dans ce cas, `Microphone.onActivity` n'est jamais appelé.
- Pour déterminer le volume sonore actuellement détecté par le microphone, utilisez `Microphone.activityLevel`.

La détection de l'activité est la capacité à détecter les niveaux audio indiquant qu'une personne est en train de parler. Lorsqu'une personne ne parle pas, il est possible d'économiser de la bande passante car il n'est pas nécessaire d'envoyer le flux continu correspondant. Ces informations peuvent être également utilisées à des fins de réponse visuelle afin qu'ils (ou d'autres personnes) sachent qu'ils sont silencieux.

Les valeurs de silence correspondent directement aux valeurs d'activité. La valeur d'activité 0 correspond au silence total. La valeur d'activité 100 correspond au bruit intense constant (le niveau d'intensité maximal pouvant être enregistré selon le paramètre de gain actuel). Une fois le réglage du gain effectué de manière appropriée, votre valeur d'activité est inférieure à votre valeur de silence lorsque vous ne parlez pas ; lorsque vous parlez, la valeur d'activité est supérieure à votre valeur de silence.

L'objectif de cette méthode est similaire à celui de `Camera.setMotionLevel()` ; les deux méthodes sont utilisées pour spécifier à quel moment il convient d'appeler le gestionnaire d'événements `onActivity`. Toutefois, l'impact de ces méthodes sur la publication des flux continus diffère de manière significative :

- `Camera.setMotionLevel()` est conçu pour détecter le mouvement et n'a aucune incidence sur l'utilisation de la bande passante. Même si un flux vidéo ne détecte pas le mouvement, la vidéo est toujours envoyée.
- `Microphone.setSilenceLevel()` est conçu pour optimiser la bande passante. Lorsqu'un flux continu est considéré comme étant silencieux, aucune donnée audio n'est envoyée. En revanche, un message unique est envoyé, indiquant le début du silence.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

## Paramètres

*`silenceLevel`*`:Number` - Entier spécifiant le volume sonore requis pour activer le microphone et invoquer `Microphone.onActivity(true)`. Les valeurs acceptables sont comprises entre 0 et 100. La valeur par défaut est 10.

*timeOut*:Number [facultatif] - Entier spécifiant le nombre de millisecondes qui doit s'écouler sans activité avant que Flash considère que le son a cessé et qu'il peut appeler `Microphone.onActivity(false)`. La valeur par défaut est 2 000 (2 secondes).

## Exemple

L'exemple suivant modifie le niveau de silence en fonction de l'entrée de l'utilisateur dans une occurrence NumericStepper intitulée `silenceLevel_nstep`. L'occurrence ProgressBar intitulée `silenceLevel_pb` modifie son aspect selon que le flux continu est considéré comme silencieux ou non. Sinon, elle affiche le niveau d'activité du flux continu.

```
var silenceLevel_pb:mx.controls.ProgressBar;
var silenceLevel_nstep:mx.controls.NumericStepper;

this.createEmptyMovieClip("sound_mc", this.getNextHighestDepth());
var active_mic:Microphone = Microphone.get();
sound_mc.attachAudio(active_mic);

silenceLevel_pb.label = "Activity level: %3";
silenceLevel_pb.mode = "manual";
silenceLevel_nstep.minimum = 0;
silenceLevel_nstep.maximum = 100;
silenceLevel_nstep.value = active_mic.silenceLevel;

var nstepListener:Object = new Object();
nstepListener.change = function(evt:Object) {
  active_mic.setSilenceLevel(evt.target.value, active_mic.silenceTimeOut);
};
silenceLevel_nstep.addEventListener("change", nstepListener);

this.onEnterFrame = function() {
  silenceLevel_pb.setProgress(active_mic.activityLevel, 100);
};
active_mic.onActivity = function(active:Boolean) {
  if (active) {
  silenceLevel_pb.indeterminate = false;
  silenceLevel_pb.setStyle("themeColor", "haloGreen");
  silenceLevel_pb.label = "Activity level: %3";
  } else {
  silenceLevel_pb.indeterminate = true;
  silenceLevel_pb.setStyle("themeColor", "0xFF0000");
  silenceLevel_pb.label = "Activity level: (inactive)";
  }
};
```

La méthode `MovieClip.getNextHighestDepth()` utilisée dans cet exemple nécessite Flash Player 7 ou une version ultérieure. Si votre fichier SWF comprend un composant de la version 2, utilisez les composants de la version 2 de la classe DepthManager au lieu de la méthode `MovieClip.getNextHighestDepth()`.

### Voir également

`setMotionLevel (méthode Camera.setMotionLevel )`, `activityLevel (propriété Microphone.activityLevel)`, `onActivity (gestionnaire microphone.onActivity)`, `setGain (méthode Microphone.setGain)`, `silenceLevel (propriété Microphone.silenceLevel)`, `silenceTimeOut (propriété Microphone.silenceTimeOut)`

## setUseEchoSuppression (méthode microphone.setUseEchoSuppression)

```
public setUseEchoSuppression(useEchoSuppression:Boolean) : Void
```

Spécifie s'il convient d'utiliser la fonctionnalité de suppression de l'écho du codec audio. La valeur par défaut est `false`, sauf si l'utilisateur sélectionne Réduire l'écho dans le panneau Paramètres du microphone de Flash Player.

La suppression de l'écho vise à réduire les effets de la réaction acoustique créée lorsque le son qui sort du haut-parleur est capté par le microphone sur le même ordinateur. (Elle diffère de l'annulation de l'écho qui supprime entièrement la réaction acoustique.)

En général, il est recommandé de supprimer l'écho lorsque le son capturé est lu via les haut-parleurs, et non par le casque, sur le même ordinateur. Si votre fichier SWF autorise les utilisateurs à spécifier le périphérique de sortie audio, il peut être souhaitable d'appeler `Microphone.setUseEchoSuppression(true)` s'ils précisent qu'ils utilisent les haut-parleurs et qu'ils ont également l'intention d'utiliser le microphone.

Les utilisateurs peuvent également ajuster ces paramètres dans le panneau Paramètres du microphone de Flash Player.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

### Paramètres

*`useEchoSuppression`*`:Boolean` - Valeur booléenne indiquant s'il convient d'utiliser la suppression de l'écho (`true`) ou non (`false`).

### Exemple

L'exemple suivant active la suppression de l'écho si l'utilisateur sélectionne une occurrence CheckBox appelée `useEchoSuppression_ch`. L'occurrence ProgressBar intitulée `activityLevel_pb` affiche le niveau d'activité actuel du flux continu.

```
var useEchoSuppression_ch:mx.controls.CheckBox;
var activityLevel_pb:mx.controls.ProgressBar;

this.createEmptyMovieClip("sound_mc", this.getNextHighestDepth());
```

```
var active_mic:Microphone = Microphone.get();
sound_mc.attachAudio(active_mic);

activityLevel_pb.mode = "manual";
activityLevel_pb.label = "Activity Level: %3";
useEchoSuppression_ch.selected = active_mic.useEchoSuppression;
this.onEnterFrame = function() {
  activityLevel_pb.setProgress(active_mic.activityLevel, 100);
};
var chListener:Object = new Object();
chListener.click = function(evt:Object) {
  active_mic.setUseEchoSuppression(evt.target.selected);
};
useEchoSuppression_ch.addEventListener("click", chListener);
```

La méthode `MovieClip.getNextHighestDepth()` utilisée dans cet exemple nécessite Flash Player 7 ou une version ultérieure. Si votre fichier SWF comprend un composant de la version 2, utilisez les composants de la version 2 de la classe DepthManager au lieu de la méthode `MovieClip.getNextHighestDepth()`.

### Voir également

`setUseEchoSuppression (méthode microphone.setUseEchoSuppression)`, `useEchoSuppression (propriété Microphone.useEchoSuppression)`

## silenceLevel (propriété Microphone.silenceLevel)

`public silenceLevel : Number [lecture seule]`

Entier spécifiant le volume sonore requis pour activer le microphone et invoquer `Microphone.onActivity(true)`. La valeur par défaut est 10.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

### Exemple

L'exemple suivant modifie le niveau de silence en fonction de l'entrée de l'utilisateur dans une occurrence NumericStepper intitulée `silenceLevel_nstep`. L'occurrence ProgressBar intitulée `silenceLevel_pb` modifie son aspect selon que le flux continu est considéré comme étant silencieux ou non. Sinon, elle affiche le niveau d'activité du flux continu.

```
var silenceLevel_pb:mx.controls.ProgressBar;
var silenceLevel_nstep:mx.controls.NumericStepper;

this.createEmptyMovieClip("sound_mc", this.getNextHighestDepth());
var active_mic:Microphone = Microphone.get();
sound_mc.attachAudio(active_mic);

silenceLevel_pb.label = "Activity level: %3";
```

```
silenceLevel_pb.mode = "manual";
silenceLevel_nstep.minimum = 0;
silenceLevel_nstep.maximum = 100;
silenceLevel_nstep.value = active_mic.silenceLevel;

var nstepListener:Object = new Object();
nstepListener.change = function(evt:Object) {
  active_mic.setSilenceLevel(evt.target.value, active_mic.silenceTimeOut);
};
silenceLevel_nstep.addEventListener("change", nstepListener);

this.onEnterFrame = function() {
  silenceLevel_pb.setProgress(active_mic.activityLevel, 100);
};
active_mic.onActivity = function(active:Boolean) {
  if (active) {
  silenceLevel_pb.indeterminate = false;
  silenceLevel_pb.setStyle("themeColor", "haloGreen");
  silenceLevel_pb.label = "Activity level: %3";
  } else {
  silenceLevel_pb.indeterminate = true;
  silenceLevel_pb.setStyle("themeColor", "0xFF0000");
  silenceLevel_pb.label = "Activity level: (inactive)";
  }
};
```

La méthode `MovieClip.getNextHighestDepth()` utilisée dans cet exemple nécessite Flash Player 7 ou une version ultérieure. Si votre fichier SWF comprend un composant de la version 2, utilisez les composants de la version 2 de la classe DepthManager au lieu de la méthode `MovieClip.getNextHighestDepth()`.

### Voir également

`gain (propriété Microphone.gain)`, `setSilenceLevel (méthode Microphone.setSilenceLevel)`

## silenceTimeOut (propriété Microphone.silenceTimeOut)

`public silenceTimeOut : Number [lecture seule]`

Valeur numérique représentant le nombre de millisecondes entre le moment où le microphone arrête la détection du son et le moment où `Microphone.onActivity(false)` est appelé. La valeur par défaut est 2 000 (2 secondes).

Pour configurer cette valeur, utilisez `Microphone.setSilenceLevel()`.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

## Exemple

L'exemple suivant permet à l'utilisateur de contrôler la durée écoulée entre le moment où le microphone arrête la détection du son et le moment où `Microphone.onActivity(false)` est appelé. L'utilisateur contrôle cette valeur via une occurrence NumericStepper intitulée silenceTimeOut_nstep. L'occurrence ProgressBar intitulée `silenceLevel_pb` modifie son aspect selon que le flux continu est considéré comme étant silencieux ou non. Sinon, elle affiche le niveau d'activité du flux continu.

```
var silenceLevel_pb:mx.controls.ProgressBar;
var silenceTimeOut_nstep:mx.controls.NumericStepper;

this.createEmptyMovieClip("sound_mc", this.getNextHighestDepth());
var active_mic:Microphone = Microphone.get();
sound_mc.attachAudio(active_mic);

silenceLevel_pb.label = "Activity level: %3";
silenceLevel_pb.mode = "manual";
silenceTimeOut_nstep.minimum = 0;
silenceTimeOut_nstep.maximum = 10;
silenceTimeOut_nstep.value = active_mic.silenceTimeOut/1000;

var nstepListener:Object = new Object();
nstepListener.change = function(evt:Object) {
  active_mic.setSilenceLevel(active_mic.silenceLevel, evt.target.value
  1000);
};
silenceTimeOut_nstep.addEventListener("change", nstepListener);

this.onEnterFrame = function() {
  silenceLevel_pb.setProgress(active_mic.activityLevel, 100);
};
active_mic.onActivity = function(active:Boolean) {
  if (active) {
  silenceLevel_pb.indeterminate = false;
  silenceLevel_pb.setStyle("themeColor", "haloGreen");
  silenceLevel_pb.label = "Activity level: %3";
  } else {
  silenceLevel_pb.indeterminate = true;
  silenceLevel_pb.setStyle("themeColor", "0xFF0000");
  silenceLevel_pb.label = "Activity level: (inactive)";
  }
};
```

La méthode `MovieClip.getNextHighestDepth()` utilisée dans cet exemple nécessite Flash Player 7 ou une version ultérieure. Si votre fichier SWF comprend un composant de la version 2, utilisez les composants de la version 2 de la classe DepthManager au lieu de la méthode `MovieClip.getNextHighestDepth()`.

### Voir également

setSilenceLevel (méthode Microphone.setSilenceLevel)

## useEchoSuppression (propriété Microphone.useEchoSuppression)

```
public useEchoSuppression : Boolean
```

Propriété (lecture seule) ; valeur booléenne `true` si la suppression de l'écho est activée, `false` sinon. La valeur par défaut est `false` sauf si l'utilisateur sélectionne Réduire l'écho dans le panneau Paramètres du microphone de Flash Player.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

### Exemple

L'exemple suivant active la suppression de l'écho si l'utilisateur sélectionne une occurrence CheckBox appelée `useEchoSuppression_ch`. L'occurrence ProgressBar intitulée `activityLevel_pb` affiche le niveau d'activité actuel du flux continu.

```
var useEchoSuppression_ch:mx.controls.CheckBox;
var activityLevel_pb:mx.controls.ProgressBar;

this.createEmptyMovieClip("sound_mc", this.getNextHighestDepth());
var active_mic:Microphone = Microphone.get();
sound_mc.attachAudio(active_mic);

activityLevel_pb.mode = "manual";
activityLevel_pb.label = "Activity Level: %3";
useEchoSuppression_ch.selected = active_mic.useEchoSuppression;
this.onEnterFrame = function() {
  activityLevel_pb.setProgress(active_mic.activityLevel, 100);
};
var chListener:Object = new Object();
chListener.click = function(evt:Object) {
  active_mic.setUseEchoSuppression(evt.target.selected);
};
useEchoSuppression_ch.addEventListener("click", chListener);
```

La méthode `MovieClip.getNextHighestDepth()` utilisée dans cet exemple nécessite Flash Player 7 ou une version ultérieure. Si votre fichier SWF comprend un composant de la version 2, utilisez les composants de la version 2 de la classe DepthManager au lieu de la méthode `MovieClip.getNextHighestDepth()`.

### Voir également

setUseEchoSuppression (méthode microphone.setUseEchoSuppression)

# Souris

```
Object
  |
  +-Mouse
```

```
public class Mouse
extends Object
```

La classe Mouse est une classe de niveau supérieur dont les propriétés et les méthodes sont accessibles sans l'aide d'un constructeur. Vous pouvez utiliser les méthodes de la classe Mouse pour masquer et afficher le pointeur de la souris (curseur) dans le fichier SWF. Le pointeur de la souris est visible par défaut. Vous pouvez cependant le masquer et implémenter un curseur personnalisé créé à partir d'un clip.

Une application Flash ne peut contrôler que les événements de souris qui se produisent dans son focus. Une application Flash ne peut pas détecter les événements de souris qui se produisent dans une autre application.

**Disponibilité :** ActionScript 1.0 ; Flash Player 5

## Résumé des propriétés

*Propriétés héritées de la classe Object*

| |
|---|
| `constructeur (propriété Object.constructor), __proto__ (Object.__proto__ property), prototype (Object.prototype, propriété), __resolve (Object.__resolve, propriété)` |

## Résumé des événements

| Evénement | Description |
|---|---|
| `onMouseDown = function() {}` | Signalé lorsque vous cliquez sur la souris. |
| `onMouseMove = function() {}` | Signalé lorsque la souris bouge. |
| `onMouseUp = function() {}` | Signalé lorsque le bouton de la souris est relâché. |
| `onMouseWheel = function([delta:Number], [scrollTarget:String]) {}` | Signalé lorsque l'utilisateur actionne la molette de la souris. |

## Résumé de la méthode

| Modificateurs | Signature | Description |
|---|---|---|
| static | addListener(listener:Object) : Void | Enregistre un objet afin de recevoir les notifications des écouteurs onMouseDown, onMouseMove, onMouseUp, et onMouseWheel. |
| static | hide() : Number | Masque le pointeur dans un fichier SWF. |
| static | removeListener(listener:Object) : Boolean | Supprime un objet précédemment enregistré avec addListener(). |
| static | show() : Number | Affiche le pointeur de la souris dans un fichier SWF. |

*Méthodes héritées de la classe Object*

addProperty (méthode Object.addProperty), hasOwnProperty (méthode Object.hasOwnProperty), isPropertyEnumerable (méthode Object.isPropertyEnumerable), isPrototypeOf (méthode Object.isPrototypeOf), registerClass (méthode Object.registerClass), toString (méthode Object.toString), unwatch (méthode Object.unwatch), valueOf (méthode Object.valueOf), watch (méthode Object.watch)

# addListener (méthode Mouse.addListener)

```
public static addListener(listener:Object) : Void
```

Enregistre un objet afin de recevoir les notifications des écouteurs onMouseDown, onMouseMove, onMouseUp, et onMouseWheel. (L'écouteur onMouseWheel est pris en charge uniquement sous Windows.)

Le paramètre listener doit contenir un objet ayant une méthode définie pour au moins l'un des écouteurs.

Lorsque l'utilisateur clique sur le bouton de la souris, la déplace, relâche le bouton ou l'utilise pour faire défiler du texte, quel que soit le focus d'entrée, la méthode onMouseDown, onMouseMove, onMouseUp, ou onMouseWheel de tous les objets écouteur enregistrés auprès de cette méthode est appelée. Plusieurs objets peuvent écouter les notifications de souris. Si l'objet listener est déjà enregistré, aucun changement ne se produit.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

## Paramètres

*listener*:Object - Objet.

### Exemple

Cet exemple est tiré du fichier *animation.fla* figurant dans le dossier Samples ActionScript.

```
// Create a mouse listener object
var mouseListener:Object = new Object();

// Every time the mouse cursor moves within the SWF file,
  update the position of the crosshair movie clip
  instance on the Stage.
mouseListener.onMouseMove = function() {
  crosshair_mc._x = _xmouse;
  crosshair_mc._y = _ymouse;
};

// When you click the mouse, check to see if the cursor is within the
  boundaries of the Stage. If so, increment the number of shots.
mouseListener.onMouseDown = function() {
  if (bg_mc.hitTest(_xmouse, _ymouse, false)) {
  _global.shots++;
  }
};
Mouse.addListener(mouseListener);
```

Pour afficher l'ensemble du script, ouvrez le fichier *animation.fla* situé dans le dossier Samples ActionScrip. La liste suivante affiche les chemins types du dossier Samples ActionScript :

- Windows : *lecteur d'amorçage*\Program Files\Macromedia\Flash 8\Samples and Tutorials\Samples\ActionScript
- Macintosh : *disque dur Macintosh*/Applications/Macromedia Flash 8/Samples and Tutorials/Samples/ActionScript

### Voir également

`onMouseDown (écouteur d'événement Mouse.onMouseDown)`, `onMouseMove (écouteur d'événement Mouse.onMouseMove)`, `onMouseUp (écouteur d'événement Mouse.onMouseUp)`, `onMouseWheel (écouteur d'événement Mouse.onMouseWheel)`

## hide (méthode Mouse.hide)

```
public static hide() : Number
```

Masque le pointeur dans un fichier SWF. Le pointeur est visible par défaut.

**Disponibilité :** ActionScript 1.0 ; Flash Player 5

### Valeur renvoyée

`Number` - Entier ; `0` ou `1`. Si le pointeur de la souris était masqué avant d'appeler `Mouse.hide()`, alors la valeur de retour est `0`. Si le pointeur de la souris était visible avant d'appeler `Mouse.hide()` alors la valeur de retour est `1`.

### Exemple

Le code suivant masque le pointeur standard de la souris et définit les positions *x* et *y* de l'occurrence de clip `pointer_mc` sur les positions *x* et *y* du pointeur. Crée un clip et définit son identifiant Linkage sur `pointer_id`. Ajoute le code ActionScript suivant à l'image 1 du scénario :

```
// to use this script you need a symbol
// in your library with a Linkage Identifier of "pointer_id".
this.attachMovie("pointer_id", "pointer_mc", this.getNextHighestDepth());
Mouse.hide();
var mouseListener:Object = new Object();
mouseListener.onMouseMove = function() {
  pointer_mc._x = _xmouse;
  pointer_mc._y = _ymouse;
  updateAfterEvent();
};
Mouse.addListener(mouseListener);
```

La méthode `MovieClip.getNextHighestDepth()` utilisée dans cet exemple nécessite Flash Player 7 ou une version ultérieure. Si votre fichier SWF comprend un composant de la version 2, utilisez les composants de la version 2 de la classe DepthManager au lieu de la méthode `MovieClip.getNextHighestDepth()`.

### Voir également

`show (méthode Mouse.show)`, `_xmouse (propriété MovieClip._xmouse)`, `_ymouse (propriété MovieClip._ymouse)`

## onMouseDown (écouteur d'événement Mouse.onMouseDown)

```
onMouseDown = function() {}
```

Signalé lorsque vous cliquez sur la souris. Pour utiliser l'écouteur `onMouseDown`, vous devez créer un objet écouteur. Vous pouvez ensuite définir une fonction pour `onMouseDown` et utiliser `addListener()` pour enregistrer l'écouteur auprès de l'objet Mouse, comme indiqué dans le code suivant :

```
var someListener:Object = new Object();
someListener.onMouseDown = function () { ... };
```

```
Mouse.addListener(someListener);
```

Les écouteurs permettent à divers blocs de code de coopérer car plusieurs écouteurs peuvent recevoir une notification sur un événement unique.

Une application Flash ne peut contrôler que les événements de souris qui se produisent dans son focus. Une application Flash ne peut pas détecter les événements de souris qui se produisent dans une autre application.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

## Exemple

L'exemple suivant a recours à l'API de dessin pour dessiner un rectangle lorsque l'utilisateur clique avec la souris, fait glisser le pointeur et relâche le bouton pendant la période d'exécution.

```
this.createEmptyMovieClip("canvas_mc", this.getNextHighestDepth());
var mouseListener:Object = new Object();
mouseListener.onMouseDown = function() {
  this.isDrawing = true;
  this.orig_x = _xmouse;
  this.orig_y = _ymouse;
  this.target_mc = canvas_mc.createEmptyMovieClip("",
  canvas_mc.getNextHighestDepth());
};
mouseListener.onMouseMove = function() {
  if (this.isDrawing) {
    this.target_mc.clear();
    this.target_mc.lineStyle(1, 0xFF0000, 100);
    this.target_mc.moveTo(this.orig_x, this.orig_y);
    this.target_mc.lineTo(_xmouse, this.orig_y);
    this.target_mc.lineTo(_xmouse, _ymouse);
    this.target_mc.lineTo(this.orig_x, _ymouse);
    this.target_mc.lineTo(this.orig_x, this.orig_y);
  }
  updateAfterEvent();
};
mouseListener.onMouseUp = function() {
  this.isDrawing = false;
};
Mouse.addListener(mouseListener);
```

La méthode `MovieClip.getNextHighestDepth()` utilisée dans cet exemple nécessite Flash Player 7 ou une version ultérieure. Si votre fichier SWF comprend un composant de la version 2, utilisez les composants de la version 2 de la classe DepthManager au lieu de la méthode `MovieClip.getNextHighestDepth()`.

### Voir également

addListener (méthode Mouse.addListener)

## onMouseMove (écouteur d'événement Mouse.onMouseMove)

```
onMouseMove = function() {}
```

Signalé lorsque la souris bouge. Pour utiliser l'écouteur `onMouseMove` vous devez créer un objet d'écoute Vous pouvez ensuite définir une fonction pour `onMouseMove` et utiliser `addListener()` pour enregistrer l'écouteur auprès de l'objet Mouse, comme indiqué dans le code suivant :

```
var someListener:Object = new Object();
someListener.onMouseMove = function () { ... };
Mouse.addListener(someListener);
```

Les écouteurs permettent à divers blocs de code de coopérer car plusieurs écouteurs peuvent recevoir une notification sur un événement unique.

Une application Flash ne peut contrôler que les événements de souris qui se produisent dans son focus. Une application Flash ne peut pas détecter les événements de souris qui se produisent dans une autre application.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

### Exemple

L'exemple suivant utilise le pointeur de la souris en tant qu'outil pour dessiner des lignes avec `onMouseMove` et l'API de dessin. L'utilisateur trace une ligne en faisant glisser le pointeur de la souris.

```
this.createEmptyMovieClip("canvas_mc", this.getNextHighestDepth());
var mouseListener:Object = new Object();
mouseListener.onMouseDown = function() {
  this.isDrawing = true;
  canvas_mc.lineStyle(2, 0xFF0000, 100);
  canvas_mc.moveTo(_xmouse, _ymouse);
};
mouseListener.onMouseMove = function() {
  if (this.isDrawing) {
    canvas_mc.lineTo(_xmouse, _ymouse);
  }
  updateAfterEvent();
};
mouseListener.onMouseUp = function() {
  this.isDrawing = false;
};
```

```
Mouse.addListener(mouseListener);
```

La méthode `MovieClip.getNextHighestDepth()` utilisée dans cet exemple nécessite Flash Player 7 ou une version ultérieure. Si votre fichier SWF comprend un composant de la version 2, utilisez les composants de la version 2 de la classe DepthManager au lieu de la méthode `MovieClip.getNextHighestDepth()`.

L'exemple suivant masque le pointeur standard de la souris et définit les positions *x* et *y* de l'occurrence de clip `pointer_mc` sur les positions *x* et *y* du pointeur. Crée un clip et définit son identifiant Linkage sur `pointer_id`. Ajoute le code ActionScript suivant à l'image 1 du scénario :

```
// to use this script you need a symbol
// in your library with a Linkage Identifier of "pointer_id".
this.attachMovie("pointer_id", "pointer_mc", this.getNextHighestDepth());
Mouse.hide();
var mouseListener:Object = new Object();
mouseListener.onMouseMove = function() {
  pointer_mc._x = _xmouse;
  pointer_mc._y = _ymouse;
  updateAfterEvent();
};
Mouse.addListener(mouseListener);
```

La méthode `MovieClip.getNextHighestDepth()` utilisée dans cet exemple nécessite Flash Player 7 ou une version ultérieure. Si votre fichier SWF comprend un composant de la version 2, utilisez les composants de la version 2 de la classe DepthManager au lieu de la méthode `MovieClip.getNextHighestDepth()`.

### Voir également

`addListener (méthode Mouse.addListener)`

## onMouseUp (écouteur d'événement Mouse.onMouseUp)

```
onMouseUp = function() {}
```

Signalé lorsque le bouton de la souris est relâché. Pour utiliser l'écouteur `onMouseUp` vous devez créer un objet d'écoute Vous pouvez ensuite définir une fonction pour `onMouseUp` et utiliser `addListener()` pour enregistrer l'écouteur auprès de l'objet Mouse, comme indiqué dans le code suivant :

```
var someListener:Object = new Object();
someListener.onMouseUp = function () { ... };
Mouse.addListener(someListener);
```

Les écouteurs permettent à divers blocs de code de coopérer car plusieurs écouteurs peuvent recevoir une notification sur un événement unique.

Une application Flash ne peut contrôler que les événements de souris qui se produisent dans son focus. Une application Flash ne peut pas détecter les événements de souris qui se produisent dans une autre application.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

## Exemple

L'exemple suivant utilise le pointeur de la souris en tant qu'outil pour dessiner des lignes avec `onMouseMove` et l'API de dessin. L'utilisateur trace une ligne en faisant glisser le pointeur de la souris. Il arrête le tracé en relâchant le bouton de la souris.

```
this.createEmptyMovieClip("canvas_mc", this.getNextHighestDepth());
var mouseListener:Object = new Object();
mouseListener.onMouseDown = function() {
  this.isDrawing = true;
  canvas_mc.lineStyle(2, 0xFF0000, 100);
  canvas_mc.moveTo(_xmouse, _ymouse);
};
mouseListener.onMouseMove = function() {
  if (this.isDrawing) {
    canvas_mc.lineTo(_xmouse, _ymouse);
  }
  updateAfterEvent();
};
mouseListener.onMouseUp = function() {
  this.isDrawing = false;
};
Mouse.addListener(mouseListener);
```

La méthode `MovieClip.getNextHighestDepth()` utilisée dans cet exemple nécessite Flash Player 7 ou une version ultérieure. Si votre fichier SWF comprend un composant de la version 2, utilisez les composants de la version 2 de la classe DepthManager au lieu de la méthode `MovieClip.getNextHighestDepth()`.

## Voir également

`addListener (méthode Mouse.addListener)`

# onMouseWheel (écouteur d'événement Mouse.onMouseWheel)

```
onMouseWheel = function([delta:Number], [scrollTarget:String]) {}
```

Signalé lorsque l'utilisateur actionne la molette de la souris. Pour utiliser l'écouteur `onMouseWheel` vous devez créer un objet d'écoute Vous pouvez ensuite définir une fonction pour `onMouseWheel` et utiliser `addListener()` pour enregistrer l'écouteur auprès de l'objet Mouse.

*Remarque :* Les écouteurs d'événements relatifs à la molette de la souris sont disponibles uniquement sur les versions Windows de Flash Player.

Une application Flash ne peut contrôler que les événements de souris qui se produisent dans son focus. Une application Flash ne peut pas détecter les événements de souris qui se produisent dans une autre application.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

## Paramètres

*`delta`*`:Number` [facultatif] - Nombre indiquant combien de lignes il convient de faire défiler chaque fois que l'utilisateur fait tourner la molette de la souris. Une valeur `delta` positive indique un défilement vers le haut ; une valeur négative indique un défilement vers le bas. Les valeurs types sont comprises entre 1 et 3 ; un défilement plus rapide peut générer des valeurs supérieures.

*`scrollTarget`*`:String` [facultatif] - Paramètre indiquant l'occurrence de clip supérieure située sous le pointeur de la souris lorsque la molette est actionnée. Si vous souhaitez spécifier une valeur pour `scrollTarget` uniquement, mais pas pour `delta`, transmettez la valeur `null` à `delta`.

## Exemple

L'exemple suivant indique comment créer un objet listener qui réagisse aux événements de la molette de la souris. Dans cet exemple, la coordonnée *x* d'un objet clip appelé `clip_mc` change dès que l'utilisateur utilise la molette de la souris :

```
var mouseListener:Object = new Object();
mouseListener.onMouseWheel = function(delta) {
  clip_mc._x += delta;
}
Mouse.addListener(mouseListener);
```

L'exemple suivant trace une ligne qui pivote en même temps que la molette de la souris. Cliquez sur le fichier SWF pendant la période d'exécution, puis faites tourner la molette de la souris pour activer le clip.

```
this.createEmptyMovieClip("line_mc", this.getNextHighestDepth());
line_mc.lineStyle(2, 0xFF0000, 100);
line_mc.moveTo(0, 100);
line_mc.lineTo(0, 0);
line_mc._x = 200;
line_mc._y = 200;

var mouseListener:Object = new Object();
mouseListener.onMouseWheel = function(delta:Number) {
  line_mc._rotation += delta;
};
mouseListener.onMouseDown = function() {
  trace("Down");
};
Mouse.addListener(mouseListener);
```

La méthode `MovieClip.getNextHighestDepth()` utilisée dans cet exemple nécessite Flash Player 7 ou une version ultérieure. Si votre fichier SWF comprend un composant de la version 2, utilisez les composants de la version 2 de la classe DepthManager au lieu de la méthode `MovieClip.getNextHighestDepth()`.

### Voir également

`addListener (méthode Mouse.addListener)`, `mouseWheelEnabled (propriété TextField.mouseWheelEnabled)`

## removeListener (méthode Mouse.removeListener)

```
public static removeListener(listener:Object) : Boolean
```

Supprime un objet précédemment enregistré avec `addListener()`.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

### Paramètres

*`listener`*`:Object` - Objet.

### Valeur renvoyée

`Boolean` - Si l'objet `listener` est supprimé avec succès, la méthode renvoie `true` ; si l'objet `listener` n'a pas été supprimé (par exemple, si `listener` ne figurait pas dans la liste des écouteurs de l'objet Mouse), la méthode renvoie `false`.

## Exemple

L'exemple suivant associe trois boutons à la scène et permet à l'utilisateur de tracer des lignes dans le fichier SWF pendant la période d'exécution et avec le pointeur de la souris. Un bouton efface l'ensemble des lignes du fichier SWF. Le deuxième bouton supprime l'écouteur de l'objet Mouse, de façon à empêcher l'utilisateur de tracer des lignes. Le troisième bouton ajoute l'écouteur de l'objet Mouse après sa suppression, de façon à ce que l'utilisateur puisse de nouveau tracer des lignes. Ajoute le code ActionScript suivant à l'image 1 du scénario :

```
this.createClassObject(mx.controls.Button, "clear_button",
  this.getNextHighestDepth(), {_x:10, _y:10, label:'clear'});
this.createClassObject(mx.controls.Button, "stopDrawing_button",
  this.getNextHighestDepth(), {_x:120, _y:10, label:'stop drawing'});
this.createClassObject(mx.controls.Button, "startDrawing_button",
  this.getNextHighestDepth(), {_x:230, _y:10, label:'start drawing'});
startDrawing_button.enabled = false;
//
this.createEmptyMovieClip("canvas_mc", this.getNextHighestDepth());
var mouseListener:Object = new Object();
mouseListener.onMouseDown = function() {
  this.isDrawing = true;
  canvas_mc.lineStyle(2, 0xFF0000, 100);
  canvas_mc.moveTo(_xmouse, _ymouse);
};
mouseListener.onMouseMove = function() {
  if (this.isDrawing) {
  canvas_mc.lineTo(_xmouse, _ymouse);
  }
  updateAfterEvent();
};
mouseListener.onMouseUp = function() {
  this.isDrawing = false;
};
Mouse.addListener(mouseListener);
var clearListener:Object = new Object();
clearListener.click = function() {
  canvas_mc.clear();
};
clear_button.addEventListener("click", clearListener);
//
var stopDrawingListener:Object = new Object();
stopDrawingListener.click = function(evt:Object) {
  Mouse.removeListener(mouseListener);
  evt.target.enabled = false;
  startDrawing_button.enabled = true;
};
stopDrawing_button.addEventListener("click", stopDrawingListener);
var startDrawingListener:Object = new Object();
startDrawingListener.click = function(evt:Object) {
  Mouse.addListener(mouseListener);
```

```
  evt.target.enabled = false;
  stopDrawing_button.enabled = true;
};
startDrawing_button.addEventListener("click", startDrawingListener);
```

La méthode `MovieClip.getNextHighestDepth()` utilisée dans cet exemple nécessite Flash Player 7 ou une version ultérieure. Si votre fichier SWF comprend un composant de la version 2, utilisez les composants de la version 2 de la classe DepthManager au lieu de la méthode `MovieClip.getNextHighestDepth()`.

# show (méthode Mouse.show)

```
public static show() : Number
```

Affiche le pointeur de la souris dans un fichier SWF. Le pointeur est visible par défaut.

**Disponibilité :** ActionScript 1.0 ; Flash Player 5

## Valeur renvoyée

`Number` - Entier ; `0` ou `1`. Si le pointeur de la souris était masqué avant d'appeler `Mouse.show()`, alors la valeur de retour est `0`. Si le pointeur de la souris était visible avant d'appeler `Mouse.show()` alors la valeur de retour est `1`.

## Exemple

L'exemple suivant associe un curseur personnalisé provenant de la bibliothèque lorsque le pointeur de la souris passe sur un clip appelé `my_mc`. Associez à un clip de la bibliothèque l'identifiant de liaison `cursor_help_id`, et ajoutez le code ActionScript suivant à l'image 1 du scénario :

```
my_mc.onRollOver = function() {
  Mouse.hide();
  this.attachMovie("cursor_help_id", "cursor_mc",
  this.getNextHighestDepth(), {_x:this._xmouse, _y:this._ymouse});
};
my_mc.onMouseMove = function() {
  this.cursor_mc._x = this._xmouse;
  this.cursor_mc._y = this._ymouse;
};
my_mc.onRollOut = function() {
  Mouse.show();
  this.cursor_mc.removeMovieClip();
};
```

La méthode `MovieClip.getNextHighestDepth()` utilisée dans cet exemple nécessite Flash Player 7 ou une version ultérieure. Si votre fichier SWF comprend un composant de la version 2, utilisez les composants de la version 2 de la classe DepthManager au lieu de la méthode `MovieClip.getNextHighestDepth()`.

### Voir également

`hide (méthode Mouse.hide)`, `_xmouse (propriété MovieClip._xmouse)`, `_ymouse (propriété MovieClip._ymouse)`

# MovieClip

```
Object
  |
  +-MovieClip
```

`public dynamic class` ***MovieClip***
`extends Object`

Les méthodes de la classe MovieClip fournissent les mêmes fonctionnalités que les actions permettant de cibler des clips. Certaines méthodes supplémentaires ne possèdent pas d'actions équivalentes dans la boîte à outils Actions du panneau Actions.

Aucune méthode constructeur n'est requise pour créer un clip. Vous disposez de trois méthodes pour créer des occurrences de clip :

- La méthode `attachMovie()` permet de créer une occurrence de clip en fonction d'un symbole de clip provenant de la bibliothèque.
- La méthode `createEmptyMovieClip()` permet de créer une occurrence de clip vide en tant qu'enfant reposant sur un autre clip.
- La méthode `duplicateMovieClip()` permet de créer une occurrence de clip à partir d'un autre clip.

Pour appeler les méthodes de la classe MovieClip, vous devez référencer les occurrences de clip en fonction de leur nom, en appliquant la syntaxe suivante, où *`my_mc`* correspond à une occurrence de clip :

```
my_mc.play();
my_mc.gotoAndPlay(3);
```

Vous pouvez étendre les méthodes et les gestionnaires d'événements de la classe MovieClip en créant une sous-classe.

**Disponibilité :** ActionScript 1.0 ; Flash Player 3

Résumé des propriétés

| Modificateurs | Propriété | Description |
|---|---|---|
| | `_alpha:Number` | Valeur de transparence alpha du clip. |
| | `blendMode:Object` | Mode de fondu de ce clip. |
| | `cacheAsBitmap:Boolean` | S'il a pour valeur `true`, Flash Player place en mémoire cache une représentation bitmap interne du clip. |
| | `_currentframe:Number` [lecture seule] | Renvoie le numéro de l'image dans laquelle se trouve la tête de lecture dans le scénario du clip. |
| | `_droptarget:String` [lecture seule] | Renvoie le chemin absolu, en utilisant une notation de syntaxe à barre oblique, de l'occurrence de clip sur laquelle ce clip a été déposé. |
| | `enabled:Boolean` | Valeur booléenne indiquant si un clip est activé. |
| | `filters:Array` | Tableau indexé contenant tous les objets filtre associés au clip. |
| | `focusEnabled:Boolean` | Si la valeur est `undefined` ou `false`, un clip ne peut pas recevoir le focus d'entrée sauf s'il s'agit d'un bouton. |
| | `_focusrect:Boolean` | Valeur booléenne indiquant si un clip est entouré d'un rectangle jaune lorsqu'il a le focus clavier. |
| | `_framesloaded:Number` [lecture seule] | Nombre d'images à charger à partir d'un fichier SWF en diffusion continue. |
| | `_height:Number` | Hauteur du clip, en pixels. |
| | `_highquality:Number` | *Déconseillé* à partir de Flash Player 7. Il est recommandé d'utiliser `MovieClip._quality`. Spécifie le niveau d'anti-aliasing appliqué au fichier SWF actuel. |
| | `hitArea:Object` | Désigne un autre clip pour faire office de zone active d'un clip. |
| | `_lockroot:Boolean` | Valeur booléenne qui spécifie ce à quoi `_root` se réfère lorsqu'un fichier SWF est chargé dans un clip. |
| | `menu:ContextMenu` | Associe l'objet ContextMenu spécifié au clip. |
| | `_name:String` | Nom d'occurrence du clip. |
| | `opaqueBackground:Number` | Couleur de l'arrière-plan opaque (non transparent) du clip, spécifiée par un nombre (une valeur RVB hexadécimale). |

| Modificateurs | Propriété | Description |
|---|---|---|
| | `_parent:MovieClip` | Référence au clip ou à l'objet contenant le clip ou l'objet actuel. |
| | `_quality:String` | Définit ou extrait la qualité du rendu appliqué à un fichier SWF. |
| | `_rotation:Number` | Spécifie la rotation du clip, en degrés, à partir de son orientation d'origine. |
| | `scale9Grid:Rectangle` | La zone rectangulaire qui définit les neuf zones de redimensionnement du clip. |
| | `scrollRect:Object` | La propriété `scrollRect` permet de parcourir rapidement le contenu du clip et d'ouvrir une fenêtre plus grande pour afficher davantage de contenu. |
| | `_soundbuftime:Number` | Spécifie le nombre de secondes pendant lequel les sons sont chargés en mémoire tampon avant d'être diffusés en continu. |
| | `tabChildren:Boolean` | Détermine si les enfants d'un clip sont inclus dans l'ordre de tabulation automatique. |
| | `tabEnabled:Boolean` | Spécifie si le clip est inclus dans l'ordre de tabulation automatique. |
| | `tabIndex:Number` | Permet de personnaliser l'ordre de tabulation des objets dans un clip. |
| | `_target:String` [lecture seule] | Renvoie le chemin cible de l'occurrence de clip, en notation avec barre oblique. |
| | `_totalframes:Number` [lecture seule] | Renvoie le nombre total d'images dans l'occurrence de clip spécifiée par le paramètre `MovieClip`. |
| | `trackAsMenu:Boolean` | Valeur booléenne indiquant si d'autres boutons ou clips peuvent recevoir des événements de relâchement de souris. |
| | `transform:Transform` | Un objet avec des propriétés se rapportant à la matrice d'un clip, à la transformation des couleurs et aux limites des pixels. |
| | `_url:String` [lecture seule] | Récupère l'URL du fichier SWF, JPEG, GIF ou PNG ayant servi à télécharger le clip. |
| | `useHandCursor:Boolean` | Valeur booléenne indiquant si le curseur en forme de main apparaît lorsque la souris passe sur un clip. |
| | `_visible:Boolean` | Valeur booléenne indiquant si le clip est visible. |
| | `_width:Number` | Largeur du clip, en pixels. |

| Modificateurs | Propriété | Description |
|---|---|---|
| | `_x:Number` | Entier qui définit la coordonnée *x* d'un clip par rapport aux coordonnées locales du clip parent. |
| | `_xmouse:Number` [lecture seule] | Renvoie la coordonnée *x* de la position de la souris. |
| | `_xscale:Number` | Détermine le redimensionnement horizontal du clip (*`percentage`*) tel qu'il est appliqué à partir du point d'alignement du clip. |
| | `_y:Number` | Définit la coordonnée *y* d'un clip par rapport aux coordonnées locales du clip parent. |
| | `_ymouse:Number` [lecture seule] | Renvoie la coordonnée *y* de la position de la souris. |
| | `_yscale:Number` | Détermine le redimensionnement vertical du clip (*`percentage`*) tel qu'il est appliqué à partir du point d'alignement du clip. |

*Propriétés héritées de la classe Object*

`constructeur (propriété Object.constructor), __proto__ (Object.__proto__ property), prototype (Object.prototype, propriété), __resolve (Object.__resolve, propriété)`

## Résumé des événements

| Evénement | Description |
|---|---|
| `onData = function() {}` | Invoqué lorsqu'un clip reçoit des données d'un appel `MovieClip.loadVariables()` ou d'un appel `MovieClip.loadMovie()`. |
| `onDragOut = function() {}` | Appelé lorsque l'utilisateur appuie sur le bouton de la souris et que le pointeur se déplace hors de l'objet. |
| `onDragOver = function() {}` | Appelé lorsque l'utilisateur fait glisser le pointeur hors du clip, puis sur le clip. |
| `onEnterFrame = function() {}` | Appelé à plusieurs reprises à la cadence du fichier SWF. |
| `onKeyDown = function() {}` | Appelé lorsqu'un clip reçoit le focus d'entrée et que l'utilisateur appuie sur une touche. |
| `onKeyUp = function() {}` | Appelé lorsqu'une touche est relâchée. |

| Evénement | Description |
| --- | --- |
| `onKillFocus = function(newFocus:Object) {}` | Appelé lorsqu'un clip perd le focus clavier. |
| `onLoad = function() {}` | Appelé lorsque le clip est instancié et apparaît dans le scénario. |
| `onMouseDown = function() {}` | Appelé lorsque l'utilisateur appuie sur le bouton de la souris. |
| `onMouseMove = function() {}` | Appelé lorsque la souris bouge. |
| `onMouseUp = function() {}` | Appelé lorsque l'utilisateur relâche le bouton de la souris. |
| `onPress = function() {}` | Appelé lorsque l'utilisateur clique sur le bouton de la souris quand le pointeur est placé sur un clip. |
| `onRelease = function() {}` | Appelé lorsqu'un utilisateur relâche le bouton de la souris sur un clip. |
| `onReleaseOutside = function() {}` | Appelé lorsqu'un utilisateur appuie sur le bouton de la souris dans une zone de clip et le relâche ensuite en dehors de la zone de clip. |
| `onRollOut = function() {}` | Appelé lorsqu'un utilisateur déplace le pointeur hors d'une zone de clip. |
| `onRollOver = function() {}` | Appelé lorsqu'un utilisateur déplace le pointeur sur une zone de clip. |
| `onSetFocus = function(oldFocus:Object) {}` | Appelé lorsqu'un clip reçoit le focus clavier. |
| `onUnload = function() {}` | Appelé dans la première image une fois la suppression du clip dans le scénario effectuée. |

Résumé de la méthode

| Modificateurs | Signature | Description |
|---|---|---|
| | `attachAudio(id:Object) : Void` | Spécifie la source audio à jouer. |
| | `attachBitmap(bmp:BitmapData, depth:Number, [pixelSnapping:String], [smoothing:Boolean]) : Void` | Associe une image bitmap à un clip. |
| | `attachMovie(id:String, name:String, depth:Number, [initObject:Object]) : MovieClip` | Sélectionne un symbole dans la bibliothèque et l'associe au clip. |
| | `beginBitmapFill(bmp:BitmapData, [matrix:Matrix], [repeat:Boolean], [smoothing:Boolean]) : Void` | Remplit une zone de dessin avec une image bitmap. |
| | `beginFill(rgb:Number, [alpha:Number]) : Void` | Indique le début d'un nouveau chemin de dessin. |
| | `beginGradientFill(fillType:String, colors:Array, alphas:Array, ratios:Array, matrix:Object, [spreadMethod:String], [interpolationMethod:String], [focalPointRatio:Number]) : Void` | Indique le début d'un nouveau chemin de dessin. |
| | `clear() : Void` | Supprime tous les graphiques créés lors de l'exécution à l'aide des méthodes de dessin de clips, y compris les styles de trait spécifiés par `MovieClip.lineStyle()`. |

| Modificateurs | Signature | Description |
|---|---|---|
| | `createEmptyMovieClip (name:String, depth:Number) : MovieClip` | Crée un clip vide en tant qu'enfant d'un clip existant. |
| | `createTextField(instanceName:String, depth:Number, x:Number, y:Number, width:Number, height:Number) : TextField` | Crée un nouveau champ de texte vide en tant qu'enfant du clip pour lequel vous avez appelé cette méthode. |
| | `curveTo(controlX:Number, controlY:Number, anchorX:Number, anchorY:Number) : Void` | Dessine une courbe en utilisant le style de ligne actuel de la position actuelle à (`anchorX`, `anchorY`) en utilisant le point de contrôle spécifié par (`(controlX`, `controlY`). |
| | `duplicateMovieClip(name:String, depth:Number, [initObject:Object]) : MovieClip` | Crée une occurrence du clip spécifié lors de la lecture du fichier SWF. |
| | `endFill() : Void` | Applique un remplissage aux lignes et aux courbes ajoutées depuis le dernier appel de `beginFill()` ou `beginGradientFill()`. |
| | `getBounds(bounds:Object) : Object` | Renvoie des propriétés dont les coordonnées *x* et *y* sont les valeurs minimales et maximales du clip, à partir du paramètre `bounds`. |
| | `getBytesLoaded() : Number` | Renvoie le nombre d'octets déjà chargés (transmis en continu) pour le clip. |
| | `getBytesTotal() : Number` | Renvoie la taille, en octets, du clip. |
| | `getDepth() : Number` | Renvoie la profondeur d'une occurrence de clip. |
| | `getInstanceAtDepth(depth:Number) : MovieClip` | Permet de déterminer si une profondeur spécifique est déjà occupée par un clip. |

| Modificateurs | Signature | Description |
|---|---|---|
| | `getNextHighestDepth() : Number` | Permet de déterminer une valeur de profondeur que vous pouvez transmettre à `MovieClip.attachMovie()`, `MovieClip.duplicateMovieClip()`, ou `MovieClip.createEmptyMovieClip()` afin de vous assurer que Flash rende le clip devant tous les autres objets sur les mêmes niveau et calque dans le clip actuel. |
| | `getRect(bounds:Object) : Object` | Renvoie des propriétés dont les coordonnées *x* et *y* sont les valeurs minimales et maximales du clip, à partir du paramètre `bounds`, ce qui exclut tout tracé sur les formes. |
| | `getSWFVersion() : Number` | Renvoie un entier indiquant la version de Flash Player pour laquelle le clip a été publié. |
| | `getTextSnapshot() : TextSnapshot` | Renvoie un objet TextSnapshot contenant le texte de tous les champs de texte statiques contenus dans le clip spécifié ; le texte des clips enfants n'est pas inclus. |
| | `getURL(url:String, [window:String], [method:String]) : Void` | Charge un document à partir de l'URL spécifiée dans la fenêtre spécifiée. |
| | `globalToLocal(pt:Object) : Void` | Convertit l'objet `pt` à partir des coordonnées de scène (globales) vers les coordonnées du clip (locales). |
| | `gotoAndPlay(frame:Object) : Void` | Commence la lecture du fichier SWF sur l'image spécifiée. |
| | `gotoAndStop(frame:Object) : Void` | Place la tête de lecture au niveau de l'image spécifiée du clip et l'arrête à cet endroit. |
| | `hitTest() : Boolean` | Evalue le clip pour savoir s'il recouvre ou recoupe la zone active identifiée par les paramètres de coordonnées `target` ou `x` et `y`. |

| Modificateurs | Signature | Description |
|---|---|---|
| | `lineGradientStyle(fillType:String, colors:Array, alphas:Array, ratios:Array, matrix:Object, [spreadMethod:String], [interpolationMethod:String], [focalPointRatio:Number]) : Void` | Spécifie un style de trait utilisé par Flash pour les prochains appels des méthodes `lineTo()` et `curveTo()`, jusqu'à ce que vous appeliez la méthode `lineStyle()` ou la méthode `lineGradientStyle()` avec des paramètres différents. |
| | `lineStyle(thickness:Number, rgb:Number, alpha:Number, pixelHinting:Boolean, noScale:String, capsStyle:String, jointStyle:String, miterLimit:Number) : Void` | Spécifie un style de trait utilisé par Flash pour les prochains appels des méthodes `lineTo()` et `curveTo()` jusqu'à ce que vous appeliez la méthode `lineStyle()` avec des paramètres différents. |
| | `lineTo(x:Number, y:Number) : Void` | Trace une ligne en utilisant le style de trait actuel de la position de dessin actuelle à (x, y) ; la position de dessin actuelle est ensuite définie sur (x, y). |
| | `loadMovie(url:String, [method:String]) : Void` | Charge un fichier SWF, JPEG, GIF ou PNG dans un clip Flash Player lors de la lecture du fichier SWF d'origine. |
| | `loadVariables(url:String, [method:String]) : Void` | Lit les données à partir d'un fichier externe et définit les valeurs des variables dans le clip. |
| | `localToGlobal(pt:Object) : Void` | Convertit l'objet `pt` à partir des coordonnées du clip (locales) vers les coordonnées de la scène (globales). |
| | `moveTo(x:Number, y:Number) : Void` | Déplace la position de dessin actuelle vers (x, y). |
| | `nextFrame() : Void` | Place la tête de lecture sur l'image suivante et l'arrête. |
| | `play() : Void` | Déplace la tête de lecture dans le scénario du clip. |
| | `prevFrame() : Void` | Place la tête de lecture sur l'image précédente et l'arrête. |

| Modificateurs | Signature | Description |
|---|---|---|
| | `removeMovieClip() : Void` | Supprime une occurrence de clip créée avec `duplicateMovieClip()`, `MovieClip.duplicateMovieClip()`, `MovieClip.createEmptyMovieClip()`, ou `MovieClip.attachMovie()`. |
| | `setMask(mc:Object) : Void` | Définit le clip du paramètre `mc` comme étant un masque qui révèle le clip appelant. |
| | `startDrag([lockCenter:Boolean], [left:Number], [top:Number], [right:Number], [bottom:Number]) : Void` | Permet à l'utilisateur de faire glisser le clip spécifié. |
| | `stop() : Void` | Arrête le clip en cours de lecture. |
| | `stopDrag() : Void` | Termine une méthode `MovieClip.startDrag()`. |
| | `swapDepths(target:Object) : Void` | Intervertit l'empilement, ou le niveau de profondeur (ordre z), de ce clip avec le clip spécifié par le paramètre `target` ou avec le clip qui occupe actuellement le niveau de profondeur spécifié dans le paramètre `target`. |
| | `unloadMovie() : Void` | Supprime le contenu d'une occurrence de clip. |

*Méthodes héritées de la classe Object*

`addProperty (méthode Object.addProperty), hasOwnProperty (méthode Object.hasOwnProperty), isPropertyEnumerable (méthode Object.isPropertyEnumerable), isPrototypeOf (méthode Object.isPrototypeOf), registerClass (méthode Object.registerClass), toString (méthode Object.toString), unwatch (méthode Object.unwatch), valueOf (méthode Object.valueOf), watch (méthode Object.watch)`

# _alpha (propriété MovieClip._alpha)

```
public _alpha : Number
```

La valeur de transparence alpha du clip. Les valeurs possibles sont comprises entre 0 (entièrement transparent) et 100 (entièrement opaque). La valeur par défaut est 100. Les objets d'un clip dont la propriété `_alpha` est définie sur 0 sont actifs, même s'ils sont invisibles. Par exemple, vous pouvez toujours cliquer sur un bouton dans un clip dont la propriété `_alpha` est définie sur 0. Pour désactiver le bouton entièrement, vous pouvez définir la propriété `_visible` du clip sur false.

Vous pouvez étendre les méthodes et les gestionnaires d'événements de la classe MovieClip en créant une sous-classe.

**Disponibilité :** ActionScript 1.0 ; Flash Player 4

## Exemple

Le code suivant définit la propriété `_alpha` d'un clip créé de façon dynamique et appelé `triangle`, sur 50 % lorsque la souris le survole. Ajoutez le code ActionScript suivant à votre fichier FLA ou AS :

```
this.createEmptyMovieClip("triangle", this.getNextHighestDepth());

triangle.beginFill(0x0000FF, 100);
triangle.moveTo(10, 10);
triangle.lineTo(10, 100);
triangle.lineTo(100, 10);
triangle.lineTo(10, 10);

triangle.onRollOver = function() {
  this._alpha = 50;
};
triangle.onRollOut = function() {
  this._alpha = 100;
};
```

La méthode `MovieClip.getNextHighestDepth()` utilisée dans cet exemple nécessite Flash Player 7 ou une version ultérieure. Si votre fichier SWF comprend un composant de la version 2, utilisez les composants de la version 2 de la classe DepthManager au lieu de la méthode `MovieClip.getNextHighestDepth()`.

## Voir également

`_alpha (propriété Button._alpha)`, `_alpha (propriété TextField._alpha)`, `_visible (propriété MovieClip._visible)`

# attachAudio (méthode MovieClip.attachAudio)

```
public attachAudio(id:Object) : Void
```

Spécifie la source audio à jouer. Pour arrêter la lecture de la source audio, transmettez la valeur `false` pour `id`.

Vous pouvez étendre les méthodes et les gestionnaires d'événements de la classe MovieClip en créant une sous-classe.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

## Paramètres

*`id`*`:Object` - Objet qui contient les données audio à lire. Les valeurs possibles sont un objet Microphone, un objet NetStream permettant de lire un fichier FLV et `false` (arrête la lecture des données audio).

## Exemple

L'exemple suivant crée une nouvelle connexion NetStream. Ajoutez un nouveau symbole vidéo en ouvrant le panneau Bibliothèque et en sélectionnant Nouvelle vidéo dans le menu d'options de la bibliothèque. Donnez au symbole le nom d'occurrence `my_video`. Chargez de façon dynamique la vidéo FLV pendant l'exécution. Utilisez la méthode `attachAudio()` pour associer le fichier audio du fichier FLV à un clip sur la scène. Vous pouvez ensuite contrôler les données audio du clip avec la classe Sound et deux boutons appelés `volUp_btn` et `volDown_btn`.

```
var my_nc:NetConnection = new NetConnection();
my_nc.connect(null);
var my_ns:NetStream = new NetStream(my_nc);
my_video.attachVideo(my_ns);
my_ns.play("yourVideo.flv");
this.createEmptyMovieClip("flv_mc", this.getNextHighestDepth());
flv_mc.attachAudio(my_ns);
var audio_sound:Sound = new Sound(flv_mc);

// Add volume buttons.
volUp_btn.onRelease = function() {
  if (audio_sound.getVolume()<100) {
  audio_sound.setVolume(audio_sound.getVolume()+10);
  updateVolume();
  }
};
volDown_btn.onRelease = function() {
  if (audio_sound.getVolume()>0) {
  audio_sound.setVolume(audio_sound.getVolume()-10);
  updateVolume();
  }
```

```
};

// Updates the volume.
this.createTextField("volume_txt", this.getNextHighestDepth(), 0, 0, 100,
  22);
updateVolume();
function updateVolume() {
  volume_txt.text = "Volume: "+audio_sound.getVolume();
}
```

L'exemple suivant spécifie un microphone en tant que source audio pour une occurrence de clip créée de façon dynamique et appelée `audio_mc`:

```
var active_mic:Microphone = Microphone.get();
this.createEmptyMovieClip("audio_mc", this.getNextHighestDepth());
audio_mc.attachAudio(active_mic);
```

La méthode `MovieClip.getNextHighestDepth()` utilisée dans cet exemple nécessite Flash Player 7 ou une version ultérieure. Si votre fichier SWF comprend un composant de la version 2, utilisez les composants de la version 2 de la classe DepthManager au lieu de la méthode `MovieClip.getNextHighestDepth()`.

### Voir également

`Microphone`, `play (méthode NetStream.play)`, `Sound`, `attachVideo (méthode Video.attachVideo)`

## attachBitmap (méthode MovieClip.attachBitmap)

```
public attachBitmap(bmp:BitmapData, depth:Number, [pixelSnapping:String],
  [smoothing:Boolean]) : Void
```

Associe une image bitmap à un clip.

Une fois le bitmap associé au clip, une référence est créée entre le clip et l'objet bitmap. Lorsque vous associez un bitmap, vous pouvez spécifier des paramètres `pixelSnapping` et `smoothing` pour affecter l'apparence du bitmap.

Les objets associés à un clip ne sont plus accessibles. Les paramètres `depth`, `pixelSnapping`,et `smoothing` doivent être définis lors de l'appel de la méthode `attachBitmap()` et ne peuvent plus être modifiés ultérieurement.

Utilisez d'abord `createEmptyMovieClip()` pour créer un clip vide. Utilisez ensuite la méthode `attachBitmap()`. De cette façon, vous pouvez appliquer des transformations au clip pour transformer le bitmap ; par exemple, par l'intermédiaire de la propriété `matrix` du clip.

L'accrochage aux pixels permet de placer le bitmap en fonction de la valeur de pixel intégral la plus proche et non pas en fonction d'une valeur partielle de pixel. Vous disposez de trois modes d'accrochage aux pixels :

- Le mode « Auto » (Automatique) procède à l'accrochage automatique tant que le bitmap n'est pas étiré ou n'a pas subi de rotation.
- Le mode « Always » (Toujours) procède systématiquement à l'accrochage aux pixels, quels que soient les facteurs d'étirement ou de rotation.
- Le mode « Never » (Jamais) désactive l'accrochage aux pixels pour le clip.

Le mode lissage affecte l'aspect de l'image lorsqu'elle est redimensionnée.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

### Paramètres

*bmp*:flash.display.BitmapData - Image bitmap transparente ou opaque.

*depth*:Number - Entier spécifiant le niveau de profondeur du clip devant recevoir l'image bitmap.

*pixelSnapping*:String [facultatif] - Les modes d'accrochage aux pixels sont `auto`, `always` et `never`. Le mode par défaut est `auto`.

*smoothing*:Boolean [facultatif] - Sélectionnez `true` pour activer le mode de lissage et `false` pour le désactiver. Le mode par défaut est désactivé.

### Exemple

L'exemple suivant associe un bitmap très simple à un clip :

```
import flash.display.*;

this.createEmptyMovieClip("bmp1", this.getNextHighestDepth());
var bmpData1:BitmapData = new BitmapData(200, 200, false, 0xaa3344);
bmp1.attachBitmap(bmpData1, 2, "auto", true);
```

Si votre fichier SWF comprend un composant de la version 2, utilisez les composants de la version 2 de la classe DepthManager au lieu de la méthode `MovieClip.getNextHighestDepth()`, utilisée dans cet exemple.

## attachMovie (méthode MovieClip.attachMovie)

```
public attachMovie(id:String, name:String, depth:Number,
  [initObject:Object]) : MovieClip
```

Sélectionne un symbole dans la bibliothèque et l'associe au clip. Utilisez `MovieClip.removeMovieClip()` ou `MovieClip.unloadMovie()` pour supprimer un fichier SWF lié à la méthode `attachMovie()`.

Vous pouvez étendre les méthodes et les gestionnaires d'événements de la classe MovieClip en créant une sous-classe.

**Disponibilité :** ActionScript 1.0 ; Flash Player 5

## Paramètres

*id*:String - Nom de liaison du symbole de clip de la bibliothèque à associer à un clip sur la scène. Il s'agit du nom entré dans le champ Identifiant de la boîte de dialogue Propriétés de liaison.

*name*:String - Nom d'occurrence unique du clip en cours de liaison au clip.

*depth*:Number - Entier spécifiant le niveau de profondeur du fichier SWF.

*initObject*:Object [facultatif] - (Pris en charge à partir de Flash Player 6) Objet contenant des propriétés permettant de remplir le clip qui vient d'être lié. Ce paramètre permet aux clips créés de façon dynamique de recevoir des paramètres. Si initObject n'est pas un objet, il est ignoré. Toutes les propriétés de initObject sont copiées dans la nouvelle occurrence. Les propriétés spécifiées avec initObject sont disponibles pour la fonction constructeur.

## Valeur renvoyée

MovieClip - Référence à la nouvelle occurrence.

## Exemple

L'exemple suivant associe le symbole portant l'identifiant de liaison circle à l'occurrence de clip, qui figure sur la scène dans le fichier SWF :

```
this.attachMovie("circle", "circle1_mc", this.getNextHighestDepth());
this.attachMovie("circle", "circle2_mc", this.getNextHighestDepth(),
  {_x:100, _y:100});
```

La méthode MovieClip.getNextHighestDepth() utilisée dans cet exemple nécessite Flash Player 7 ou une version ultérieure. Si votre fichier SWF comprend un composant de la version 2, utilisez les composants de la version 2 de la classe DepthManager au lieu de la méthode MovieClip.getNextHighestDepth().

## Voir également

removeMovieClip (méthode MovieClip.removeMovieClip), unloadMovie (méthode MovieClip.unloadMovie), Fonction removeMovieClip

# beginBitmapFill (méthode MovieClip.beginBitmapFill)

```
public beginBitmapFill(bmp:BitmapData, [matrix:Matrix], [repeat:Boolean],
  [smoothing:Boolean]) : Void
```

Remplit une zone de dessin avec une image bitmap. Le bitmap peut être répété ou former une mosaïque afin de remplir la zone.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

## Paramètres

*bmp*:flash.display.BitmapData - Image bitmap transparente ou opaque.

*matrix*:flash.geom.Matrix [facultatif] - Objet matrix (appartenant à la classe flash.geom.Matrix), qui permet de définir les transformations du bitmap. Par exemple, vous pouvez utiliser la matrice suivante pour faire pivoter un bitmap de 45 degrés (radians pi/4) :

```
var matrix = new flash.geom.Matrix();
matrix.rotate(Math.PI/4);
```

*repeat*:Boolean [facultatif] - Si la valeur est `true`, l'image bitmap se reproduit pour former un motif. Si la valeur est `false`, l'image bitmap ne se répète pas et les bords du bitmap sont utilisés pour tout remplissage qui dépasse le bitmap.

Considérons par exemple le bitmap suivant (un motif en damier de 20 x 20 pixels) :

Lorsque `repeat` est défini comme `true` (comme dans l'exemple suivant), le remplissage bitmap répète le bitmap :

Lorsque `repeat` est défini comme `false`, le remplissage bitmap utilise les pixels du bord pour le remplissage en dehors du bitmap :

*smoothing*:Boolean [facultatif] - Si la valeur est `false`, les images bitmap agrandies sont rendues en appliquant un algorithme d'approximation et ont un aspect pixélisé. Si la valeur est `true`, les images bitmap agrandies sont rendues avec un algorithme bilinéaire. Les rendus obtenus par l'intermédiaire de l'algorithme d'approximation sont généralement plus rapides. La valeur par défaut de ce paramètre est `false`.

### Exemple

Le code suivant définit un bitmap unique, puis utilise `beginBitmapFill()` pour remplir un clip en formant une mosaïque.

```
import flash.display.*;
import flash.geom.*;

var bmpd:BitmapData = new BitmapData(20,20);
var rect1:Rectangle = new Rectangle(0,0,10,10);
var rect2:Rectangle = new Rectangle(0, 10, 10, 20);
var rect3:Rectangle = new Rectangle(10, 0, 20, 10);
var rect4:Rectangle = new Rectangle(10, 10, 20, 20);
bmpd.fillRect(rect1, 0xAA0000FF);
bmpd.fillRect(rect2, 0xAA00FF00);
bmpd.fillRect(rect3, 0xAAFF0000);
bmpd.fillRect(rect4, 0xAA999999);

this.createEmptyMovieClip("bmp_fill_mc", this.getNextHighestDepth());
with (bmp_fill_mc) {
  matrix = new Matrix();
  matrix.rotate(Math.PI/8);
  repeat = true;
  smoothing = true;
  beginBitmapFill(bmpd, matrix, repeat, smoothing);
  moveTo(0, 0);
  lineTo(0, 60);
  lineTo(60, 60);
  lineTo(60, 0);
  lineTo(0, 0);
  endFill();
}

bmp_fill_mc._xscale = 200;
bmp_fill_mc._yscale = 200;
```

Si votre fichier SWF comprend un composant de la version 2, utilisez les composants de la version 2 de la classe DepthManager au lieu de la méthode `MovieClip.getNextHighestDepth()`, utilisée dans cet exemple.

## beginFill (méthode MovieClip.beginFill)

```
public beginFill(rgb:Number, [alpha:Number]) : Void
```

Indique le début d'un nouveau chemin de dessin. Si un tracé ouvert existe (autrement dit, si la position de dessin actuelle n'est pas égale à la position précédente spécifiée dans une méthode `MovieClip.moveTo()`) et qu'un remplissage y est associé, ce tracé est fermé à l'aide d'une ligne, puis rempli. Cette méthode produit les mêmes effets que lorsqu'une méthode `MovieClip.endFill()` est appelée.

Vous pouvez étendre les méthodes et les gestionnaires d'événements de la classe MovieClip en créant une sous-classe.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

## Paramètres

*rgb*:Number - Valeur colorimétrique hexadécimale ; par exemple, rouge correspond à 0xFF0000 et bleu à 0x0000FF. Si cette valeur n'est pas fournie ou n'est pas définie, le clip n'est pas rempli.

*alpha*:Number [facultatif] - Entier compris entre 0 et 100 qui spécifie la valeur alpha du remplissage. En l'absence de cette valeur, 100 (uni) s'applique. Si cette valeur est inférieure à 0, Flash utilise 0. Si elle est supérieure à 100, Flash applique 100.

## Exemple

L'exemple suivant crée un carré rouge sur la scène :

```
this.createEmptyMovieClip("square_mc", this.getNextHighestDepth());
square_mc.beginFill(0xFF0000);
square_mc.moveTo(10, 10);
square_mc.lineTo(100, 10);
square_mc.lineTo(100, 100);
square_mc.lineTo(10, 100);
square_mc.lineTo(10, 10);
square_mc.endFill();
```

La méthode `MovieClip.getNextHighestDepth()` utilisée dans cet exemple nécessite Flash Player 7 ou une version ultérieure. Si votre fichier SWF comprend un composant de la version 2, utilisez les composants de la version 2 de la classe DepthManager au lieu de la méthode `MovieClip.getNextHighestDepth()`.

Vous trouverez également un exemple dans le fichier drawingapi.fla du dossier Samples\ActionScript\DrawingAPI. La liste suivante présente les chemins type vers ce dossier :

- Windows : \Program Files\Macromedia\Flash 8\Samples and Tutorials\Samples\
- Macintosh : HD/Applications/Macromedia Flash 8/Samples and Tutorials/Samples/

## Voir également

`moveTo (méthode MovieClip.moveTo)`, `endFill (méthode MovieClip.endFill)`, `beginGradientFill (méthode MovieClip.beginGradientFill)`

# beginGradientFill (méthode MovieClip.beginGradientFill)

```
public beginGradientFill public (fillType:String, colors:Array,
  alphas:Array, ratios:Array, matrix:Object, [spreadMethod:String],
  [interpolationMethod:String], [focalPointRatio:Number]) : Void
```

Indique le début d'un nouveau chemin de dessin. Si le premier paramètre est `undefined`, ou si aucun paramètre n'est transmis, le chemin ne comporte pas de remplissage. Si un tracé ouvert existe (autrement dit si la position de dessin actuelle n'est pas égale à la position précédente spécifiée dans une méthode `MovieClip.moveTo()`), et si un remplissage y est associé, ce tracé est fermé à l'aide d'une ligne, puis rempli. Cette méthode est similaire à `MovieClip.endFill()`.

Cette méthode échoue si l'une des conditions suivantes est présente :

- Les nombres d'éléments dans les paramètres `colors`, `alphas`, et `ratios` ne sont pas égaux.
- Le paramètre `fillType` n'est pas `"linear"` ou `"radial"`.
- L'un des champs de l'objet correspondant au paramètre `matrix` est manquant ou non valide.

Vous pouvez étendre les méthodes et les gestionnaires d'événements de la classe MovieClip en créant une sous-classe.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

## Paramètres

*`fillType`*`:String` - Les valeurs possibles sont la chaîne « `linear` » et la chaîne « `radial` ».

*`colors`*`:Array` - Tableau de valeurs hexadécimales de couleurs RVB à utiliser pour le dégradé ; par exemple, rouge correspond à 0xFF0000, bleu à 0x0000FF. Vous pouvez définir jusqu'à 15 couleurs. Pour chaque couleur, veillez à définir une valeur correspondante dans les paramètres `alphas` et `ratios`.

*`alphas`*`:Array` - Tableau de valeurs alpha pour les couleurs correspondantes dans le tableau `colors` ; les valeurs valides vont de 0 à 100. Si la valeur est inférieure à 0, Flash utilise 0. Si la valeur est supérieure à 100, Flash utilise 100.

*`ratios`*`:Array` - Tableau de rapports de distribution des couleurs ; les valeurs valides vont de 0 à 255. Cette valeur définit le pourcentage de la largeur où la couleur est échantillonnée à 100 %. Spécifiez une valeur pour chaque valeur dans le paramètre `colors`.

Par exemple, pour un dégradé linéaire qui comprend deux couleurs, bleu et vert, la figure suivante illustre l'emplacement des couleurs dans le dégradé selon les différentes valeurs du tableau `ratios` :

| `ratios` | **Dégradé** |
|---|---|
| `[0, 127]` |  |
| `[0, 255]` |  |
| `[127, 255]` |  |

Les valeurs du tableau doivent augmenter de manière séquentielle ; par exemple, `[0, 63, 127, 190, 255]`.

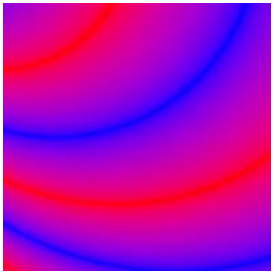
*`matrix`*`:Object` - Matrice de transformation qui peut prendre l'une des trois formes suivantes :

- Un objet matrix (pris en charge uniquement à partir de Flash Player 8), tel que défini par la classe flash.geom.Matrix. La classe flash.geom.Matrix inclut une méthode `createGradientBox()`, qui permet de configurer facilement la matrice en vue de son utilisation avec la méthode `beginGradientFill()` de la classe MovieClip. Macromedia recommande d'utiliser cette forme de l'objet matrix avec Flash Player 8 ou une version plus récente.
- L'exemple suivant utilise la méthode `beginGradientFill()` avec un paramètre `matrix` du type suivant :

```
import flash.geom.*

this.createEmptyMovieClip("gradient_mc", this.getNextHighestDepth());
with (gradient_mc)
{
  colors = [0xFF0000, 0x0000FF];
  fillType = "radial"
  alphas = [100, 100];
  ratios = [0, 0xFF];
  spreadMethod = "reflect";
  interpolationMethod = "linearRGB";
  focalPointRatio = 0.9;
  matrix = new Matrix();
  matrix.createGradientBox(100, 100, Math.PI, 0, 0);
  beginGradientFill(fillType, colors, alphas, ratios, matrix,
    spreadMethod, interpolationMethod, focalPointRatio);
  moveTo(100, 100);
  lineTo(100, 300);
  lineTo(300, 300);
  lineTo(300, 100);
```

```
  lineTo(100, 100);
  endFill();
}
```

- Ce code dessine l'image suivante à l'écran :



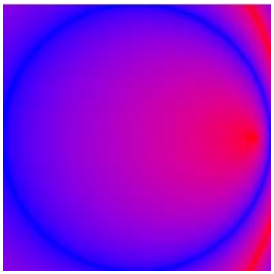
- Vous pouvez utiliser les propriétés `a`, `b`, `c`, `d`, `e`, `f`, `g`, `h`, et `i`, qui peuvent servir à décrire une matrice 3 x 3 de la forme suivante :

```
a b c
d e f
g h i
```

- *Remarque :* A partir de Flash Player 8, Macromedia recommande de définir le paramètre matrix sous la forme d'un objet `flash.geom.Matrix` (comme indiqué dans le premier élément de cette liste).
- L'exemple suivant utilise la méthode `beginGradientFill()` avec un paramètre `matrix` du type suivant :

```
this.createEmptyMovieClip("gradient_mc", this.getNextHighestDepth());
with (gradient_mc)
{
  colors = [0xFF0000, 0x0000FF];
  fillType = "radial"
  alphas = [100, 100];
  ratios = [0, 0xFF];
  spreadMethod = "reflect";
  interpolationMethod = "linearRGB";
  focalPointRatio = 0.9;
  matrix = {a:200, b:0, c:0, d:0, e:200, f:0, g:200, h:200, i:1};
  beginGradientFill(fillType, colors, alphas, ratios, matrix, spreadMethod,
interpolationMethod, focalPointRatio);
  moveTo(100, 100);
  lineTo(100, 300);
  lineTo(300, 300);
  lineTo(300, 100);
  lineTo(100, 100);
  endFill();
}
```

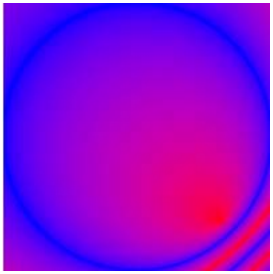
- Ce code dessine l'image suivante à l'écran :



- Un objet avec les propriétés suivantes : `matrixType`, `x`, `y`, `w`, `h`, `r`.
- Les propriétés ont la signification suivante : `matrixType` correspond à la chaîne `"box"`, `x` désigne la position horizontale par rapport au point d'alignement du clip parent pour le coin supérieur gauche du dégradé, `y` indique la position verticale par rapport au point d'alignement du clip parent pour le coin supérieur gauche du dégradé, `w` correspond à la largeur du dégradé, `h` à sa hauteur, et `r` indique la rotation en radians du dégradé.
- *Remarque :* A partir de Flash Player 8, Macromedia recommande de définir le paramètre `matrix` sous la forme d'un objet `flash.geom.Matrix` (comme indiqué dans le premier élément de cette liste).
- L'exemple suivant utilise la méthode `beginGradientFill()` avec un paramètre `matrix` du type suivant :

```
this.createEmptyMovieClip("gradient_mc", this.getNextHighestDepth());
with (gradient_mc)
{
  colors = [0xFF0000, 0x0000FF];
  fillType = "radial"
  alphas = [100, 100];
  ratios = [0, 0xFF];
  spreadMethod = "reflect";
  interpolationMethod = "linearRGB";
  focalPointRatio = 0.9;
  matrix = {matrixType:"box", x:100, y:100, w:200, h:200, r:(45/
  180)*Math.PI};
  beginGradientFill(fillType, colors, alphas, ratios, matrix,
    spreadMethod, interpolationMethod, focalPointRatio);
  moveTo(100, 100);
  lineTo(100, 300);
  lineTo(300, 300);
  lineTo(300, 100);
  lineTo(100, 100);
  endFill();
}
```

■ Ce code dessine l'image suivante à l'écran :

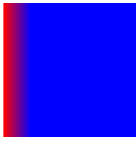


*spreadMethod*:String [facultatif] - Ajouté à Flash Player 8. Soit « pad », « reflect » ou « repeat », ce qui contrôle le mode du remplissage en dégradé. La valeur par défaut est « pad ».

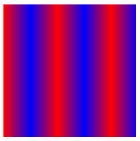
Par exemple, considérons un dégradé linéaire simple entre deux couleurs :

```
import flash.geom.*;
var fillType:String = "linear"
var colors:Array = [0xFF0000, 0x0000FF];
var alphas:Array = [100, 100];
var ratios:Array = [0x00, 0xFF];
var matrix:Matrix = new Matrix();
matrix.createGradientBox(20, 20, 0, 0, 0);
var spreadMethod:String = "pad";
this.beginGradientFill(fillType, colors, alphas, ratios, matrix,
  spreadMethod);
this.moveTo(0, 0);
this.lineTo(0, 100);
this.lineTo(100, 100);
this.lineTo(100, 0);
this.lineTo(0, 0);
this.endFill();
```

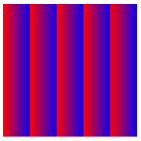
Cet exemple utilise `"pad"` comme méthode d'étalement, et le remplissage en dégradé prend donc l'aspect suivant :



Si vous avez utilisé `"reflect"` comme méthode d'étalement, le remplissage en dégradé prend l'aspect suivant :

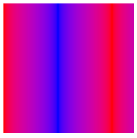
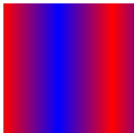


Si vous avez utilisé `"repeat"` comme méthode d'étalement, le remplissage en dégradé prend l'aspect suivant :

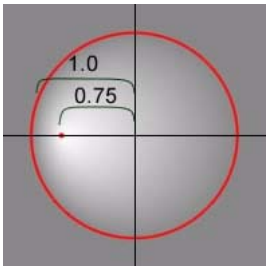


*interpolationMethod*:`String` [facultatif] - Ajouté dans Flash Player 8. Soit « RGB », soit « linearRGB ». Avec « linearRGB », les couleurs sont distribuées de façon linéaire dans le dégradé. La valeur par défaut est « RGB ».

Prenons, par exemple, un dégradé linéaire simple entre deux couleurs (avec le paramètre `spreadMethod` défini sur « reflect »). Les différentes méthodes d'interpolation influent sur l'aspect de la façon suivante :

| | |
|---|---|
|  |  |
| `"linearRGB"` | `"RGB"` |

*focalPointRatio*:Number [facultatif] - Ajouté dans Flash Player 8. Nombre qui contrôle l'emplacement du point focal du dégradé. La valeur 0 signifie que le point focal est au centre. La valeur 1 signifie que le point focal est au bord du cercle du dégradé. La valeur -1 signifie que le point focal est sur l'autre bord du cercle du dégradé. Toute valeur inférieure à -1 ou supérieure à 1 est arrondie à -1 ou 1. L'exemple suivant présente une valeur `focalPointRatio` définie sur 0,75 :




## Exemple

Le code suivant crée un effet d'ombre sphérique :

```
import flash.geom.*
this.createEmptyMovieClip("gradient_mc", this.getNextHighestDepth());
with (gradient_mc)
  {
    fillType = "radial"
  colors = [0x000000, 0xFFFFFF];
    alphas = [50, 90];
    ratios = [0, 0xFF];
    spreadMethod = "pad";
    interpolationMethod = "RGB";
  focalPointRatio = 0.3;
  matrix = new Matrix();
  matrix.createGradientBox(100, 100, 0, 0, 0);
  beginGradientFill(fillType, colors, alphas, ratios, matrix,
    spreadMethod, interpolationMethod, focalPointRatio);
  moveTo(0, 0);
  lineTo(0, 100);
  lineTo(100, 100);
  lineTo(100, 0);
  lineTo(0, 0);
  endFill();
}
```

Ceci dessine l'illustration ci-dessous (mise à l'échelle à 50 %) :

Si votre fichier SWF comprend un composant de la version 2, utilisez les composants de la version 2 de la classe DepthManager au lieu de la méthode `MovieClip.getNextHighestDepth()`, utilisée dans cet exemple.

### Voir également

`createGradientBox (méthode Matrix.createGradientBox)`, `beginFill (méthode MovieClip.beginFill)`, `endFill (méthode MovieClip.endFill)`, `lineStyle (méthode MovieClip.lineStyle)`, `lineTo (méthode MovieClip.lineTo)`, `moveTo (méthode MovieClip.moveTo)`

## blendMode (propriété MovieClip.blendMode)

```
public blendMode : Object
```

Le mode de fondu de ce clip. Le mode de fondu affecte l'apparence du clip lorsqu'il se trouve dans un calque au-dessus d'un autre objet à l'écran.

Flash Player applique la propriété `blendMode` sur chaque pixel du clip. Chaque pixel est composé de trois couleurs constituantes (rouge, vert et bleu) et chaque couleur a une valeur comprise entre 0x00 et 0xFF. Flash Player compare chaque couleur constituante d'un pixel dans le clip avec la couleur correspondante du pixel d'arrière-plan. Par exemple si `blendMode` est défini sur `"lighten"`, Flash Player compare la valeur rouge du clip avec la valeur rouge de l'arrière-plan, et utilise la plus claire des deux comme valeur pour le composant rouge de la couleur affichée.

Le tableau suivant répertorie les paramètres `blendMode`. Pour définir la propriété `blendMode` vous pouvez utiliser un entier compris entre 1 et 14 ou une chaîne. Les illustrations du tableau ci-dessous présentent les valeurs `blendMode` appliquées à un clip circulaire (2) superposé sur un autre objet (1) à l'écran.

| Valeur de l'entier | Valeur de chaîne | Illustration | Description |
|---|---|---|---|
| 1 | `"normal"` |  | Le clip apparaît devant l'arrière-plan. Les valeurs de pixel du clip remplacent celles de l'arrière-plan. Lorsque le clip est transparent, l'arrière-plan est visible. |
| 2 | `"layer"` |  | Force la création d'un tampon temporaire pour la précomposition du clip. Ceci se produit automatiquement si un clip comporte plusieurs objets enfants et si un paramètre `blendMode` différent de `"normal"` est sélectionné pour l'enfant. |

| Valeur de l'entier | Valeur de chaîne | Illustration | Description |
|---|---|---|---|
| 3 | `"multiply"` |  | Multiplie les valeurs pour les couleurs constituantes du clip par celles de la couleur d'arrière-plan, puis les normalise en les divisant par 0xFF, ce qui donne des couleurs plus sombres. Ce paramètre est souvent utilisé pour les effets d'ombre et de profondeur.<br>Par exemple, si une couleur constituante (comme le rouge) d'un pixel dans un clip et la couleur correspondante du pixel de l'arrière-plan ont toutes les deux une valeur de 0x88, le résultat de la multiplication sera 0x4840. La division par 0xFF donne une valeur de 0x48 pour cette couleur constituante, qui est plus sombre que celle du clip ou de l'arrière-plan. |

| Valeur de l'entier | Valeur de chaîne | Illustration | Description |
|---|---|---|---|
| 4 | `"screen"` |  | Multiplie le complément (inverse) de la couleur du clip par le complément de la couleur de l'arrière-plan, ce qui crée un effet de blanchiment. Ce paramètre est couramment utilisé pour la mise en valeur ou pour supprimer les parties noires du clip. |
| 5 | `"lighten"` |  | Sélectionne les plus claires des couleurs constituantes du clip et de l'arrière-plan (celles qui ont les valeurs les plus grandes). Ce paramètre est couramment utilisé pour le type de superposition.<br>Par exemple, si le clip a un pixel dont la valeur RVB est de 0xFFCC33, et que le pixel d'arrière-plan a une valeur RVB de 0xDDF800, alors la valeur RVB obtenue pour le pixel affiché est de 0xFFF833 (car 0xFF > 0xDD, 0xCC < 0xF8, et 0x33 > 0x00 = 33). |

| Valeur de l'entier | Valeur de chaîne | Illustration | Description |
|---|---|---|---|
| 6 | "darken" |  | Sélectionne les plus sombres des couleurs constituantes du clip et de l'arrière-plan (celles qui ont les valeurs les plus petites). Ce paramètre est couramment utilisé pour le type de superposition.<br>Par exemple, si le clip a un pixel dont la valeur RVB est de 0xFFCC33, et que le pixel d'arrière-plan a une valeur RVB de 0xDDF800, alors la valeur RVB obtenue pour le pixel affiché est de 0xDDCC00 (car 0xFF > 0xDD, 0xCC < 0xF8, et 0x33 > 0x00 = 33). |

| Valeur de l'entier | Valeur de chaîne | Illustration | Description |
|---|---|---|---|
| 7 | `"difference"` |  | Compare les couleurs constituantes d'un clip avec celles de son arrière-plan et soustrait la plus sombre des deux couleurs constituantes de la plus claire. Ce paramètre est couramment utilisé pour obtenir des couleurs plus vives. Par exemple, si le clip a un pixel dont la valeur RVB est de 0xFFCC33, et que le pixel d'arrière-plan a une valeur RVB de 0xDDF800, alors la valeur RVB obtenue pour le pixel affiché est de 0x222C33 (car 0xFF - 0xDD = 0x22, 0xF8 - 0xCC = 0x2C, et 0x33 - 0x00 = 0x33). |

| Valeur de l'entier | Valeur de chaîne | Illustration | Description |
|---|---|---|---|
| 8 | "add" |  | Ajoute les valeurs des couleurs constituantes du clip à celles de l'arrière-plan, en appliquant un plafond de 0xFF. Ce paramètre est couramment utilisé pour animer une dissolution éclaircissante entre deux objets.<br>Par exemple, si le clip a un pixel dont la valeur RVB est de 0xAAA633, et que le pixel d'arrière-plan a une valeur RVB de 0xDD2200, alors la valeur RVB obtenue pour le pixel affiché est de 0xFFC833 (car 0xAA + 0xDD > 0xFF, 0xA6 + 0x22 = 0xC8, et 0x33 + 0x00 = 0x33). |

| Valeur de l'entier | Valeur de chaîne | Illustration | Description |
|---|---|---|---|
| 9 | "subtract" |  | Soustrait les valeurs des couleurs constituantes du clip de celles de l'arrière-plan, en appliquant un plancher de 0. Ce paramètre est couramment utilisé pour animer la dissolution de deux objets en assombrissant progressivement leurs couleurs.<br>Par exemple, si le clip a un pixel dont la valeur RVB est de 0xAA2233, et que le pixel d'arrière-plan a une valeur RVB de 0xDDA600, alors la valeur RVB obtenue pour le pixel affiché est de 0x338400 (car 0xDD - 0xAA = 0x33, 0xA6 - 0x22 = 0x84, et 0x00 - 0x33 › 0x00). |
| 10 | "invert" |  | Inverse l'arrière-plan. |

| Valeur de l'entier | Valeur de chaîne | Illustration | Description |
|---|---|---|---|
| 11 | `"alpha"` |  | Applique la valeur alpha de chaque pixel du clip à l'arrière-plan. Ceci nécessite que le paramètre `blendMode` " layer " soit appliqué à un clip parent. Par exemple, dans l'illustration, le clip parent, qui est un arrière-plan blanc, a un paramètre `blendMode = "layer"`. |
| 12 | `"erase"` |  | Efface l'arrière-plan à partir de la valeur alpha du clip. Ceci nécessite que le paramètre `blendMode` " layer " soit appliqué à un clip parent. Par exemple, dans l'illustration, le clip parent, qui est un arrière-plan blanc, a un paramètre `blendMode = "layer"`. |

| Valeur de l'entier | Valeur de chaîne | Illustration | Description |
|---|---|---|---|
| 13 | `"overlay"` |  | Ajuste la couleur de chaque bitmap à partir de la teinte sombre de l'arrière-plan. Si l'arrière-plan est plus clair qu'un gris à 50 %, les couleurs du clip et de l'arrière-plan sont masquées, ce qui permet d'obtenir une couleur plus claire. Si l'arrière-plan est plus sombre qu'un gris à 50 %, les couleurs sont multipliées, ce qui permet d'obtenir une couleur plus sombre. Ce paramètre est couramment utilisé pour des effets d'ombre. |
| 14 | `"hardlight"` |  | Ajuste la couleur de chaque bitmap à partir de la teinte sombre du clip. Si le clip est plus clair qu'un gris à 50 %, les couleurs du clip et de l'arrière-plan sont masquées, ce qui permet d'obtenir une couleur plus claire. Si le clip est plus sombre qu'un gris à 50 %, les couleurs sont multipliées, ce qui permet d'obtenir une couleur plus sombre. Ce paramètre est couramment utilisé pour des effets d'ombre. |

Si vous tentez de définir la propriété `blendMode` sur une autre valeur, Flash Player la définit sur `"normal"`.

Toutefois, si vous définissez cette propriété sur un entier, Flash Player convertit cette valeur en une chaîne correspondante :

```
this.createEmptyMovieClip("mclip", this.getNextHighestDepth());
mclip.blendMode = 8;
trace (mclip.blendMode) // add
```

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

## Exemple

L'exemple suivant crée deux clips avec des remplissages en dégradé et change le mode de fondu de celui au premier plan toutes les secondes. Pour que le mode de fondu `"alpha"` s'affiche avec un effet, le dégradé pour le clip `mc2` comprend une plage de rapports alpha, et le mode de fondu `"layer"` est appliqué au clip parent (`this.blendMode="layer"`).

```
  this.createEmptyMovieClip("mc1", this.getNextHighestDepth());
  this.createEmptyMovieClip("mc2", this.getNextHighestDepth());
  this.blendMode="layer";
  this.createTextField("blendLabel", this.getNextHighestDepth(), 50, 150,
  100, 100)

  fillClip(mc1, 0x00AA00, 0x22FFFF, 100, 100)
  fillClip(mc2, 0xFF0000, 0x2211FF, 100, 50)
  mc2._x = 33;
  mc2._y = 33;

  var blendModeIndex = 0;

  setInterval(changeBlendMode, 1000);
  function changeBlendMode()
  {
    mc2.blendMode = blendModeIndex % 14 + 1 ;
      // values 1 - 14
  blendLabel.text = (blendModeIndex% 14 + 1) + ": " + mc2.blendMode;
    blendModeIndex++;
  }

  function fillClip(mc:MovieClip, color1:Number, color2:Number,
        alpha1:Number, alpha2: Number)
{
    matrix = {a:100, b:0, c:0, d:0, e:100, f:0, g:50, h:20, i:1};
    mc.beginGradientFill("linear", [color1, color2], [alpha1, alpha2], [0,
  0xFF], matrix);
    mc.lineStyle(8,0x888888,100)
    mc.moveTo(0, 0);
    mc.lineTo(0, 100);
    mc.lineTo(100, 100);
```

```
    mc.lineTo(100, 0);
    mc.lineTo(0, 0);
    mc.endFill();
  }
```

Si votre fichier SWF comprend un composant de la version 2, utilisez les composants de la version 2 de la classe DepthManager au lieu de la méthode `MovieClip.getNextHighestDepth()`, utilisée dans l'exemple précédent.

## cacheAsBitmap (propriété MovieClip.cacheAsBitmap)

```
public cacheAsBitmap : Boolean
```

Si défini sur `true`, Flash Player place en mémoire cache une version bitmap interne du clip. Cette propriété peut permettre d'améliorer les performances des clips incluant un contenu vectoriel complexe.

Toutes les données vectorielles d'un clip contenant un bitmap en mémoire cache sont tracées sur le bitmap et non pas sur la scène principale. Ce bitmap est ensuite copié sur la scène principale sous forme de pixels sans étirement ou rotation et accroché aux limites de pixels les plus proches. Les correspondances des pixels avec l'objet parent se font selon un rapport de un à un. Si les limites du bitmap changent, le bitmap est recréé au lieu d'être étiré.

Aucun bitmap interne n'est créé sauf si la propriété `cacheAsBitmap` est définie sur `true`.

Après avoir défini la propriété `cacheAsBitmap` du clip sur `true`, le rendu ne change pas, bien que le clip procède automatiquement à l'accrochage aux pixels. La vitesse d'animation peut être beaucoup plus importante, selon la complexité du contenu vectoriel.

La propriété `cacheAsBitmap` est automatiquement définie sur `true` chaque fois que vous appliquez un filtre à un clip (lorsque son tableau `filter` n'est pas vide). Si un filtre est appliqué à un clip, la propriété `cacheAsBitmap` est signalée comme `true` pour ce clip, même si vous avez défini cette propriété sur `false`. Si vous effacez tous les filtres d'un clip, le paramètre `cacheAsBitmap` reprend sa *définition* précédente.

Dans les cas suivants, les clips n'utilisent pas de bitmap, même si la propriété `cacheAsBitmap` est définie sur `true`, et procède plutôt au rendu du clip à partir de données vectorielles :

- Le bitmap est trop grand : plus de 2 880 pixels dans les deux directions.
- Le bitmap ne peut pas être alloué à une mémoire (erreur de type saturation de la mémoire).

La propriété `cacheAsBitmap` est utilisée de préférence avec les clips dont le contenu est principalement statique et qui n'est ni redimensionné, ni pivoté de façon fréquente. Avec ce type de clip, `cacheAsBitmap` permet d'améliorer les performances lors de la conversion (lorsque les positions *x* et *y* sont changées).

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

## Exemple

L'exemple suivant applique un filtre d'ombre portée à une occurrence de clip. Il présente ensuite la valeur de la propriété `cacheAsBitmap` définie sur `true` lorsqu'un filtre est appliqué.

```
import flash.filters.DropShadowFilter;

var container:MovieClip = setUpShape();
trace(container.cacheAsBitmap); // false
var dropShadow:DropShadowFilter = new DropShadowFilter(6, 45, 0x000000, 50,
  5, 5, 1, 2, false, false, false);
container.filters = new Array(dropShadow);
trace(container.cacheAsBitmap); // true

function setUpShape():MovieClip {
  var mc:MovieClip = this.createEmptyMovieClip("container",
  this.getNextHighestDepth());
  mc._x = 10;
  mc._y = 10;
  var w:Number = 50;
  var h:Number = 50;
  mc.beginFill(0xFFCC00);
  mc.lineTo(w, 0);
  mc.lineTo(w, h);
  mc.lineTo(0, h);
  mc.lineTo(0, 0);
  mc.endFill();
  return mc;
}
```

## Voir également

`opaqueBackground (propriété MovieClip.opaqueBackground)`, `cacheAsBitmap (propriété MovieClip.cacheAsBitmap)`

# clear (méthode MovieClip.clear)

```
public clear() : Void
```

Supprime tous les graphiques créés lors de l'exécution à l'aide des méthodes de dessin de clips, y compris les styles de trait spécifiés par `MovieClip.lineStyle()`. Les formes et les lignes tracées manuellement pendant la programmation (à l'aide des outils de dessin Flash) ne sont pas affectées.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

## Exemple

L'exemple suivant trace un cadre sur la scène. Lorsque l'utilisateur clique sur l'image du cadre, il supprime cette dernière de la scène.

```
this.createEmptyMovieClip("box_mc", this.getNextHighestDepth());
box_mc.onRelease = function() {
  this.clear();
};
drawBox(box_mc, 10, 10, 320, 240);
function drawBox(mc:MovieClip, x:Number, y:Number, w:Number, h:Number):Void
  {
  mc.lineStyle(0);
  mc.beginFill(0xEEEEEE);
  mc.moveTo(x, y);
  mc.lineTo(x+w, y);
  mc.lineTo(x+w, y+h);
  mc.lineTo(x, y+h);
  mc.lineTo(x, y);
  mc.endFill();
}
```

La méthode `MovieClip.getNextHighestDepth()` utilisée dans cet exemple nécessite Flash Player 7 ou une version ultérieure. Si votre fichier SWF comprend un composant de la version 2, utilisez les composants de la version 2 de la classe DepthManager au lieu de la méthode `MovieClip.getNextHighestDepth()`.

Vous trouverez également un exemple dans le fichier drawingapi.fla du dossier Samples\ActionScript\DrawingAPI. La liste suivante présente les chemins type vers ce dossier :

- Windows : \Program Files\Macromedia\Flash 8\Samples and Tutorials\Samples\ActionScript
- Macintosh : HD/Applications/Macromedia Flash 8/Samples and Tutorials/Samples/ActionScript

### Voir également

`lineStyle (méthode MovieClip.lineStyle)`

## createEmptyMovieClip (méthode MovieClip.createEmptyMovieClip)

```
public createEmptyMovieClip(name:String, depth:Number) : MovieClip
```

Crée un clip vide en tant qu'enfant d'un clip existant. Cette méthode agit de façon similaire à la méthode `attachMovie()`, mais il n'est pas nécessaire de fournir d'identifiant de liaison externe pour le nouveau clip. Le point d'alignement d'un clip vide nouvellement créé se situe dans le coin supérieur gauche. Cette méthode échoue si l'un des paramètres suivants est manquant.

Vous pouvez étendre les méthodes et les gestionnaires d'événements de la classe MovieClip en créant une sous-classe.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

### Paramètres

*`name`*`:String` - Chaîne qui identifie le nom d'occurrence du nouveau clip.

*`depth`*`:Number` - Entier qui spécifie la profondeur du nouveau clip.

### Valeur renvoyée

`MovieClip` - Référence au nouveau clip.

### Exemple

L'exemple suivant crée un clip vide appelé `container`, crée un nouveau TextField à l'intérieur, puis définit la nouvelle propriété `TextField.text`.

```
var container:MovieClip = this.createEmptyMovieClip("container",
  this.getNextHighestDepth());
var label:TextField = container.createTextField("label", 1, 0, 0, 150, 20);
label.text = "Hello World";
```

La méthode `MovieClip.getNextHighestDepth()` utilisée dans cet exemple nécessite Flash Player 7 ou une version ultérieure. Si votre fichier SWF comprend un composant de la version 2, utilisez les composants de la version 2 de la classe DepthManager au lieu de la méthode `MovieClip.getNextHighestDepth()`.

### Voir également

`attachMovie (méthode MovieClip.attachMovie)`

# createTextField (méthode MovieClip.createTextField)

```
public createTextField(instanceName:String, depth:Number, x:Number,
  y:Number, width:Number, height:Number) : TextField
```

Crée un nouveau champ de texte vide en tant qu'enfant du clip pour lequel vous avez appelé cette méthode. Vous pouvez utiliser la méthode `createTextField()` pour créer des champs de texte lors de la lecture d'un fichier SWF. Le paramètre `depth` détermine le niveau de profondeur (la position de l'ordre z) du nouveau champ de texte dans le clip. Chaque niveau de profondeur peut contenir uniquement un objet. Si vous créez un nouveau champ de texte sur une profondeur disposant déjà d'un champ de texte, le nouveau champ de texte remplace le champ de texte existant. Pour éviter d'écraser des champs de texte existants, utilisez la méthode `MovieClip.getInstanceAtDepth()` afin de déterminer si une profondeur spécifique est déjà occupée, ou la méthode `MovieClip.getNextHighestDepth()` afin de déterminer la profondeur inoccupée la plus élevée. Le champ de texte est positionné aux coordonnées (*x*, *y*) en adoptant les dimensions définies par les paramètres `width` x *`height`*. Les paramètres `x` et `y` sont calculés par rapport au conteneur du clip ; ces paramètres correspondent aux propriétés `_x` et `_y` du champ de texte. Les paramètres `width` et `height` correspondent aux propriétés `_width` et `_height` du champ de texte.

Les propriétés par défaut d'un champ de texte sont les suivantes :

```
type = "dynamic"
border = false
background = false
password = false
multiline = false
html = false
embedFonts = false
selectable = true
wordWrap = false
mouseWheelEnabled = true
condenseWhite = false
restrict = null
variable = null
maxChars = null
styleSheet = undefined
tabInded = undefined
```

Un champ de texte créé avec `createTextField()` reçoit les paramètres d'objet TextFormat par défaut suivants :

```
font = "Times New Roman" // "Times" on Mac OS
size = 12
color = 0x000000
bold = false
italic = false
```

```
underline = false
url = ""
target = ""
align = "left"
leftMargin = 0
rightMargin = 0
indent = 0
leading = 0
blockIndent = 0
bullet = false
display = block
tabStops = [] // (empty array)
```

Vous pouvez étendre les méthodes et les gestionnaires d'événements de la classe MovieClip en créant une sous-classe.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

## Paramètres

*instanceName*:String - Chaîne qui identifie le nom d'occurrence du nouveau champ de texte.

*depth*:Number - Entier positif qui spécifie la profondeur du nouveau champ de texte.

*x*:Number - Entier qui spécifie la coordonnée *x* du nouveau champ de texte.

*y*:Number - Entier qui spécifie la coordonnée *y* du nouveau champ de texte.

*width*:Number - Entier positif qui spécifie la largeur du nouveau champ de texte.

*height*:Number - Entier positif qui spécifie la hauteur du nouveau champ de texte.

## Valeur renvoyée

TextField - Flash Player 8 renvoie une référence à l'objet TextField créé. Les versions antérieures à Flash Player 8 renvoient void.

## Exemple

L'exemple suivant crée un champ de texte d'une largeur de 300, d'une hauteur de 100, une coordonnée *x* de 100, une coordonnée *y* de 100, pas de bordure, texte en rouge et souligné :

```
this.createTextField("my_txt", 1, 100, 100, 300, 100);
my_txt.multiline = true;
my_txt.wordWrap = true;
var my_fmt:TextFormat = new TextFormat();
my_fmt.color = 0xFF0000;
my_fmt.underline = true;
my_txt.text = "This is my first test field object text.";
my_txt.setTextFormat(my_fmt);
```

Vous trouverez également un exemple dans le fichier animation.fla du dossier Samples\ActionScriptAnimation. La liste suivante présente les chemins type vers ce dossier :

- Windows : \Program Files\Macromedia\Flash 8\Samples et Tutorials\Samples\
- Macintosh : HD/Applications/Macromedia Flash 8/Samples et Tutorials/Samples/

### Voir également

`getInstanceAtDepth (méthode MovieClip.getInstanceAtDepth)`, `getNextHighestDepth (méthode MovieClip.getNextHighestDepth)`, `TextFormat`

## _currentframe (propriété MovieClip._currentframe)

```
public _currentframe : Number [lecture seule]
```

Renvoie le numéro de l'image dans laquelle se trouve la tête de lecture dans le scénario du clip.

**Disponibilité :** ActionScript 1.0 ; Flash Player 4

### Exemple

L'exemple suivant utilise la propriété `_currentframe` pour faire avancer de cinq images la tête de lecture du clip `actionClip_mc` par rapport à sa position actuelle :

```
actionClip_mc.gotoAndStop(actionClip_mc._currentframe + 5);
```

## curveTo (méthode MovieClip.curveTo)

```
public curveTo(controlX:Number, controlY:Number, anchorX:Number,
  anchorY:Number) : Void
```

Dessine une courbe en utilisant le style de ligne actuel à partir de la position actuelle à (`anchorX`, `anchorY`) en utilisant le point de contrôle spécifié par (`controlX`, `controlY`). La position de dessin actuelle est ensuite définie sur (`anchorX`, `anchorY`). Si le clip dans lequel vous effectuez le traçage possède un contenu créé à l'aide des outils de dessin Flash, les appels de la méthode `curveTo()` sont tracés sous ce contenu. Si vous appelez la méthode `curveTo()` avant tout appel à la méthode `moveTo()`, la position de dessin actuelle est définie sur la valeur par défaut (0,0). Si l'un des paramètres est manquant, cette méthode échoue et la position de dessin actuelle n'est pas modifiée.

Vous pouvez étendre les méthodes et les gestionnaires d'événements de la classe MovieClip en créant une sous-classe.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

## Paramètres

*controlX*:Number - Entier qui spécifie la position horizontale du point de contrôle par rapport au point d'alignement du clip parent.

*controlY*:Number - Entier qui spécifie la position verticale du point de contrôle par rapport au point d'alignement du clip parent.

*anchorX*:Number - Entier qui spécifie la position horizontale du point d'ancrage suivant par rapport au point d'alignement du clip parent.

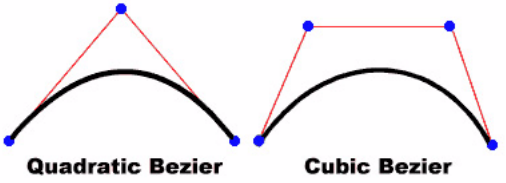
*anchorY*:Number - Entier qui spécifie la position verticale du point d'ancrage suivant par rapport au point d'alignement du clip parent.

## Exemple

L'exemple suivant dessine une courbe quasi-circulaire avec un trait bleu uni en filet et un remplissage rouge uni.

```
this.createEmptyMovieClip("circle_mc", 1);
with (circle_mc) {
  lineStyle(0, 0x0000FF, 100);
  beginFill(0xFF0000);
  moveTo(0, 100);
  curveTo(0,200,100,200);
  curveTo(200,200,200,100);
  curveTo(200,0,100,0);
  curveTo(0,0,0,100);
  endFill();
}
```

La courbe dessinée dans cet exemple est une courbe de Bézier quadratique. Les courbes de Bézier quadratiques comprennent deux points d'ancrage et un point de contrôle. La courbe interpole les deux points d'ancrage et s'incurve en direction du point de contrôle.




Le script suivant utilise la méthode `curveTo()` et la classe Math pour créer un cercle :

```
this.createEmptyMovieClip("circle2_mc", 2);
circle2_mc.lineStyle(0, 0x000000);
drawCircle(circle2_mc, 100, 100, 100);
function drawCircle(mc:MovieClip, x:Number, y:Number, r:Number):Void {
  mc.moveTo(x+r, y);
```

```
  mc.curveTo(r+x, Math.tan(Math.PI/8)*r+y, Math.sin(Math.PI/4)*r+x,
Math.sin(Math.PI/4)*r+y);
  mc.curveTo(Math.tan(Math.PI/8)*r+x, r+y, x, r+y);
  mc.curveTo(-Math.tan(Math.PI/8)*r+x, r+y, -Math.sin(Math.PI/4)*r+x,
Math.sin(Math.PI/4)*r+y);
  mc.curveTo(-r+x, Math.tan(Math.PI/8)*r+y, -r+x, y);
  mc.curveTo(-r+x, -Math.tan(Math.PI/8)*r+y, -Math.sin(Math.PI/4)*r+x,
-Math.sin(Math.PI/4)*r+y);
  mc.curveTo(-Math.tan(Math.PI/8)*r+x, -r+y, x, -r+y);
  mc.curveTo(Math.tan(Math.PI/8)*r+x, -r+y, Math.sin(Math.PI/4)*r+x,
-Math.sin(Math.PI/4)*r+y);
  mc.curveTo(r+x, -Math.tan(Math.PI/8)*r+y, r+x, y);
}
```

Vous trouverez également un exemple dans le fichier drawingapi.fla du dossier Samples\ActionScript\DrawingAPI. La liste suivante présente les chemins type vers ce dossier :

- Windows : \Program Files\Macromedia\Flash 8\Samples et Tutorials\Samples\
- Macintosh : HD/Applications/Macromedia Flash 8/Samples et Tutorials/Samples/

### Voir également

`beginFill (méthode MovieClip.beginFill)`, `createEmptyMovieClip (méthode MovieClip.createEmptyMovieClip)`, `endFill (méthode MovieClip.endFill)`, `lineStyle (méthode MovieClip.lineStyle)`, `lineTo (méthode MovieClip.lineTo)`, `moveTo (méthode MovieClip.moveTo)`, `Math`

## _droptarget (propriété MovieClip._droptarget)

```
public _droptarget : String [lecture seule]
```

Renvoi le chemin absolu, en utilisant une notation de syntaxe à barre oblique, de l'occurrence de clip sur laquelle ce clip a été déposé. La propriété `_droptarget` renvoie toujours un chemin qui commence par une barre oblique (`/`). Pour comparer la propriété `_droptarget` d'une occurrence à une référence, utilisez la fonction `eval()` afin de convertir la valeur renvoyée d'une syntaxe à barre oblique en référence de syntaxe à point.

*Remarque :* Vous devez effectuer cette conversion si vous utilisez ActionScript 2.0 car celui-ci ne prend pas en charge la syntaxe à barre oblique.

**Disponibilité :** ActionScript 1.0 ; Flash Player 4

### Exemple

L'exemple suivant évalue la propriété `_droptarget` de l'occurrence de clip `garbage_mc` et utilise `eval()` pour convertir la syntaxe à barre oblique en syntaxe à point. La référence `garbage_mc` est alors comparée à la référence de l'occurrence de clip `trashcan_mc`. Si les deux références sont équivalentes, la visibilité de `garbage_mc` est définie sur `false`. Si elles divergent, l'occurrence `garbage` reprend sa position d'origine.

```
origX = garbage_mc._x;
origY = garbage_mc._y;
garbage_mc.onPress = function() {
  this.startDrag();
};
garbage_mc.onRelease = function() {
  this.stopDrag();
  if (eval(this._droptarget) == trashcan_mc) {
  this._visible = false;
  } else {
  this._x = origX;
  this._y = origY;
  }
};
```

### Voir également

`startDrag (méthode MovieClip.startDrag)`, `stopDrag (méthode MovieClip.stopDrag)`, `Fonction eval`

## duplicateMovieClip (méthode MovieClip.duplicateMovieClip)

```
public duplicateMovieClip(name:String, depth:Number, [initObject:Object]) :
  MovieClip
```

Crée une occurrence du clip spécifié lors de la lecture du fichier SWF. La lecture des clips dupliqués commence toujours à l'image 1, quelle que soit l'image dans laquelle se trouve le clip initial lorsque vous appelez la méthode `duplicateMovieClip()`. Les variables du clip parent ne sont pas copiées dans le clip dupliqué. Les clips créés avec la méthode `duplicateMovieClip()` ne sont pas dupliqués si vous appelez la méthode `duplicateMovieClip()` sur leur parent. Si le clip parent est supprimé, le clip dupliqué l'est également. Si vous avez chargé un clip via la classe `MovieClip.loadMovie()` ou le MovieClipLoader, le contenu du fichier SWF n'est pas dupliqué. Cela signifie que vous ne pouvez pas économiser de la bande passante en chargeant un fichier JPEG, GIF, PNG ou SWF, puis en dupliquant le clip.

Comparez cette méthode à la version fonction globale de `duplicateMovieClip()`. La version globale de cette méthode nécessite un paramètre spécifiant le clip cible à dupliquer. Ce type de paramètre n'est pas nécessaire pour la version classe MovieClip, dans la mesure où la cible de la méthode est l'occurrence de clip pour laquelle la méthode est appelée. De plus, la version globale de `duplicateMovieClip()` ne prend en charge ni le paramètre `initobject` ni la valeur renvoyée par une référence à la nouvelle occurrence de MovieClip.

**Disponibilité :** ActionScript 1.0 ; Flash Player 5

## Paramètres

*`name`*`:String` - Identifiant unique pour le clip dupliqué.

*`depth`*`:Number` - Entier unique spécifiant la profondeur à laquelle le nouveau clip doit être placé. Utilisez la profondeur -16384 pour placer la nouvelle occurrence de clip sous l'ensemble des contenus créés dans l'environnement de programmation. Les valeurs comprises entre -16383 et -1, inclus, sont réservées à l'environnement de programmation et ne doivent pas être utilisées avec cette méthode. Les valeurs de profondeur restantes vont de 0 à 1048575, inclus.

*`initObject`*`:Object` [facultatif] - (Pris en charge à partir de Flash Player 6.) Objet contenant les propriétés permettant de remplir le clip dupliqué. Ce paramètre permet aux clips créés de façon dynamique de recevoir des paramètres. Si `initObject` n'est pas un objet, il est ignoré. Toutes les propriétés de `initObject` sont copiées dans la nouvelle occurrence. Les propriétés spécifiées avec `initObject` sont disponibles pour la fonction constructeur.

## Valeur renvoyée

`MovieClip` - Référence au clip dupliqué (pris en charge à partir de Flash Player 6).

## Exemple

L'exemple suivant duplique un nouveau MovieClip un certain nombre de fois et présente la cible pour chaque double.

```
var container:MovieClip = setUpContainer();
var ln:Number = 10;
var spacer:Number = 1;
var duplicate:MovieClip;
for(var i:Number = 1; i < ln; i++) {
  var newY:Number = i * (container._height + spacer);
  duplicate = container.duplicateMovieClip("clip-" + i, i, {_y:newY});
  trace(duplicate); // _level0.clip-[number]
}

function setUpContainer():MovieClip {
```

```
  var mc:MovieClip = this.createEmptyMovieClip("container",
  this.getNextHighestDepth());
  var w:Number = 100;
  var h:Number = 20;
  mc.beginFill(0x333333);
  mc.lineTo(w, 0);
  mc.lineTo(w, h);
  mc.lineTo(0, h);
  mc.lineTo(0, 0);
  mc.endFill();
  return mc;
}
```

La méthode `MovieClip.getNextHighestDepth()` utilisée dans cet exemple nécessite Flash Player 7 ou une version ultérieure. Si votre fichier SWF comprend un composant de la version 2, utilisez les composants de la version 2 de la classe DepthManager au lieu de la méthode `MovieClip.getNextHighestDepth()`.

### Voir également

`loadMovie (méthode MovieClip.loadMovie)`, `removeMovieClip (méthode MovieClip.removeMovieClip)`, `Fonction duplicateMovieClip`

## enabled (propriété MovieClip.enabled)

```
public enabled : Boolean
```

Valeur booléenne indiquant si un clip est activé. La valeur par défaut de `enabled` est `true`. Si `enabled` est défini sur `false`, les méthodes de rappel et les gestionnaires d'événements `on` *`action`* du clip ne sont plus appelés, et les images Dessus, Abaissé et Haut sont désactivées. La propriété `enabled` n'affecte pas le scénario du clip ; si un clip est en cours de lecture, celle-ci continue. Le clip continue à recevoir des événements de clips (par exemple, `mouseDown`, `mouseUp`, `keyDown`, et `keyUp`).

La propriété `enabled` gère uniquement les propriétés spécifiques aux boutons d'un clip. Vous pouvez modifier la propriété `enabled` à tout moment ; le clip modifié est immédiatement activé ou désactivé. La propriété `enabled` peut être extraite d'un objet prototype. Si la propriété `enabled` est définie sur `false`, l'objet n'est pas inclus dans l'ordre de tabulation automatique.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

### Exemple

L'exemple suivant désactive le clip `circle_mc` lorsque l'utilisateur clique dessus :

```
circle_mc.onRelease = function() {
```

```
  trace("disabling the "+this._name+" movie clip.");
  this.enabled = false;
};
```

# endFill (méthode MovieClip.endFill)

```
public endFill() : Void
```

Applique un remplissage aux lignes et aux courbes ajoutées depuis le dernier rappel à `beginFill()` ou `beginGradientFill()`. Flash utilise le remplissage spécifié lors de l'appel précédent de `beginFill()` ou `beginGradientFill()`. Si la position de dessin actuelle n'est pas égale à la position précédente spécifiée dans une méthode `moveTo()` et si un remplissage est défini, le tracé est fermé à l'aide d'une ligne, puis rempli.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

## Exemple

L'exemple suivant crée un carré rouge sur la scène :

```
this.createEmptyMovieClip("square_mc", this.getNextHighestDepth());
square_mc.beginFill(0xFF0000);
square_mc.moveTo(10, 10);
square_mc.lineTo(100, 10);
square_mc.lineTo(100, 100);
square_mc.lineTo(10, 100);
square_mc.lineTo(10, 10);
square_mc.endFill();
```

La méthode `MovieClip.getNextHighestDepth()` utilisée dans cet exemple nécessite Flash Player 7 ou une version ultérieure. Si votre fichier SWF comprend un composant de la version 2, utilisez les composants de la version 2 de la classe DepthManager au lieu de la méthode `MovieClip.getNextHighestDepth()`.

Vous trouverez également un exemple dans le fichier drawingapi.fla du dossier Samples\ActionScript\DrawingAPI. La liste suivante présente les chemins type vers ce dossier :

- Windows : \Program Files\Macromedia\Flash 8\Samples et Tutorials\Samples\
- Macintosh : HD/Applications/Macromedia Flash 8/Samples et Tutorials/Samples/

## Voir également

`beginFill (méthode MovieClip.beginFill)`, `beginGradientFill (méthode MovieClip.beginGradientFill)`, `moveTo (méthode MovieClip.moveTo)`

# filters (propriété MovieClip.filters)

```
public filters : Array
```

Un tableau indexé contenant tous les objets filtre associés au clip. Le package flash.filters contient plusieurs classes qui définissent des filtres spécifiques.

Ces filtres peuvent s'appliquer dans l'outil de programmation de Flash pendant la phase de conception ou d'exécution du code ActionScript. Pour appliquer un filtre avec ActionScript, vous devez créer une copie temporaire de l'intégralité du tableau `MovieClip.filters`, modifier le tableau temporaire, puis reporter les valeurs de ce tableau temporaire dans le tableau `MovieClip.filters`. Vous ne pouvez pas ajouter directement un nouvel objet filtre au tableau `MovieClip.filters`. Le code suivant n'a aucun effet sur le clip cible, appelé `myMC` :

```
myMC.filters[0].push(myDropShadow);
```

Pour ajouter un filtre avec ActionScript, vous devez suivre les étapes ci-dessous (dans cet exemple le clip est appelé myMC) :

- Créez un objet filtre avec la fonction constructeur de la classe de filtre retenue.
- Affectez la valeur du tableau `myMC.filters` à un tableau temporaire, tel qu'un tableau appelé `myFilters`.
- Ajoutez le nouvel objet filtre au tableau temporaire, `myFilters`.
- Affectez la valeur du tableau temporaire au tableau `myMC.filters`.

Si le tableau `filters` est vide, il n'est pas nécessaire d'utiliser un tableau temporaire. Par contre, vous pouvez affecter directement un littéral de tableau contenant un ou plusieurs des objets filtre que vous avez créés.

Pour modifier un objet filtre existant, que ce dernier ait été créé pendant la phase de conception ou d'exécution, vous devez appliquer la technique de modification d'une copie du tableau `filters` :

- Affectez la valeur du tableau `myMC.filters` à un tableau temporaire, tel qu'un tableau appelé `myFilters`.
- Modifiez la propriété avec le tableau temporaire, `myFilters`. Par exemple, si vous souhaitez définir la propriété quality du premier filtre du tableau, utilisez le code suivant : `myList[0].quality = 1;`
- Affectez la valeur du tableau temporaire au tableau `myMC.filters`.

Pour supprimer les filtres pour un clip, définissez `filters` sur un tableau vide (`[]`).

Lors du chargement, si un clip est associé à un filtre, ce clip se place en mémoire cache en tant que bitmap transparent. A partir de ce stade, tant que le clip possède une liste de filtres valide, le lecteur place le clip en mémoire cache au format bitmap. Ce bitmap source est ensuite repris en tant qu'image source pour les effets de filtrage. Tout clip comporte généralement deux bitmaps : l'un correspond au clip d'origine sans filtres et l'autre à l'image finale après filtrage. L'image finale est utilisée pour le rendu. Tant que le clip ne change pas, l'image source ne nécessite aucune mise à jour.

Si vous manipulez un tableau `filters` contenant plusieurs filtres et devez suivre le type de filtre affecté à chaque index de tableau, vous pouvez conserver votre propre tableau `filters` et utiliser une structure de données distincte pour suivre le type de filtre associé à chaque index de tableau. Il n'existe aucune méthode simple permettant de déterminer le type de filtre associé à chaque index de tableau `filters`.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

### Exemple

L'exemple suivant ajoute un filtre d'ombre portée à un clip appelé `myMC` :

```
var myDropFilter = new flash.filters.DropShadowFilter();
var myFilters:Array = myMC.filters;
myFilters.push(myDropFilter);
myMC.filters = myFilters;
```

L'exemple suivant donne au paramètre `quality` du premier filtre du tableau la valeur 15 (cet exemple ne peut fonctionner que si au moins un objet filtre a été associé au clip `myMC`) :

```
var myList:Array = myMC.filters;
myList[0].quality = 15;
myMC.filters = myList;
```

### Voir également

## focusEnabled (propriété MovieClip.focusEnabled)

```
public focusEnabled : Boolean
```

Si la valeur est `undefined` ou `false`, un clip ne peut pas recevoir le focus d'entrée sauf s'il s'agit d'un bouton. Si la valeur de la propriété `focusEnabled` a pour valeur `true`, un clip peut recevoir le focus d'entrée même s'il ne s'agit pas d'un bouton.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

### Exemple

L'exemple suivant définit la propriété `focusEnabled` pour un clip `my_mc` sur `false` :

```
my_mc.focusEnabled = false;
```

# _focusrect (propriété MovieClip._focusrect)

```
public _focusrect : Boolean
```

Valeur booléenne indiquant si un clip est entouré d'un rectangle jaune lorsqu'il a le focus clavier. Cette propriété peut annuler la propriété `_focusrect` globale. La valeur par défaut de la propriété `_focusrect` d'une occurrence de clip est `null`, ce qui signifie que l'occurrence de clip n'annule pas la propriété `_focusrect` globale. Si la propriété `_focusrect` d'une occurrence de clip est définie sur `true` ou `false`, elle annule le paramètre de la propriété globale `_focusrect` de l'occurrence de clip unique.

Dans les fichiers SWF de Flash Player 4 ou Flash Player 5, la propriété `_focusrect` contrôle la propriété `_focusrect` globale. Il s'agit d'une valeur booléenne. Ce comportement a été modifié dans Flash Player 6 et les versions ultérieures afin de permettre la personnalisation de la propriété `_focusrect` au niveau d'un clip individuel.

Si la propriété `_focusrect` est définie sur `false`, l'accès clavier au clip est limité à la touche de tabulation. Toutes les autres touches, ce qui inclut la touche Entrée et les touches directionnelles, sont ignorées. Pour restaurer l'intégralité de l'accès clavier, vous devez définir `_focusrect` sur `true`.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

## Exemple

Cet exemple démontre comment masquer le rectangle jaune qui entoure une occurrence de clip donnée dans un fichier SWF lorsque ce dernier reçoit le focus dans une fenêtre de navigation. Créez trois clips appelés `mc1_mc`, `mc2_mc`, et `mc3_mc`, puis ajoutez le code ActionScript suivant sur l'image 1 du scénario :

```
mc1_mc._focusrect = true;
mc2_mc._focusrect = false;
mc3_mc._focusrect = true;

mc1_mc.onRelease = traceOnRelease;
mc3_mc.onRelease = traceOnRelease;

function traceOnRelease() {
  trace(this._name);
}
```

Testez le fichier SWF dans la fenêtre d'un navigateur en sélectionnant Fichier > Aperçu avant publication > HTML. Donnez le focus au fichier SWF en cliquant dessus dans la fenêtre du navigateur, puis appuyez sur la touche de tabulation pour déplacer le focus vers les différentes occurrences. Vous ne pouvez pas exécuter de code pour ce clip dans le navigateur en appuyant sur Entrée ou la barre d'espace lorsque `_focusrect` est désactivé.

En outre, vous pouvez tester votre fichier SWF dans l'environnement de test. Sélectionnez Contrôle > Désactiver les raccourcis clavier dans le menu principal de l'environnement de test. Ceci permet d'afficher le rectangle de focus entourant les occurrences dans le fichier SWF.

### Voir également

`_focusrect, propriété`, `_focusrect (propriété Button._focusrect)`

## _framesloaded (propriété MovieClip._framesloaded)

```
public _framesloaded : Number [lecture seule]
```

Le nombre d'images à charger à partir d'un fichier SWF en diffusion continue. Cette propriété est utile pour déterminer si le contenu d'une image spécifique, et de toutes les images qui la précèdent, est chargé et est disponible localement dans le navigateur. Elle est également utile pour contrôler le téléchargement de fichiers SWF volumineux. Par exemple, vous voudrez peut-être afficher un message aux utilisateurs indiquant que le chargement du fichier SWF ne commence pas tant que le chargement d'une image spécifiée dans le fichier SWF n'est pas terminé.

**Disponibilité :** ActionScript 1.0 ; Flash Player 4

### Exemple

L'exemple suivant utilise la propriété `_framesloaded` pour activer un fichier SWF lorsque toutes les images sont chargées. Si certaines images ne sont pas chargées, la propriété `_xscale` de l'occurrence de clip `bar_mc` est augmentée proportionnellement pour créer une barre de progression.

Entrez le code ActionScript suivant dans l'image 1 du scénario :

```
var pctLoaded:Number = Math.round(this.getBytesLoaded()/
  this.getBytesTotal()*100);
bar_mc._xscale = pctLoaded;
```

Ajoutez le code suivant sur l'image 2 :

```
if (this._framesloaded < this._totalframes) {
  this.gotoAndPlay(1);
} else {
  this.gotoAndStop(3);
}
```

Placez le contenu dans ou après l'image 3. Puis ajoutez le code suivant sur l'image 3 :

```
stop();
```

### Voir également

`MovieClipLoader`

# getBounds (méthode MovieClip.etBounds)

```
public getBounds(bounds:Object) : Object
```

Renvoie des propriétés qui sont les valeurs de coordonnées *x* et *y* minimales et maximales du clip, à partir du paramètre `bounds`.

*Remarque :* Utilisez les méthodes `MovieClip.localToGlobal()` et `MovieClip.globalToLocal()` pour convertir les coordonnées locales du clip en coordonnées de scène, ou des coordonnées de scène en coordonnées locales, respectivement.

Vous pouvez étendre les méthodes et les gestionnaires d'événements de la classe MovieClip en créant une sous-classe.

**Disponibilité :** ActionScript 1.0 ; Flash Player 5

## Paramètres

*`bounds`*`:Object` - Chemin cible du scénario dont vous souhaitez utiliser le système de coordonnées en tant que point de référence.

## Valeur renvoyée

`Object` - Objet avec les propriétés `xMin`, `xMax`, `yMin`, et `yMax`.

## Exemple

L'exemple suivant crée un clip appelé `square_mc`. Le code trace un carré pour ce clip et utilise la méthode `MovieClip.getBounds()` pour afficher la valeur des coordonnées de l'occurrence dans le panneau de sortie.

```
this.createEmptyMovieClip("square_mc", 1);
square_mc._x = 10;
square_mc._y = 10;
square_mc.beginFill(0xFF0000);
square_mc.moveTo(0, 0);
square_mc.lineTo(100, 0);
square_mc.lineTo(100, 100);
square_mc.lineTo(0, 100);
square_mc.lineTo(0, 0);
square_mc.endFill();

var bounds_obj:Object = square_mc.getBounds(this);
for (var i in bounds_obj) {
  trace(i+" --> "+bounds_obj[i]);
}
```

Les informations suivantes apparaissent dans le panneau de sortie :

```
yMax --> 110
yMin --> 10
```

```
xMax --> 110
xMin --> 10
```

### Voir également

getRect (méthode MovieClip.getRect), globalToLocal (méthode MovieClip.globalToLocal), localToGlobal (méthode MovieClip.localToGlobal)

## getBytesLoaded (méthode MovieClip.getBytesLoaded)

```
public getBytesLoaded() : Number
```

Renvoie le nombre d'octets déjà chargés (transmis en continu) pour le clip. Vous pouvez comparer cette valeur à la valeur renvoyée par `MovieClip.getBytesTotal()` afin de déterminer le pourcentage de chargement d'un clip.

Vous pouvez étendre les méthodes et les gestionnaires d'événements de la classe MovieClip en créant une sous-classe.

**Disponibilité :** ActionScript 1.0 ; Flash Player 5

### Valeur renvoyée

`Number` - Entier indiquant le nombre d'octets chargés.

### Exemple

L'exemple suivant utilise la propriété `_framesloaded` pour activer un fichier SWF lorsque toutes les images sont chargées. Si certaines images ne sont pas chargées, la propriété `_xscale` de l'occurrence de clip `loader` est augmentée proportionnellement pour créer une barre de progression.

Entrez le code ActionScript suivant sur l'image 1 du scénario :

```
var pctLoaded:Number = Math.round(this.getBytesLoaded()/
  this.getBytesTotal() * 100);
bar_mc._xscale = pctLoaded;
```

Ajoutez le code suivant sur l'image 2 :

```
if (this._framesloaded<this._totalframes) {
  this.gotoAndPlay(1);
} else {
  this.gotoAndStop(3);
}
```

Placez le contenu dans ou après l'image 3, puis ajoutez le code suivant sur l'image 3 :

```
stop();
```

### Voir également

getBytesTotal (méthode MovieClip.getBytesTotal)

## getBytesTotal (méthode MovieClip.getBytesTotal)

```
public getBytesTotal() : Number
```

Renvoie la taille, en octets, du clip. Pour les clips externes (le fichier SWF racine ou un clip chargé dans une cible ou un niveau), la valeur de retour est la taille non compressée du fichier SWF.

Vous pouvez étendre les méthodes et les gestionnaires d'événements de la classe MovieClip en créant une sous-classe.

**Disponibilité :** ActionScript 1.0 ; Flash Player 5

### Valeur renvoyée

`Number` - Entier indiquant la taille totale, en octets, du clip.

### Exemple

L'exemple suivant utilise la propriété `_framesloaded` pour activer un fichier SWF lorsque toutes les images sont chargées. Si certaines images ne sont pas chargées, la propriété `_xscale` de l'occurrence de clip `loader` est augmentée proportionnellement pour créer une barre de progression.

Entrez le code ActionScript suivant sur l'image 1 du scénario :

```
var pctLoaded:Number = Math.round(this.getBytesLoaded()/
  this.getBytesTotal()*100);
bar_mc._xscale = pctLoaded;
```

Ajoutez le code suivant sur l'image 2 :

```
if (this._framesloaded<this._totalframes) {
  this.gotoAndPlay(1);
} else {
  this.gotoAndStop(3);
}
```

Placez le contenu dans ou après l'image 3. Puis ajoutez le code suivant sur l'image 3 :

```
stop();
```

### Voir également

getBytesLoaded (méthode MovieClip.getBytesLoaded)

# getDepth (méthode MovieClip.getDepth)

```
public getDepth() : Number
```

Renvoie la profondeur d'une occurrence de clip.

Tout clip, bouton et champ texte est associé à une profondeur unique qui détermine l'aspect de l'objet devant ou derrière d'autres objets. Les objets dont les valeurs de profondeur sont plus importantes s'affichent au premier plan. Le contenu créé lors de la conception (dans l'outil de programmation) commence à une profondeur de - 16383.

Vous pouvez étendre les méthodes et les gestionnaires d'événements de la classe MovieClip en créant une sous-classe.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

## Valeur renvoyée

`Number` - Profondeur du clip.

## Exemple

Le code suivant suit la profondeur de toutes les occurrences de clip de la scène :

```
for (var i in this) {
  if (typeof (this[i]) == "movieclip") {
  trace("movie clip '"+this[i]._name+"' is at depth "+this[i].getDepth());
  }
}
```

## Voir également

`getInstanceAtDepth (méthode MovieClip.getInstanceAtDepth)`, `getNextHighestDepth (méthode MovieClip.getNextHighestDepth)`, `swapDepths (méthode MovieClip.swapDepths)`, `getDepth (méthode TextField.getDepth)`, `getDepth (méthode Button.getDepth)`

# getInstanceAtDepth (méthode MovieClip.getInstanceAtDepth)

```
public getInstanceAtDepth(depth:Number) : MovieClip
```

Permet de déterminer si une profondeur spécifique est déjà occupée par un clip. Vous pouvez utiliser cette méthode avant d'utiliser `MovieClip.attachMovie()`, `MovieClip.duplicateMovieClip()`, ou `MovieClip.createEmptyMovieClip()` pour déterminer si le paramètre de profondeur à transmettre à l'une de ces méthodes contient déjà un clip.

Vous pouvez étendre les méthodes et les gestionnaires d'événements de la classe MovieClip en créant une sous-classe.

**Disponibilité :** ActionScript 1.0 ; Flash Player 7

### Paramètres

*depth*:Number - Entier qui spécifie le niveau de profondeur à déterminer.

### Valeur renvoyée

MovieClip - Référence à l'occurrence MovieClip qui se trouve à la profondeur spécifiée, ou undefined si aucun clip ne se trouve à cette profondeur.

### Exemple

L'exemple suivant affiche la profondeur occupée par l'occurrence de clip triangle dans le panneau de sortie :

```
this.createEmptyMovieClip("triangle", 1);

triangle.beginFill(0x0000FF, 100);
triangle.moveTo(100, 100);
triangle.lineTo(100, 150);
triangle.lineTo(150, 100);
triangle.lineTo(100, 100);

trace(this.getInstanceAtDepth(1)); // output: _level0.triangle
```

### Voir également

attachMovie (méthode MovieClip.attachMovie), duplicateMovieClip (méthode MovieClip.duplicateMovieClip), createEmptyMovieClip (méthode MovieClip.createEmptyMovieClip), getDepth (méthode MovieClip.getDepth), getNextHighestDepth (méthode MovieClip.getNextHighestDepth), swapDepths (méthode MovieClip.swapDepths)

## getNextHighestDepth (méthode MovieClip.getNextHighestDepth)

```
public getNextHighestDepth() : Number
```

Permet de déterminer une valeur de profondeur que vous pouvez transmettre à MovieClip.attachMovie(), MovieClip.duplicateMovieClip(), ou MovieClip.createEmptyMovieClip() afin de vous assurer que Flash rende le clip devant tous les autres objets sur les mêmes niveau et calque dans le clip actuel. La valeur renvoyée est plus grande ou égale à 0 (autrement dit, les nombres négatifs ne sont pas renvoyés).

Vous pouvez étendre les méthodes et les gestionnaires d'événements de la classe MovieClip en créant une sous-classe.

*Remarque :* Si vous utilisez des composants de la version 2, n'utilisez pas cette méthode. Si vous placez un composant de la version 2 sur la scène ou dans la bibliothèque, la méthode `getNextHighestDepth()` renvoie parfois une valeur de profondeur de 1048676, qui est en dehors de la limite valide. Si vous utilisez des composants de la version 2, vous devez toujours utiliser les composants de la version 2 de la classe DepthManager.

**Disponibilité :** ActionScript 1.0 ; Flash Player 7

## Valeur renvoyée

`Number` - Entier reflétant le prochain index de profondeur disponible rendu au-dessus de tous les autres objets au même niveau et sur le même calque dans le clip.

## Exemple

L'exemple suivant dessine trois occurrences de clip, en utilisant la méthode `getNextHighestDepth()` comme paramètre `depth` de la méthode `createEmptyMovieClip()` et étiquette chaque clip avec sa profondeur :

```
for (i = 0; i < 3; i++) {
  drawClip(i);
}

function drawClip(n:Number):Void {
  this.createEmptyMovieClip("triangle" + n, this.getNextHighestDepth());
  var mc:MovieClip = eval("triangle" + n);
  mc.beginFill(0x00aaFF, 100);
  mc.lineStyle(4, 0xFF0000, 100);
  mc.moveTo(0, 0);
  mc.lineTo(100, 100);
  mc.lineTo(0, 100);
  mc.lineTo(0, 0);
  mc._x = n * 30;
  mc._y = n * 50
  mc.createTextField("label", this.getNextHighestDepth(), 20, 50, 200, 200)
  mc.label.text = mc.getDepth();
}
```

## Voir également

`getDepth (méthode MovieClip.getDepth)`, `getInstanceAtDepth (méthode MovieClip.getInstanceAtDepth)`, `swapDepths (méthode MovieClip.swapDepths)`, `attachMovie (méthode MovieClip.attachMovie)`, `duplicateMovieClip (méthode MovieClip.duplicateMovieClip)`, `createEmptyMovieClip (méthode MovieClip.createEmptyMovieClip)`

# getRect (méthode MovieClip.getRect)

`public getRect(bounds:Object) : Object`

Renvoie des propriétés qui sont les valeurs de coordonnées *x* et *y* minimales et maximales du clip, à partir du paramètre `bounds` ce qui exclut tout tracé sur les formes. Les valeurs renvoyées par `getRect()` sont identiques ou inférieures aux valeurs renvoyées par `MovieClip.getBounds()`.

*Remarque :* Utilisez les méthodes `MovieClip.localToGlobal()` et `MovieClip.globalToLocal()` pour convertir les coordonnées locales du clip en coordonnées de scène, ou des coordonnées de scène en coordonnées locales, respectivement.

Vous pouvez étendre les méthodes et les gestionnaires d'événements de la classe MovieClip en créant une sous-classe.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

## Paramètres

*`bounds`*`:Object` - Chemin cible du scénario dont vous souhaitez utiliser le système de coordonnées en tant que point de référence.

## Valeur renvoyée

`Object` - Un objet avec les propriétés `xMin`, `xMax`, `yMin`, et `yMax`.

## Exemple

L'exemple suivant crée un clip et dessine à l'intérieur un carré avec une épaisseur de trait de 4 pixels. L'exemple appel ensuite les méthodes `MovieClip.getBounds()` et `MovieClip.getRect()` pour montrer la différence entre les deux. La méthode `getBounds()` renvoie les valeurs de coordonnées minimales et maximales de tout le clip, y compris l'épaisseur de trait du carré. La méthode `getRect()` renvoie les valeurs de coordonnées minimales et maximales à l'exclusion de l'épaisseur de trait de 4 pixels.

```
this.createEmptyMovieClip("square_mc", 1);
square_mc._x = 10;
square_mc._y = 10;
square_mc.beginFill(0xFF0000);
square_mc.lineStyle(4, 0xFF00FF, 100, true, "none", "round", "miter", 1);
square_mc.moveTo(0, 0);
square_mc.lineTo(100, 0);
square_mc.lineTo(100, 100);
square_mc.lineTo(0, 100);
square_mc.lineTo(0, 0);
square_mc.endFill();

var bounds_obj:Object = square_mc.getBounds(this);
```

```
trace("getBounds() output:");
for (var i in bounds_obj) {
  trace(i+" --> "+bounds_obj[i]);
}

var rect_obj:Object = square_mc.getRect(this);
trace("getRect() output:");
for (var i in rect_obj) {
  trace(i+" --> "+rect_obj[i]);
}
```

L'instruction `trace()` renvoie l'image suivante.

```
getBounds() output:
yMax --> 112
yMin --> 8
xMax --> 112
xMin --> 8
getRect() output:
yMax --> 110
yMin --> 10
xMax --> 110
xMin --> 10
```

### Voir également

`getBounds (méthode MovieClip.etBounds)`, `globalToLocal (méthode MovieClip.globalToLocal)`, `localToGlobal (méthode MovieClip.localToGlobal)`

## getSWFVersion (méthode MovieClip.getSWFVersion)

```
public getSWFVersion() : Number
```

Renvoie un entier indiquant la version de Flash Player pour laquelle le clip a été publié. Si le clip est un fichier JPEG, GIF ou PNG, ou si une erreur se produit et que Flash Player ne peut pas déterminer la version SWF du clip, la valeur -1 est renvoyée.

Vous pouvez étendre les méthodes et les gestionnaires d'événements de la classe MovieClip en créant une sous-classe.

**Disponibilité :** ActionScript 1.0 ; Flash Player 7

### Valeur renvoyée

`Number` - Entier spécifiant la version de Flash Player ciblée une fois le chargement du fichier SWF dans le clip publié.

### Exemple

L'exemple suivant crée un nouveau container et renvoie la valeur de `getSWFVersion()`. Il utilise ensuite MovieClipLoader pour charger un fichier SWF externe publié pour Flash Player 7 et renvoie la valeur de `getSWFVersion()` après le déclenchement du gestionnaire `onLoadInit`.

```
var container:MovieClip = this.createEmptyMovieClip("container",
  this.getUpperEmptyDepth());
var listener:Object = new Object();
listener.onLoadInit = function(target:MovieClip):Void {
  trace("target: " + target.getSWFVersion()); // target: 7
}
var mcLoader:MovieClipLoader = new MovieClipLoader();
mcLoader.addListener(listener);
trace("container: " + container.getSWFVersion()); // container: 8
mcLoader.loadClip("FlashPlayer7.swf", container);
```

## getTextSnapshot (méthode MovieClip.getTextSnapshot)

```
public getTextSnapshot() : TextSnapshot
```

Renvoie un objet TextSnapshot contenant le texte de tous les champs de texte statiques contenus dans le clip spécifié ; le texte des clips enfants n'est pas inclus. Cette méthode renvoie toujours un objet TextSnapshot.

Flash concatène le texte et le place dans l'objet TextSnapshot de manière à refléter l'ordre d'index de tabulation des champs de texte statiques dans le clip. Les champs de texte n'ayant pas de valeurs d'index de tabulation sont placés dans un ordre aléatoire dans l'objet, et précèdent le texte issu de champs ayant des valeurs d'index de tabulation. Aucun saut de ligne ou formatage ne permet d'indiquer l'endroit où se termine un champ et où commence le suivant.

*Remarque :* Vous ne pouvez pas spécifier de valeur d'index de tabulation pour le texte statique dans Flash. Cependant, d'autres produits permettent de le faire (par exemple, Macromedia FlashPaper).

Le contenu de l'objet TextSnapshot n'est pas dynamique ; autrement dit, si le clip se déplace vers une autre image, ou est modifié de quelque manière que ce soit (par exemple, si les objets du clip sont ajoutés ou supprimés), l'objet TextSnapshot peut ne pas représenter le texte actuel dans le clip. Pour vous assurer que le contenu de l'objet est à jour, émettez cette commande autant de fois que nécessaire.

Vous pouvez étendre les méthodes et les gestionnaires d'événements de la classe MovieClip en créant une sous-classe.

**Disponibilité :** ActionScript 1.0 ; Flash Player 7

### Valeur renvoyée

`TextSnapshot` - Objet TextSnapshot qui contient le texte statique du clip.

### Exemple

L'exemple suivant montre comment utiliser cette méthode. Pour utiliser ce code, placez un champ de texte statique contenant le texte « TextSnapshot Example » sur la scène.

```
var textSnap:TextSnapshot = this.getTextSnapshot();
trace(textSnap.getText(0, textSnap.getCount(), false));
```

### Voir également

`TextSnapshot`

## getURL (méthode MovieClip.getURL)

```
public getURL(url:String, [window:String], [method:String]) : Void
```

Charge un document à partir de l'URL spécifiée dans la fenêtre spécifiée. Vous pouvez également utiliser la méthode `getURL()` pour transmettre des variables à une autre application définie à l'URL en utilisant une méthode `GET` ou `POST`.

Les pages Web qui hébergent un contenu Flash doivent définir de façon explicite l'attribut `allowScriptAccess` pour autoriser ou bloquer la programmation de Flash Player à l'aide de code HTML (dans la balise `PARAM` d'Internet Explorer ou `EMBED` de Netscape Navigator) :

- Lorsque `allowScriptAccess` a pour valeur `"never"`, les scripts externes échouent systématiquement.
- Lorsque `allowScriptAccess` a pour valeur `"always"`, les scripts externes sont acceptés systématiquement.
- Lorsque `allowScriptAccess` a pour valeur `"sameDomain"` (pris en charge par les fichiers SWF à partir de la version 8 ), les scripts externes sont autorisés si le fichier SWF provient du même domaine que la page Web hébergeante.
- Si `allowScriptAccess` n'est pas spécifié par une page HTML, la valeur par défaut est `"sameDomain"` pour les fichiers SWF de la version 8, et la valeur par défaut est `"always"` pour les fichiers SWF des versions antérieures.

Lorsque vous utilisez cette méthode, tenez compte du modèle de sécurité Flash Player. Pour Flash Player 8, la méthode n'est pas autorisée si le fichier SWF appelant est dans le sandbox local avec système de fichier et que la ressource n'est pas locale.

Pour plus d'informations, voir les sections suivantes :

- Chapitre 17, « Fonctionnement de la sécurité », du manuel *Formation à ActionScript 2.0 dans Flash*
- Livre blanc sur la sécurité de Flash Player 8 disponible à l'adresse : http://www.macromedia.com/go/fp8_security
- Livre blanc sur les API relatif à la sécurité de Flash Player 8 disponible à l'adresse : http://www.macromedia.com/go/fp8_security_apis

Vous pouvez étendre les méthodes et les gestionnaires d'événements de la classe MovieClip en créant une sous-classe.

**Disponibilité :** ActionScript 1.0 ; Flash Player 5

## Paramètres

*url*:String - URL permettant d'obtenir le document.

*window*:String [facultatif] - Paramètre spécifiant le nom, le cadre ou l'expression qui spécifie la fenêtre ou le cadre HTML où le document est chargé. Vous pouvez également utiliser l'un des noms cible réservés suivants : `_self` spécifie l'image actuelle dans la fenêtre actuelle, `_blank` spécifie une nouvelle fenêtre, `_parent` spécifie le parent de l'image actuelle et `_top` spécifie l'image de plus haut niveau dans la fenêtre actuelle.

*method*:String [facultatif] - Chaîne (`"GET"` ou `"POST"`) qui spécifie une méthode d'envoi de variables associées au fichier SWF à charger. En l'absence de ces variables, omettez ce paramètre ; sinon, spécifiez si les variables doivent être chargées avec la méthode `GET` ou `POST`. La méthode `GET` ajoute les variables à la fin de l'URL et est utilisée lorsque les variables sont peu nombreuses. La méthode `POST` place les variables dans un en-tête HTTP distinct et s'applique aux variables longues de type chaîne.

## Exemple

Le code ActionScript suivant crée une occurrence de clip et ouvre le site Web de Macromedia dans une nouvelle fenêtre :

```
this.createEmptyMovieClip("loader_mc", this.getNextHighestDepth());
loader_mc.getURL("http://www.macromedia.com", "_blank");
```

La méthode `getURL()` permet également d'envoyer des variables à un script distant, côté serveur, comme indiqué dans le code suivant :

```
this.createEmptyMovieClip("loader_mc", this.getNextHighestDepth());
loader_mc.username = "some user input";
loader_mc.password = "random string";
loader_mc.getURL("http://www.flash-mx.com/mm/viewscope.cfm", "_blank",
  "GET");
```

La méthode `MovieClip.getNextHighestDepth()` utilisée dans ces exemples nécessite Flash Player 7 ou une version ultérieure. Si votre fichier SWF comprend un composant de la version 2, utilisez les composants de la version 2 de la classe DepthManager au lieu de la méthode `MovieClip.getNextHighestDepth()`.

Lorsque vous utilisez cette méthode, tenez compte du modèle de sécurité Flash Player.

- Pour Flash Player 8, la méthode `MovieClip.getURL()` n'est pas autorisée si le fichier SWF appelant est dans le sandbox local avec système de fichier et que la ressource n'est pas locale.

Pour plus d'informations, voir les sections suivantes :

- Chapitre 17, « Fonctionnement de la sécurité », dans *Formation à ActionScript 2.0 dans Flash*
- Le livre blanc sur la sécurité de Flash Player 8 disponible à l'adresse : http://www.macromedia.com/go/fp8_security
- Le livre blanc sur les API relatif à la sécurité de Flash Player 8 disponible à l'adresse : http://www.macromedia.com/go/fp8_security_apis

### Voir également

`Fonction getURL`, `sendAndLoad (méthode LoadVars.sendAndLoad)`, `send (méthode LoadVars.send)`

## globalToLocal (méthode MovieClip.globalToLocal)

```
public globalToLocal(pt:Object) : Void
```

Convertit l'objet `pt` à partir des coordonnées de scène (globales) vers les coordonnées du clip (locales).

La méthode `MovieClip.globalToLocal()` permet de convertir les coordonnées *x* et *y* à partir des valeurs relatives au coin supérieur gauche de la scène en valeurs relatives au coin supérieur gauche d'un clip donné.

Vous devez tout d'abord créer un objet générique comportant deux propriétés, x et y. Ces valeurs `x` et `y` (qui doivent être appelées `x` et `y`) sont appelées coordonnées globales dans la mesure où elles font référence au coin supérieur gauche de la scène. La propriété x représente le décalage horizontal par rapport au coin supérieur gauche. En d'autres termes, elle représente la position droite du point. Par exemple, si x = 50, le point est situé à 50 pixels à droite du coin supérieur gauche. La propriété y représente le décalage vertical par rapport au coin supérieur gauche. En d'autres termes, elle représente la position basse du point. Par exemple, si y = 20, le point est situé à 20 pixels en dessous du coin supérieur gauche. Le code suivant crée un objet générique avec ces coordonnées :

```
var myPoint:Object = new Object();
myPoint.x = 50;
myPoint.y = 20;
```

En outre, vous pouvez créer l'objet et affecter les valeurs en même temps avec une valeur Object littérale :

```
var myPoint:Object = {x:50, y:20};
```

Après avoir créé un objet point avec des coordonnées globales, vous pouvez convertir les coordonnées en coordonnées locales. La méthode `globalToLocal()` ne renvoie pas de valeurs dans la mesure où elle change les valeurs de `x` et `y` dans l'objet générique envoyé en tant que paramètre. Elle les transforme de valeurs relatives à la scène (coordonnées globales) en valeurs relatives à un clip spécifique (coordonnées locales).

Par exemple, si vous créez un clip qui est placé au point `(_x:100, _y:100)`, puis que vous transmettez le point global représentant le coin supérieur gauche de la scène `(x:0, y:0)` à la méthode `globalToLocal()`, la méthode doit convertir les valeurs `x` et `y` en coordonnées locales, soit `(x:-100, y:-100)`. Ceci est dû au fait que les coordonnées *x* et *y* sont désormais exprimées par rapport au coin supérieur gauche de votre clip et non pas par rapport au coin supérieur gauche de la scène. Ces valeurs sont négatives dans la mesure où pour se déplacer du coin supérieur gauche du clip au coin supérieur gauche de la scène, vous devez vous déplacer de 100 pixels vers la gauche (*x* négatif) et de 100 pixels vers le haut (*y* négatif).

Les coordonnées du clip ont été représentées par `_x` et `_y`, dans la mesure où il s'agit des propriétés MovieClip permettant de définir les valeurs *x* et *y* pour MovieClips. Cependant, votre objet générique utilise `x` et `y` sans le signe souligné. Le code suivant convertit les valeurs *x* et *y* en coordonnées locales :

```
var myPoint:Object = {x:0, y:0}; // Create your generic point object.
this.createEmptyMovieClip("myMovieClip", this.getNextHighestDepth());
myMovieClip._x = 100; // _x for movieclip x position
myMovieClip._y = 100; // _y for movieclip y position

myMovieClip.globalToLocal(myPoint);
trace ("x: " + myPoint.x); // output: -100
trace ("y: " + myPoint.y); // output: -100
```

Vous pouvez étendre les méthodes et les gestionnaires d'événements de la classe MovieClip en créant une sous-classe.

**Disponibilité :** ActionScript 1.0 ; Flash Player 5

## Paramètres

*`pt`*`:Object` - Nom ou identificateur d'un objet créé avec la classe générique Object. Cet objet spécifie les coordonnées *x* et *y* en tant que propriétés.

### Exemple

Cet exemple ajoute le code ActionScript suivant à un fichier FLA ou AS dans le même répertoire qu'une image appelée photo1.jpg :

```
this.createTextField("coords_txt", this.getNextHighestDepth(), 10, 10, 100,
  22);
coords_txt.html = true;
coords_txt.multiline = true;
coords_txt.autoSize = true;
this.createEmptyMovieClip("target_mc", this.getNextHighestDepth());
target_mc._x = 100;
target_mc._y = 100;
target_mc.loadMovie("photo1.jpg");

var mouseListener:Object = new Object();
mouseListener.onMouseMove = function() {
  var point:Object = {x:_xmouse, y:_ymouse};
  target_mc.globalToLocal(point);
  var rowHeaders = "<b>   \t</b><b>_x\t</b><b>_y</b>";
  var row_1 = "_root\t"+_xmouse+"\t"+_ymouse;
  var row_2 = "target_mc\t"+point.x+"\t"+point.y;
  coords_txt.htmlText = "<textformat tabstops='[100, 150]'>";
  coords_txt.htmlText += rowHeaders;
  coords_txt.htmlText += row_1;
  coords_txt.htmlText += row_2;
  coords_txt.htmlText += "</textformat>";
};
Mouse.addListener(mouseListener);
```

La méthode `MovieClip.getNextHighestDepth()` utilisée dans cet exemple nécessite Flash Player 7 ou une version ultérieure. Si votre fichier SWF comprend un composant de la version 2, utilisez les composants de la version 2 de la classe DepthManager au lieu de la méthode `MovieClip.getNextHighestDepth()`.

### Voir également

`getBounds (méthode MovieClip.etBounds)`, `localToGlobal (méthode MovieClip.localToGlobal)`, `Object`

## gotoAndPlay (méthode MovieClip.gotoAndPlay)

```
public gotoAndPlay(frame:Object) : Void
```

Commence la lecture du fichier SWF sur l'image spécifiée. Pour spécifier une séquence et une image, utilisez `gotoAndPlay()`.

Vous pouvez étendre les méthodes et les gestionnaires d'événements de la classe MovieClip en créant une sous-classe.

**Disponibilité :** ActionScript 1.0 ; Flash Player 5

### Paramètres

*frame*:Object - Nombre représentant le numéro d'image ou une chaîne représentant l'étiquette de l'image cible de la tête de lecture.

### Exemple

L'exemple suivant utilise la propriété _framesloaded pour activer un fichier SWF lorsque toutes les images sont chargées. Si certaines images ne sont pas chargées, la propriété _xscale de l'occurrence de clip loader est augmentée proportionnellement pour créer une barre de progression.

Entrez le code ActionScript suivant sur l'image 1 du scénario :

```
var pctLoaded:Number = Math.round(this.getBytesLoaded()/
  this.getBytesTotal()*100);
bar_mc._xscale = pctLoaded;
```

Ajoutez le code suivant sur l'image 2 :

```
if (this._framesloaded<this._totalframes) {
  this.gotoAndPlay(1);
} else {
  this.gotoAndStop(3);
}
```

Placez le contenu dans ou après l'image 3. Puis ajoutez le code suivant sur l'image 3 :

```
stop();
```

### Voir également

Fonction gotoAndPlay, Fonction play

## gotoAndStop (méthode MovieClip.gotoAndStop)

```
public gotoAndStop(frame:Object) : Void
```

Place la tête de lecture au niveau de l'image spécifiée du clip et l'arrête à cet endroit. Pour spécifier une séquence en plus d'une image, utilisez la méthode gotoAndStop().

Vous pouvez étendre les méthodes et les gestionnaires d'événements de la classe MovieClip en créant une sous-classe.

**Disponibilité :** ActionScript 1.0 ; Flash Player 5

### Paramètres

*frame*:Object - Numéro de l'image cible de la tête de lecture.

### Exemple

L'exemple suivant utilise la propriété `_framesloaded` pour activer un fichier SWF lorsque toutes les images sont chargées. Si certaines images ne sont pas chargées, la propriété `_xscale` de l'occurrence de clip `loader` est augmentée proportionnellement pour créer une barre de progression.

Entrez le code ActionScript suivant sur l'image 1 du scénario :

```
var pctLoaded:Number = Math.round(this.getBytesLoaded()/
  this.getBytesTotal()*100);
bar_mc._xscale = pctLoaded;
```

Ajoutez le code suivant sur l'image 2 :

```
if (this._framesloaded<this._totalframes) {
  this.gotoAndPlay(1);
} else {
  this.gotoAndStop(3);
}
```

Placez le contenu dans ou après l'image 3. Puis ajoutez le code suivant sur l'image 3 :

```
stop();
```

### Voir également

`Fonction gotoAndStop`, `Fonction stop`

## _height (propriété MovieClip._height)

```
public _height : Number
```

Hauteur du clip, en pixels.

**Disponibilité :** ActionScript 1.0 ; Flash Player 4

### Exemple

L'exemple de code suivant affiche la hauteur et la largeur d'un clip dans le panneau de sortie :

```
this.createEmptyMovieClip("image_mc", this.getNextHighestDepth());
var image_mcl:MovieClipLoader = new MovieClipLoader();
var mclListener:Object = new Object();
mclListener.onLoadInit = function(target_mc:MovieClip) {
  trace(target_mc._name+" = "+target_mc._width+" X "+target_mc._height+"
  pixels");
};
image_mcl.addListener(mclListener);

image_mcl.loadClip("example.jpg", image_mc);
```

La méthode `MovieClip.getNextHighestDepth()` utilisée dans cet exemple nécessite Flash Player 7 ou une version ultérieure. Si votre fichier SWF comprend un composant de la version 2, utilisez les composants de la version 2 de la classe DepthManager au lieu de la méthode `MovieClip.getNextHighestDepth()`.

La classe MovieClipLoader utilisée dans cet exemple nécessite Flash Player 7 ou une version ultérieure.

### Voir également

`_width (propriété MovieClip._width)`

## _highquality (propriété MovieClip._highquality)

```
public _highquality : Number
```

*Déconseillé* à partir de Flash Player 7. Il est recommandé d'utiliser `MovieClip._quality`.

Spécifie le niveau d'anti-aliasing appliqué au fichier SWF actuel. Spécifiez 2 (meilleure qualité) pour bénéficier de la meilleure qualité possible et activer le lissage de façon permanente. Spécifiez 1 (haute qualité) pour procéder à l'anti-aliasing ; ceci lisse les bitmaps si le fichier SWF ne contient pas d'animation. Spécifiez 0 (faible qualité) pour empêcher l'anti-aliasing. Cette propriété peut remplacer la propriété `_highquality` globale.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

### Exemple

Le code ActionScript suivant permet d'appliquer la meilleure qualité possible d'anti-aliasing au fichier SWF.

```
my_mc._highquality = 2;
```

### Voir également

`_quality (propriété MovieClip._quality)`, `_quality, propriété`

## hitArea (propriété MovieClip.hitArea)

```
public hitArea : Object
```

Désigne un autre clip pour faire office de zone active d'un clip. Si la propriété `hitArea` n'existe pas, ou si sa valeur est `null` ou `undefined`, le clip fait office de zone active. La valeur de la propriété `hitArea` peut être une référence à un objet de clip.

Vous pouvez modifier la propriété `hitArea` à tout moment ; le clip modifié accepte immédiatement le nouveau comportement de la zone active. Il n'est pas nécessaire que le clip désigné comme étant la zone active soit visible ; sa forme graphique, bien qu'elle ne soit pas visible, est encore détectée comme zone active.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

### Exemple

L'exemple suivant définit le clip `circle_mc` en tant que zone réactive pour le clip `square_mc`. Placez ces deux clips sur la scène et testez le document. Lorsque vous cliquez sur `circle_mc`, le clip `square_mc` indique que vous avez cliqué.

```
square_mc.hitArea = circle_mc;
square_mc.onRelease = function() {
  trace("hit! "+this._name);
};
```

Vous pouvez également définir pour le clip `circle_mc`, une propriété `visible` sur `false` pour masquer la zone réactive de `square_mc`.

```
circle_mc._visible = false;
```

### Voir également

`hitTest (méthode MovieClip.hitTest)`

## hitTest (méthode MovieClip.hitTest)

```
public hitTest() : Boolean
```

Evalue le clip pour savoir s'il recouvre ou recoupe la zone active identifiée par les paramètres de coordonnée `target` ou `x` et `y`.

Usage 1 : Compare les coordonnées *x* et *y* à la forme ou au cadre de délimitation de l'occurrence spécifiée, selon le paramètre `shapeFlag`. Si `shapeFlag` est défini sur `true`, seule la zone occupée par l'occurrence sur la scène est évaluée ; si *x* et *y* se chevauchent en un point quelconque, une valeur `true` est renvoyée. Cette évaluation est utile pour déterminer si le clip se trouve dans une zone active ou sensible spécifiée.

Usage 2 : Evalue les cadres de délimitation de l'occurrence `target` et spécifiée, et renvoie `true` s'ils se chevauchent ou se croisent en un point quelconque.

*Paramètres* `x: Number` La coordonnée *x* de la zone active sur la scène. `y: Number` Coordonnée *y* de la zone active de la scène. Les coordonnées *x* et *y* sont définies dans l'espace de coordonnées global. `shapeFlag: Boolean` Valeur booléenne indiquant s'il convient d'évaluer la forme entière de l'occurrence spécifiée (`true`), ou uniquement le cadre de délimitation (`false`). Ce paramètre peut être spécifié uniquement si la zone active est identifiée à l'aide des paramètres des coordonnées `x` et `y`. `target: Object` Chemin cible de la zone active susceptible de couvrir partiellement ou de recouvrir le clip. Le paramètre `target` représente généralement un bouton ou un champ de saisie.

**Disponibilité :** ActionScript 1.0 ; Flash Player 5

### Valeur renvoyée

`Boolean` - Valeur booléenne `true` si le clip recouvre la zone active spécifiée, `false` sinon.

### Exemple

L'exemple suivant utilise `hitTest()` pour déterminer si le clip `circle_mc` couvre ou recouvre partiellement le clip `square_mc` lorsque l'utilisateur relâche le bouton de la souris :

```
square_mc.onPress = function() {
  this.startDrag();
};
square_mc.onRelease = function() {
  this.stopDrag();
  if (this.hitTest(circle_mc)) {
  trace("you hit the circle");
  }
};
```

### Voir également

`getBounds (méthode MovieClip.etBounds)`, `globalToLocal (méthode MovieClip.globalToLocal)`, `localToGlobal (méthode MovieClip.localToGlobal)`

## lineGradientStyle (méthode MovieClip.lineGradientStyle)

```
public lineGradientStyle(fillType:String, colors:Array, alphas:Array,
  ratios:Array, matrix:Object, [spreadMethod:String],
  [interpolationMethod:String], [focalPointRatio:Number]) : Void
```

Spécifie un style de trait utilisé par Flash pour les prochains appels des méthodes `lineTo()` et `curveTo()`, jusqu'à ce que vous appeliez la méthode `lineStyle()` ou la méthode `lineGradientStyle()` avec des paramètres différents. Vous pouvez appeler la méthode `lineGradientStyle()` au cours du traçage pour spécifier différents styles pour divers segments de ligne dans un tracé.

*Remarque :* Appelez `lineStyle()` avant d'appeler `lineGradientStyle()` pour tracer un trait, sinon la valeur du style de la ligne reste `undefined`.

*Remarque :* Les appels de `clear()` redéfinissent le style de trait sur `undefined`.

Vous pouvez étendre les méthodes et les gestionnaires d'événements de la classe MovieClip en créant une sous-classe.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

## Paramètres

*`fillType`*`:String` - Les valeurs valides sont `"linear"` ou `"radial"`.

*`colors`*`:Array` - Tableau de valeurs de couleurs RVB hexadécimales à utiliser pour le dégradé (par exemple, rouge correspond à 0xFF0000, bleu à 0x0000FF, etc.). Vous pouvez définir jusqu'à 15 couleurs. Pour chaque couleur, veillez à définir une valeur correspondante dans les paramètres `alphas` et `ratios`.

*`alphas`*`:Array` - Tableau de valeurs alpha pour les couleurs correspondantes dans le tableau *`colors`* ; les valeurs valides vont de 0 à 100. Si la valeur est inférieure à 0, Flash utilise 0. Si la valeur est supérieure à 100, Flash utilise 100.

*`ratios`*`:Array` - Tableau de rapports de distribution des couleurs ; les valeurs valides vont de 0 à 255. Cette valeur définit le pourcentage de la largeur où la couleur est échantillonnée à 100 %. Spécifiez une valeur pour chaque valeur dans le paramètre `colors`.

Par exemple, pour un dégradé linéaire qui comprend deux couleurs, bleu et vert, la figure suivante illustre l'emplacement des couleurs dans le dégradé selon les différentes valeurs du tableau `ratios` :

| `ratios` | **Dégradé** |
|---|---|
| `[0, 127]` |  |
| `[0, 255]` |  |
| `[127, 255]` |  |

Les valeurs du tableau doivent augmenter de manière séquentielle ; par exemple, `[0, 63, 127, 190, 255]`.

*matrix*:Object - Matrice de transformation qui est un objet comportant l'un des jeux de propriétés suivants :

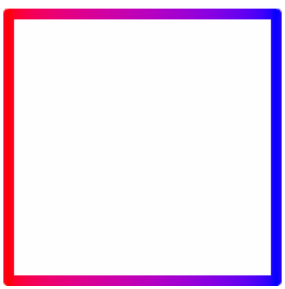
- Vous pouvez utiliser les propriétés a, b, c, d, e, f, g, h, et i pour décrire une matrice 3 x 3 de la forme suivante :

```
a b c
d e f
g h i
```

- L'exemple suivant utilise la méthode lineGradientFill() avec le paramètre matrix qui est un objet comportant les propriétés suivantes :

```
this.createEmptyMovieClip("gradient_mc", 1);
with (gradient_mc) {
  colors = [0xFF0000, 0x0000FF];
    alphas = [100, 100];
    ratios = [0, 0xFF];
    matrix = {a:200, b:0, c:0, d:0, e:200, f:0, g:200, h:200, i:1};
    spreadMethod = "reflect";
    interpolationMethod = "linearRGB";
    focalPointRatio = 0.9;
  lineStyle(8);
    lineGradientStyle("linear", colors, alphas, ratios, matrix,
    spreadMethod, interpolationMethod, focalPointRatio);
    moveTo(100, 100);
    lineTo(100, 300);
    lineTo(300, 300);
    lineTo(300, 100);
    lineTo(100, 100);
    endFill();
}
```

- Ce code dessine l'image suivante à l'écran :



- matrixType, x, y, w, h, r.

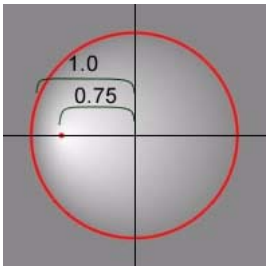
- Les propriétés ont la signification suivante : `matrixType` correspond à la chaîne « `box` », `x` désigne la position horizontale par rapport au point d'alignement du clip parent pour le coin supérieur gauche du dégradé, `y` indique la position verticale par rapport au point d'alignement du clip parent pour le coin supérieur gauche du dégradé, `w` correspond à la largeur du dégradé, `h` à sa hauteur, et `r` indique la rotation en radians du dégradé.
- L'exemple suivant utilise la méthode `lineGradientFill()` avec le paramètre `matrix` qui est un objet comportant les propriétés suivantes :

```
this.createEmptyMovieClip("gradient_mc", 1);
with (gradient_mc) {
  colors = [0xFF0000, 0x0000FF];
  alphas = [100, 100];
  ratios = [0, 0xFF];
  matrix = {matrixType:"box", x:100, y:100, w:200, h:200, r:(45/
  180)*Math.PI};
  spreadMethod = "reflect";
  interpolationMethod = "linearRGB";
  lineStyle(8);
  lineGradientStyle("linear", colors, alphas, ratios, matrix,
    spreadMethod, interpolationMethod);
  moveTo(100, 100);
  lineTo(100, 300);
  lineTo(300, 300);
  lineTo(300, 100);
  lineTo(100, 100);
  endFill();
}
```

*`spreadMethod`*`:String` [facultatif] - Les valeurs valides sont « pad », « reflect » ou « repeat », qui contrôlent le mode de remplissage en dégradé.

*`interpolationMethod`*`:String` [facultatif] - Les valeurs valides sont « RGB » ou « linearRGB ».

*focalPointRatio*:Number [facultatif] - Nombre qui contrôle l'emplacement du point focal du dégradé. La valeur 0 signifie que le point focal est au centre. La valeur 1 signifie que le point focal est au bord du cercle du dégradé. La valeur -1 signifie que le point focal est sur l'autre bord du cercle du dégradé. Les valeurs inférieures à -1 ou supérieures à 1 sont arrondies à -1 ou 1. L'image suivante présente un dégradé où la valeur de `focalPointRatio` est de -0,75 :

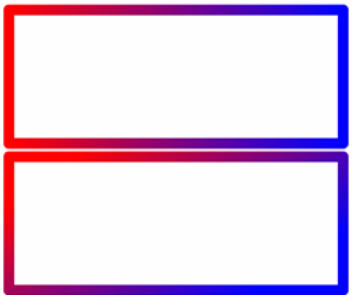



## Exemple

Le code suivant utilise les deux méthodes pour dessiner deux rectangles empilés avec un remplissage linéaire dégradé rouge-bleu :

```
this.createEmptyMovieClip("gradient_mc", 1);
with (gradient_mc) {
  colors = [0xFF0000, 0x0000FF];
  alphas = [100, 100];
  ratios = [0, 0xFF];
  matrix = {a:500, b:0, c:0, d:0, e:200, f:0, g:350, h:200, i:1};
  lineStyle(16);
  lineGradientStyle("linear", colors, alphas, ratios, matrix);
  moveTo(100, 100);
  lineTo(100, 300);
  lineTo(600, 300);
  lineTo(600, 100);
  lineTo(100, 100);
  endFill();
  matrix2 = {matrixType:"box", x:100, y:310, w:500, h:200, r:(30/
  180)*Math.PI};
  lineGradientStyle("linear", colors, alphas, ratios, matrix2);
  moveTo(100, 320);
  lineTo(100, 520);
  lineTo(600, 520);
  lineTo(600, 320);
  lineTo(100, 320);
  endFill();
}
```

Ce code dessine l'illustration ci-dessous (mise à l'échelle à 50 %) :



### Voir également

`beginGradientFill (méthode MovieClip.beginGradientFill)`, `lineStyle (méthode MovieClip.lineStyle)`, `lineTo (méthode MovieClip.lineTo)`, `moveTo (méthode MovieClip.moveTo)`

## lineStyle (méthode MovieClip.lineStyle)

```
public lineStyle(thickness:Number, rgb:Number, alpha:Number,
  pixelHinting:Boolean, noScale:String, capsStyle:String,
  jointStyle:String, miterLimit:Number) : Void
```

Spécifie un style de trait utilisé par Flash pour les prochains appels des méthodes `lineTo()` et `curveTo()` jusqu'à ce que vous appeliez la méthode `lineStyle()` avec des paramètres différents. Vous pouvez appeler `lineStyle()` au cours du traçage pour spécifier différents styles pour divers segments de ligne dans un tracé.

*Remarque :* Les appels de méthode `clear()` redéfinissent la valeur de style de trait sur `undefined`.

Vous pouvez étendre les méthodes et les gestionnaires d'événements de la classe MovieClip en créant une sous-classe.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

### Paramètres

*`thickness`*`:Number` - Entier qui indique l'épaisseur de la ligne en points ; les valeurs valides sont comprises entre 0 et 255. Si aucun nombre n'est spécifié ou si le paramètre a la valeur `undefined`, aucune ligne n'est tracée. Si vous transmettez une valeur négative, Flash Player applique 0. La valeur 0 correspond à un filet ; l'épaisseur maximum est de 255. Si vous transmettez une valeur supérieure à 255, l'interpréteur Flash applique une valeur de 255.

*rgb*:Number - Valeur hexadécimale de couleur de la ligne (par exemple, rouge correspond à 0xFF0000, bleu à 0x0000FF, etc.). En l'absence de valeur, Flash applique 0x000000 (noir).

*alpha*:Number - Entier qui indique la valeur alpha de couleur de la ligne ; les valeurs valides sont comprises entre 0 et 100. En l'absence de valeurs, Flash applique 100 (uni). Si cette valeur est négative, Flash applique 0. Si elle est supérieure à 100, Flash applique 100.

*pixelHinting*:Boolean - Ajouté dans Flash Player 8. Valeur booléenne qui permet d'ajouter des indices supplémentaires de lissage des pixels. Cette valeur affecte à la fois la position des ancres de courbe et la taille du trait. En l'absence de valeurs, Flash Player n'applique pas les indices de lissage des pixels.

*noScale*:String - Ajouté dans Flash Player 8. Chaîne qui spécifie comment redimensionner un trait. Les valeurs possibles sont :

- "normal" Redimensionne toujours l'épaisseur (valeur par défaut).
- "none" Ne redimensionne jamais l'épaisseur.
- "vertical" Ne redimensionne pas l'épaisseur si l'objet est uniquement redimensionné à la verticale.
- "horizontal" Ne redimensionne pas l'épaisseur si l'objet est uniquement redimensionné à l'horizontale.

*capsStyle*:String - Ajouté dans Flash Player 8. Chaîne qui spécifie le type d'extrémité au bout des lignes. Les valeurs possibles sont les suivantes : "round", "square", et "none". En l'absence de valeur, Flash utilise des extrémités rondes.

Par exemple, l'illustration suivante présente les différents paramètres capsStyle. Pour chaque paramètre, l'illustration présente une ligne bleue dont l'épaisseur est de 30 (pour laquelle capsStyle s'applique), et une ligne noire superposée dont l'épaisseur est de 1 (pour laquelle aucun capsStyle ne s'applique) :

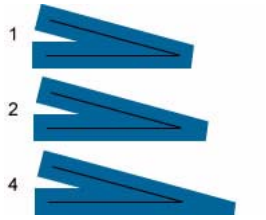

*jointStyle*:String - Ajouté dans Flash Player 8. Chaîne qui spécifie le type d'apparence de liaison utilisé dans les angles. Les valeurs possibles sont les suivantes : "round", "miter", et "bevel". En l'absence de valeurs, Flash utilise des liaisons rondes.

Par exemple, l'illustration suivante présente les différents paramètres `capsStyle`. Pour chaque paramètre, l'illustration présente une ligne bleue en angle dont l'épaisseur est de 30 (pour laquelle `jointStyle` s'applique), et une ligne noire en angle superposée dont l'épaisseur est de 1 (pour laquelle aucun `jointStyle` ne s'applique) :


Vous pouvez constater que lorsque `jointStyle` est défini sur `"miter"`, vous pouvez limiter la longueur du point de pointe en utilisant le paramètre `miterLimit`.

*`miterLimit`*`:Number` - Ajouté dans Flash Player 8. Nombre qui indique la limite à laquelle une pointe est coupée. Les valeurs possibles vont de 1 à 255 (et les valeurs hors de cette plage sont arrondies à 1 ou 255). Cette valeur n'est utilisée que si `jointStyle` est défini sur `"miter"`. Si aucune valeur n'est indiquée, Flash utilise 3. La valeur `miterLimit` représente la longueur maximale d'une pointe au-delà du point auquel les lignes se rencontrent pour former une liaison. La valeur exprime un facteur de `thickness` de la ligne. Par exemple, avec un facteur `miterLimit` de 2,5 et une valeur de `thickness` de 10 pixels, la pointe est coupée à 25 pixels.

Considérons par exemple les lignes en angle suivantes, chacune étant dessinée avec une valeur de `thickness` de 20, mais avec une valeur de `miterLimit` définie sur 1, 2, et 4. Les lignes de référence noires sont superposées pour représenter les points de rencontre des liaisons :




Notez que pour une valeur `miterLimit` donnée, il y a un angle maximum spécifique pour lequel la pointe est coupée. Vous trouverez quelques exemples dans le tableau suivant :

| `Value of miterLimit` **valeur :** | **Angles inférieurs à ceci sont coupés :** |
|---|---|
| 1.414 | 90 degrés |
| 2 | 60 degrés |

| Value of miterLimit **valeur :** | **Angles inférieurs à ceci sont coupés :** |
|---|---|
| 4 | 30 degrés |
| 8 | 15 degrés |

### Exemple

Le code suivant dessine un triangle avec une ligne unie en magenta de 5 pixels sans remplissage, avec des indices de lissage des pixels, sans redimensionnement du trait, sans extrémités et des liaisons de pointes avec un paramètre `miterLimit` de 1 :

```
this.createEmptyMovieClip("triangle_mc", this.getNextHighestDepth());
triangle_mc.lineStyle(5, 0xff00ff, 100, true, "none", "round", "miter", 1);
triangle_mc.moveTo(200, 200);
triangle_mc.lineTo(300, 300);
triangle_mc.lineTo(100, 300);
triangle_mc.lineTo(200, 200);
```

Si votre fichier SWF comprend un composant de la version 2, utilisez les composants de la version 2 de la classe DepthManager au lieu de la méthode `MovieClip.getNextHighestDepth()`, utilisée dans cet exemple.

### Voir également

`beginFill (méthode MovieClip.beginFill)`, `beginGradientFill (méthode MovieClip.beginGradientFill)`, `clear (méthode MovieClip.clear)`, `curveTo (méthode MovieClip.curveTo)`, `lineTo (méthode MovieClip.lineTo)`, `moveTo (méthode MovieClip.moveTo)`

## lineTo (méthode MovieClip.lineTo)

```
public lineTo(x:Number, y:Number) : Void
```

Trace une ligne en utilisant le style de trait actuel à partir de la position de dessin actuelle jusqu'à (x, y) ; la position de dessin actuelle est ensuite définie sur (x, y). Si le clip dans lequel vous tracez contient du contenu créé à l'aide des outils de dessin Flash, les appels de `lineTo()` sont tracés sous le contenu. Si vous appelez `lineTo()` avant d'appeler la méthode `moveTo()`, la position de dessin actuelle prend la valeur par défaut (`0`,`0`). Si l'un des paramètres est manquant, cette méthode échoue et la position de dessin actuelle n'est pas modifiée.

Vous pouvez étendre les méthodes et les gestionnaires d'événements de la classe MovieClip en créant une sous-classe.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

### Paramètres

*x*:Number - Entier indiquant la position horizontale par rapport au point d'alignement du clip parent.

*y*:Number - Entier indiquant la position verticale par rapport au point d'alignement du clip parent.

### Exemple

L'exemple suivant dessine un triangle avec une ligne en magenta de 5 pixels et un remplissage bleu partiellement transparent.

```
this.createEmptyMovieClip("triangle_mc", 1);
triangle_mc.beginFill(0x0000FF, 30);
triangle_mc.lineStyle(5, 0xFF00FF, 100);
triangle_mc.moveTo(200, 200);
triangle_mc.lineTo(300, 300);
triangle_mc.lineTo(100, 300);
triangle_mc.lineTo(200, 200);
triangle_mc.endFill();
```

### Voir également

beginFill (méthode MovieClip.beginFill), createEmptyMovieClip (méthode MovieClip.createEmptyMovieClip), endFill (méthode MovieClip.endFill), lineStyle (méthode MovieClip.lineStyle), moveTo (méthode MovieClip.moveTo)

## loadMovie (méthode MovieClip.loadMovie)

```
public loadMovie(url:String, [method:String]) : Void
```

Charge un fichier SWF, JPEG, GIF ou PNG dans un clip Flash Player lors de la lecture du fichier SWF d'origine. La prise en charge des fichiers GIF non animés, des fichiers PNG et des fichiers JPEG a été ajoutée à Flash Player 8. Si vous chargez un fichier GIF animé, seule la première image s'affiche.

*Conseil :* Pour suivre la progression du téléchargement, appliquez la méthode `MovieClipLoader.loadClip()` de préférence à la méthode `loadMovie()`.

Si la méthode `loadMovie()` n'est pas utilisée, Flash Player affiche un fichier SWF unique, puis le ferme. L'utilisation de la méthode `loadMovie()` permet d'afficher plusieurs fichiers SWF simultanément, puis de basculer entre les fichiers SWF sans charger d'autre document HTML.

Un fichier SWF ou une image chargé(e) dans un clip hérite des propriétés position, rotation et scale (échelle) du clip. Vous pouvez utiliser le chemin cible du clip pour cibler le fichier SWF chargé.

Lorsque vous appelez la méthode `loadMovie()`, définissez la propriété `MovieClip._lockroot` sur `true` dans l'animation de chargeur, comme le montre l'exemple de code suivant. Si vous ne définissez pas `_lockroot` sur `true` dans l'animation de chargeur, toute référence à `_root` dans l'animation chargée pointe vers la propriété `_root` du chargeur et non pas la propriété `_root` de l'animation chargée :

```
myMovieClip._lockroot = true;
```

Utilisez la méthode `MovieClip.unloadMovie()` pour supprimer les fichiers SWF ou les images chargés avec la méthode `loadMovie()`.

Utilisez la méthode `MovieClip.loadVariables()`, l'objet `XML`, Flash Remoting ou des objets partagés lors de l'exécution pour conserver le fichier SWF actif et y charger de nouvelles données.

L'utilisation de gestionnaires d'événement avec `MovieClip.loadMovie()` peut être imprévisible. Si vous liez un gestionnaire d'événements à un bouton avec `on()`, ou si vous créez un gestionnaire dynamique avec une méthode telle que `MovieClip.onPress()`, puis appelez `loadMovie()`, le gestionnaire d'événements ne sera plus disponible après le chargement du nouveau contenu. Cependant, si vous liez un gestionnaire d'événements à un clip avec `onClipEvent()` ou `on()`, puis que vous appelez `loadMovie()` pour ce clip, le gestionnaire d'événements reste disponible après le chargement du nouveau contenu.

Lorsque vous utilisez cette méthode, tenez compte du modèle de sécurité Flash Player.

Pour Flash Player 8 :

- Le chargement n'est pas permis si le clip appelant est dans le sandbox local avec système de fichier et que le clip chargé provient d'un sandbox réseau.
- Le chargement n'est pas autorisé si le fichier SWF appelant est sur un sandbox réseau et que le clip à charger est local.
- L'accès au sandbox réseau à partir d'un sandbox local approuvé ou de réseau local nécessite une autorisation du site au travers d'un fichier de régulation interdomaine.
- Les clips situés dans le sandbox local avec système de fichier ne peuvent pas inscrire de clips dans le sandbox local avec accès au réseau (l'inverse est également impossible).

Flash Player 7 et versions ultérieures :

- Les sites peuvent autoriser un accès interdomaine à une ressource au travers d'un fichier de régulation interdomaines.
- La programmation entre les fichiers SWF est limitée par le domaine d'origine des fichiers SWF. Utilisez la méthode `System.security.allowDomain()` pour ajuster ces restrictions.

Pour plus d'informations, voir les sections suivantes :

- Chapitre 17, « Fonctionnement de la sécurité », dans *Formation à ActionScript 2.0 dans Flash*
- Le livre blanc sur la sécurité de Flash Player 8 disponible à l'adresse : http://www.macromedia.com/go/fp8_security
- Le livre blanc sur les API relatif à la sécurité de Flash Player 8

Vous pouvez étendre les méthodes et les gestionnaires d'événements de la classe MovieClip en créant une sous-classe.

**Disponibilité :** ActionScript 1.0 ; Flash Player 5

## Paramètres

*url*:String - URL absolue ou relative du fichier SWF, JPEG, GIF ou PNG à charger. Un chemin relatif doit être relatif au fichier SWF au niveau 0. Les URL absolues doivent inclure la référence de protocole, telle que `http://` ou `file:///`.

*method*:String [facultatif] - Spécifie une méthode HTTP d'envoi ou de chargement des variables. Ce paramètre doit correspondre à la chaîne `GET` ou `POST`. En l'absence de variables à envoyer, omettez ce paramètre. La méthode `GET` ajoute les variables à la fin de l'URL et est utilisée lorsque les variables sont peu nombreuses. La méthode `POST` place les variables dans un en-tête HTTP distinct et s'applique aux variables longues de type chaîne.

## Exemple

L'exemple suivant crée un nouveau clip, puis un enfant à l'intérieur et charge une image PNG dans l'enfant. Cela permet au parent de conserver toutes les valeurs d'occurrence affectées avant l'appel de `loadMovie`.

```
var mc:MovieClip = this.createEmptyMovieClip("mc",
  this.getNextHighestDepth());
mc.onRelease = function():Void {
  trace(this.image._url); // http://www.w3.org/Icons/w3c_main.png
}
var image:MovieClip = mc.createEmptyMovieClip("image",
  mc.getNextHighestDepth());
image.loadMovie("http://www.w3.org/Icons/w3c_main.png");
```

La méthode `MovieClip.getNextHighestDepth()` utilisée dans cet exemple nécessite Flash Player 7 ou une version ultérieure. Si votre fichier SWF comprend un composant de la version 2, utilisez les composants de la version 2 de la classe DepthManager au lieu de la méthode `MovieClip.getNextHighestDepth()`.

### Voir également

`_lockroot (propriété MovieClip._lockroot)`, `unloadMovie (méthode MovieClip.unloadMovie)`, `loadVariables (méthode MovieClip.loadVariables)`, `loadMovie (méthode MovieClip.loadMovie)`, `onPress (gestionnaire MovieClip.onPress)`, `MovieClipLoader`, `Gestionnaire onClipEvent`, `Gestionnaire on`, `Fonction loadMovieNum`, `Fonction unloadMovie`, `Fonction unloadMovieNum`

## loadVariables (méthode MovieClip.loadVariables)

```
public loadVariables(url:String, [method:String]) : Void
```

Lit les données à partir d'un fichier externe et définit les valeurs des variables dans le clip. Le fichier externe peut être un fichier texte généré par Macromedia ColdFusion, un script CGI, un Active Server Page (ASP), un script PHP, ou tout autre fichier de texte correctement formaté. Le fichier peut contenir un nombre illimité de variables.

Vous pouvez également utiliser la méthode `loadVariables()` pour mettre à jour des variables dans le clip actif avec de nouvelles valeurs.

La méthode `loadVariables()` requiert que le texte de l'URL soit au format MIME standard : *application/x-www-form-urlencoded* (format de script CGI).

Pour les fichiers SWF lus par une version antérieure à Flash Player 7, l'`url` doit correspondre au superdomaine du fichier SWF envoyant cet appel. Le superdomaine est dérivé en supprimant le composant le plus à gauche de l'URL d'un fichier. Par exemple, un fichier SWF enregistré dans www.someDomain.com peut charger des données à partir d'une source figurant dans store.someDomain.com, car les deux fichiers appartiennent au même superdomaine que someDomain.com.

Pour les fichiers SWF, quelle que soit leur version, lus par Flash Player 7 ou une version plus récente, `url` doit correspondre exactement au superdomaine du fichier SWF émettant cet appel. Par exemple, un fichier SWF situé à l'adresse www.unDomaine.com peut charger des données provenant uniquement de sources situées également à l'adresse www.unDomaine.com. Pour charger des données provenant d'un domaine différent, vous pouvez placer un fichier de régulation interdomaines sur le serveur hébergeant la source de données à laquelle vous accédez.

Pour charger des variables à un niveau spécifique, utilisez `loadVariablesNum()` à la place de `loadVariables()`.

Vous pouvez étendre les méthodes et les gestionnaires d'événements de la classe MovieClip en créant une sous-classe.

**Disponibilité :** ActionScript 1.0 ; Flash Player 5

## Paramètres

*url*:String - URL absolue ou relative du fichier externe qui contient les variables à charger. Si le fichier SWF effectuant cet appel s'exécute dans un navigateur Web, l'url doit appartenir au même domaine que le fichier SWF. Pour plus de détails, consultez la section « Description » suivante.

*method*:String [facultatif] - Spécifie une méthode HTTP d'envoi des variables. Ce paramètre doit correspondre à la chaîne GET ou POST. En l'absence de variables à envoyer, omettez ce paramètre. La méthode GET ajoute les variables à la fin de l'URL et est utilisée lorsque les variables sont peu nombreuses. La méthode POST place les variables dans un en-tête HTTP distinct et s'applique aux variables longues de type chaîne.

## Exemple

L'exemple suivant charge des informations à partir d'un fichier texte appelé params.txt dans le clip target_mc créé avec createEmptyMovieClip(). Utilisez la fonction setInterval() pour contrôler la progression du chargement. Le script recherche une variable dans le fichier params.txt appelé done.

```
this.createEmptyMovieClip("target_mc", this.getNextHighestDepth());
target_mc.loadVariables("params.txt");
function checkParamsLoaded() {
  if (target_mc.done == undefined) {
  trace("not yet.");
  } else {
  trace("finished loading. killing interval.");
  trace("-------------");
  for (i in target_mc) {
    trace(i+": "+target_mc[i]);
  }
  trace("-------------");
  clearInterval(param_interval);
  }
}
var param_interval = setInterval(checkParamsLoaded, 100);
```

Le fichier params.txt, inclut le texte suivant :

```
var1="hello"&var2="goodbye"&done="done"
```

La méthode MovieClip.getNextHighestDepth() utilisée dans cet exemple nécessite Flash Player 7 ou une version ultérieure. Si votre fichier SWF comprend un composant de la version 2, utilisez les composants de la version 2 de la classe DepthManager au lieu de la méthode MovieClip.getNextHighestDepth().

### Voir également

loadMovie (méthode MovieClip.loadMovie), Fonction loadVariablesNum, unloadMovie (méthode MovieClip.unloadMovie)

## localToGlobal (méthode MovieClip.localToGlobal)

```
public localToGlobal(pt:Object) : Void
```

Convertit l'objet `pt` à partir des coordonnées du clip (locales) vers les coordonnées de la scène (globales).

La méthode `MovieClip.localToGlobal()` permet de convertir les coordonnées *x* et *y* à partir des valeurs relatives au coin supérieur gauche d'un clip donné en valeurs relatives au coin supérieur gauche de la scène.

Vous devez d'abord créer un objet générique qui a deux propriétés, `x` et `y`. Ces valeurs `x` et `y` (qui doivent être appelées `x` et `y`) sont appelées coordonnées locales dans la mesure où elles font référence au coin supérieur gauche du clip. La propriété x représente le décalage horizontal par rapport au coin supérieur gauche du clip. En d'autres termes, elle représente la position droite du point. Par exemple, si x = 50, le point est situé à 50 pixels à droite du coin supérieur gauche. La propriété y représente le décalage vertical par rapport au coin supérieur gauche du clip. En d'autres termes, elle représente la position basse du point. Par exemple, si y = 20, le point est situé à 20 pixels en dessous du coin supérieur gauche. Le code suivant crée un objet générique avec ces coordonnées.

```
var myPoint:Object = new Object();
myPoint.x = 50;
myPoint.y = 20;
```

En outre, vous pouvez créer l'objet et affecter les valeurs en même temps avec une valeur Object littérale.

```
var myPoint:Object = {x:50, y:20};
```

Après avoir créé un objet point avec des coordonnées locales, vous pouvez convertir les coordonnées en coordonnées globales. La méthode `localToGlobal()` ne renvoie pas de valeur dans la mesure où elle change les valeurs de `x` et `y` dans l'objet générique envoyé en tant que paramètre. Elle les transforme de valeurs relatives à un clip (coordonnées locales) en valeurs relatives à la scène (coordonnées globales).

Par exemple, si vous créez un clip qui est placé au point (_x:100, _y:100), puis que vous transmettez le point local représentant un point près du coin supérieur gauche du clip (x:10, y:10) à la méthode `localToGlobal()`, la méthode doit convertir les valeurs *x* et *y* en coordonnées globales, soit (x:110, y:110). Ceci est dû au fait que les coordonnées *x* et *y* sont désormais exprimées par rapport au coin supérieur gauche de la scène et non pas par rapport au coin supérieur gauche du clip.

Les coordonnées du clip ont été représentées par _x et _y, dans la mesure où il s'agit des propriétés MovieClip permettant de définir les valeurs *x* et *y* pour les MovieClips. Cependant, votre objet générique utilise x et y sans le signe souligné. Le code suivant convertit les coordonnées *x* et *y* en coordonnées globales :

```
var myPoint:Object = {x:10, y:10}; // create your generic point object
this.createEmptyMovieClip("myMovieClip", this.getNextHighestDepth());
myMovieClip._x = 100; // _x for movieclip x position
myMovieClip._y = 100; // _y for movieclip y position

myMovieClip.localToGlobal(myPoint);
trace ("x: " + myPoint.x); // output: 110
trace ("y: " + myPoint.y); // output: 110
```

Vous pouvez étendre les méthodes et les gestionnaires d'événements de la classe MovieClip en créant une sous-classe.

**Disponibilité :** ActionScript 1.0 ; Flash Player 5

## Paramètres

*pt*:Object - Nom ou identifiant d'un objet créé avec la classe Object, spécifiant les coordonnées *x* et *y* en tant que propriétés.

## Exemple

L'exemple suivant convertit les coordonnées *x* et *y* de l'objet my_mc à partir des coordonnées du clip (locales) et vers les coordonnées de la scène (globales). Le point central du clip est déplacé lorsque vous cliquez sur une occurrence et la faites glisser.

```
this.createTextField("point_txt", this.getNextHighestDepth(), 0, 0, 100,
  22);
var mouseListener:Object = new Object();
mouseListener.onMouseMove = function() {
 var point:Object = {x:my_mc._width/2, y:my_mc._height/2};
 my_mc.localToGlobal(point);
 point_txt.text = "x:"+point.x+", y:"+point.y;
};
Mouse.addListener(mouseListener);
my_mc.onPress = function() {
 this.startDrag();
};
my_mc.onRelease = function() {
 this.stopDrag();
};
```

La méthode MovieClip.getNextHighestDepth() utilisée dans cet exemple nécessite Flash Player 7 ou une version ultérieure. Si votre fichier SWF comprend un composant de la version 2, utilisez les composants de la version 2 de la classe DepthManager au lieu de la méthode MovieClip.getNextHighestDepth().

### Voir également

globalToLocal (méthode MovieClip.globalToLocal)

## _lockroot (propriété MovieClip._lockroot)

```
public _lockroot : Boolean
```

Une valeur booléenne qui spécifie ce à quoi _root se réfère lorsqu'un fichier SWF est chargé dans un clip. La propriété _lockroot est undefined par défaut. Vous pouvez définir cette propriété dans le fichier SWF en cours de chargement ou dans le gestionnaire qui charge le clip.

Par exemple, supposons que vous ayez un document appelé Games.fla permettant à un utilisateur de sélectionner un jeu, puis de le charger (par exemple, Chess.swf) dans le clip game_mc. Veiller à ce que lorsqu'il est chargé dans Games.swf, toute utilisation de _root dans Chess.swf fasse référence à _root dans Chess.swf (pas _root dans Games.swf). Si vous avez accès à Chess.fla et le publiez dans Flash Player 7 ou une version ultérieure, vous pouvez ajouter cette instruction à Chess.fla sur le scénario principal :

```
this._lockroot = true;
```

Si vous n'avez pas accès à Chess.fla (par exemple, si vous chargez Chess.swf à partir du site d'un autre utilisateur dans chess_mc), vous pouvez définir la propriété _lockroot de Chess.swf, lorsque vous le chargez. Placez le code ActionScript suivant sur le scénario principal de Games.fla :

```
chess_mc._lockroot = true;
```

Dans ce cas, Chess.swf peut être publié pour n'importe quelle version de Flash Player, dans la mesure où Games.swf est publié pour Flash Player 7 ou une version ultérieure.

Lorsque vous appelez la méthode loadMovie(), définissez la propriété MovieClip._lockroot sur true dans l'animation de chargeur, comme le montre l'exemple de code suivant. Si vous ne définissez pas _lockroot sur true dans l'animation de chargeur, toute référence à _root dans l'animation chargée pointe vers la propriété _root du chargeur et non pas la propriété _root de l'animation chargée :

```
myMovieClip._lockroot = true;
```

**Disponibilité :** ActionScript 1.0 ; Flash Player 7

### Exemple

Dans l'exemple suivant, lockroot.fla reçoit la propriété _lockroot qui est appliquée au fichier SWF principal. Si le fichier SWF est chargé dans un autre document FLA, la propriété _root fait toujours référence au domaine de lockroot.swf, ce qui permet d'éviter les conflits. Placez le code ActionScript suivant sur le scénario principal de lockroot.fla :

```
this._lockroot = true;
_root.myVar = 1;
_root.myOtherVar = 2;
trace("from lockroot.swf");
for (i in _root) {
  trace(" "+i+" -> "+_root[i]);
}
trace("");
```

ce qui permet de suivre les informations suivantes :

```
from lockroot.swf
myOtherVar -> 2
myVar -> 1
_lockroot -> true
$version -> WIN 7,0,19,0
```

L'exemple suivant charge deux fichiers SWF, lockroot.swf et nolockroot.swf. Le document lockroot.fla contient le code ActionScript de l'exemple précédent. Le fichier nolockroot.fla ajoute le code suivant à l'image 1 du scénario :

```
_root.myVar = 1;
_root.myOtherVar = 2;
trace("from nolockroot.swf");
for (i in _root) {
  trace(" "+i+" -> "+_root[i]);
}
trace("");
```

La propriété `_lockroot` est appliquée au fichier lockroot.swf, mais pas à nolockroot.swf. Une fois les fichiers chargés, chaque fichier fait sortir les variables de son domaine `_root`. Placez le code ActionScript suivant sur le scénario principal d'un document FLA :

```
this.createEmptyMovieClip("lockroot_mc", this.getNextHighestDepth());
lockroot_mc.loadMovie("lockroot.swf");
this.createEmptyMovieClip("nolockroot_mc", this.getNextHighestDepth());
nolockroot_mc.loadMovie("nolockroot.swf");
function dumpRoot() {
  trace("from current SWF file");
  for (i in _root) {
  trace(" "+i+" -> "+_root[i]);
  }
  trace("");
}
dumpRoot();
```

ce qui permet de suivre les informations suivantes :

```
 from current SWF file
 dumpRoot -> [type Function]
 $version -> WIN 7,0,19,0
 nolockroot_mc -> _level0.nolockroot_mc
 lockroot_mc -> _level0.lockroot_mc
```

```
from nolockroot.swf
myVar -> 1
i -> lockroot_mc
dumpRoot -> [type Function]
$version -> WIN 7,0,19,0
nolockroot_mc -> _level0.nolockroot_mc
lockroot_mc -> _level0.lockroot_mc

from lockroot.swf
myOtherVar -> 2
myVar -> 1
```

Le fichier auquel `_lockroot` n'a pas été appliqué contient également toutes les autres variables contenues dans le fichier SWF racine. Si vous n'avez pas accès au fichier nolockroot.fla, vous pouvez utiliser le code ActionScript suivant, qui a été ajouté au scénario principal, pour modifier la propriété `_lockroot` dans le document FLA principal précédent :

```
this.createEmptyMovieClip("nolockroot_mc", this.getNextHighestDepth());
nolockroot_mc._lockroot = true;
nolockroot_mc.loadMovie("nolockroot.swf");
```

qui suit alors les éléments suivants :

```
from current SWF file
dumpRoot -> [type Function]
$version -> WIN 7,0,19,0
nolockroot_mc -> _level0.nolockroot_mc
lockroot_mc -> _level0.lockroot_mc

from nolockroot.swf
myOtherVar -> 2
myVar -> 1

from lockroot.swf
myOtherVar -> 2
myVar -> 1
```

## Voir également

`_root, propriété`, `_lockroot (propriété MovieClip._lockroot)`, `attachMovie (méthode MovieClip.attachMovie)`, `loadMovie (méthode MovieClip.loadMovie)`, `onLoadInit (écouteur d'événement MovieClipLoader.onLoadInit)`

# menu (propriété MovieClip.menu)

```
public menu : ContextMenu
```

Associe l'objet ContextMenu spécifié au clip. La classe ContextMenu permet de modifier le menu contextuel qui s'affiche lorsque l'utilisateur clique avec le bouton droit de la souris (Windows) ou appuie sur la touche Contrôle (Macintosh) dans Flash Player.

**Disponibilité :** ActionScript 1.0 ; Flash Player 7

## Exemple

L'exemple suivant associe l'objet ContextMenu `menu_cm` au clip `image_mc`. L'objet ContextMenu contient un élément de menu personnalisé appelé « View Image in Browser » qui est associé à une fonction appelée `viewImage()`:

```
var menu_cm:ContextMenu = new ContextMenu();
menu_cm.customItems.push(new ContextMenuItem("View Image in Browser...",
  viewImage));
this.createEmptyMovieClip("image_mc", this.getNextHighestDepth());
var mclListener:Object = new Object();
mclListener.onLoadInit = function(target_mc:MovieClip) {
  target_mc.menu = menu_cm;
};
var image_mcl:MovieClipLoader = new MovieClipLoader();
image_mcl.addListener(mclListener);
image_mcl.loadClip("photo1.jpg", image_mc);

function viewImage(target_mc:MovieClip, obj:Object) {
  getURL(target_mc._url, "_blank");
}
```

Lorsque vous cliquez avec le bouton droit de la souris (Windows) ou en appuyant sur la touche Contrôle (Macintosh) sur l'image pendant l'exécution, sélectionnez View Image in Browser dans le menu contextuel pour ouvrir l'image dans une fenêtre de navigateur.

## Voir également

`menu (propriété Button.menu)`, `ContextMenu`, `ContextMenuItem`, `menu (propriété TextField.menu)`

# moveTo (méthode MovieClip.moveTo)

```
public moveTo(x:Number, y:Number) : Void
```

Déplace la position de dessin actuelle vers (x, y). Si l'un des paramètres est manquant, cette méthode échoue et la position de dessin actuelle n'est pas modifiée.

Vous pouvez étendre les méthodes et les gestionnaires d'événements de la classe MovieClip en créant une sous-classe.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

### Paramètres

*x*:Number - Entier indiquant la position horizontale par rapport au point d'alignement du clip parent.

*y*:Number - Entier indiquant la position verticale par rapport au point d'alignement du clip parent.

### Exemple

L'exemple suivant dessine un triangle avec une ligne en magenta de 5 pixels et un remplissage bleu partiellement transparent.

```
this.createEmptyMovieClip("triangle_mc", 1);
triangle_mc.beginFill(0x0000FF, 30);
triangle_mc.lineStyle(5, 0xFF00FF, 100);
triangle_mc.moveTo(200, 200);
triangle_mc.lineTo(300, 300);
triangle_mc.lineTo(100, 300);
triangle_mc.lineTo(200, 200);
triangle_mc.endFill();
```

### Voir également

createEmptyMovieClip (méthode MovieClip.createEmptyMovieClip), lineStyle (méthode MovieClip.lineStyle), lineTo (méthode MovieClip.lineTo)

## _name (propriété MovieClip._name)

```
public _name : String
```

Le nom d'occurrence du clip.

**Disponibilité :** ActionScript 1.0 ; Flash Player 4

### Exemple

L'exemple suivant permet de cliquer avec le bouton droit de la souris (Windows) ou en appuyant sur la touche contrôle (Macintosh) sur un clip figurant dans la scène et de sélectionner `Info` dans le menu contextuel pour afficher des informations sur cette occurrence. Ajoutez plusieurs clips avec des noms d'occurrence, puis ajoutez le code ActionScript suivant au fichier AS ou FLA :

```
var menu_cm:ContextMenu = new ContextMenu();
```

```
menu_cm.customItems.push(new ContextMenuItem("Info...", getMCInfo));
function getMCInfo(target_mc:MovieClip, obj:Object) {
  trace("You clicked on the movie clip '"+target_mc._name+"'.");
  trace("\t width:"+target_mc._width+", height:"+target_mc._height);
  trace("");
}
for (var i in this) {
  if (typeof (this[i]) == 'movieclip') {
  this[i].menu = menu_cm;
  }
}
```

### Voir également

_name (propriété Button._name)

## nextFrame (méthode MovieClip.nextFrame)

```
public nextFrame() : Void
```

Place la tête de lecture sur l'image suivante et l'arrête.

Vous pouvez étendre les méthodes et les gestionnaires d'événements de la classe MovieClip en créant une sous-classe.

**Disponibilité :** ActionScript 1.0 ; Flash Player 5

### Exemple

L'exemple suivant utilise `_framesloaded` et `nextFrame()` pour charger le contenu dans un fichier SWF. N'ajoutez pas de code sur l'image 1. Par contre, ajoutez le code ActionScript suivant sur l'image 2 du scénario :

```
if (this._framesloaded >= 3) {
  this.nextFrame();
} else {
  this.gotoAndPlay(1);
}
```

Ensuite, ajoutez le code suivant (et le contenu à charger) dans l'image 3 :

```
stop();
```

### Voir également

Fonction nextFrame, Fonction prevFrame, prevFrame (méthode MovieClip.prevFrame)

# onData (gestionnaire MovieClip.onData)

```
onData = function() {}
```

Invoqué lorsqu'un clip reçoit des données d'un appel `MovieClip.loadVariables()` ou un appel `MovieClip.loadMovie()`. Vous devez définir une fonction qui s'exécute lorsque le gestionnaire d'événements est appelé. Vous pouvez définir la fonction sur le scénario ou dans un fichier de classe qui étend la classe MovieClip et est lié à un symbole dans la bibliothèque.

Vous pouvez utiliser ce gestionnaire uniquement avec la méthode `MovieClip.loadVariables()` ou la fonction globale `loadVariables()`. Si vous voulez invoquer un gestionnaire d'événements avec la méthode `MovieClip.loadMovie()` ou la fonction `loadMovie()`, vous devez utiliser `onClipEvent(data)` à la place de ce gestionnaire.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

## Exemple

L'exemple suivant illustre l'utilisation correcte de `MovieClip.onData()`. Il charge un fichier appelé *OnData.txt* à partir du répertoire contenant le fichier FLA. Lorsque les données du fichier sont chargées dans l'objet `MovieClip`, `onData()` s'exécute et nous suivons les données.

```
var mc:MovieClip = this.createEmptyMovieClip("mc",
  this.getNextHighestDepth());

mc.onData = function() {
  for(var i in this) {
    trace(">> " + i + ": " + this[i]);
  }
}

mc.loadVariables("OnData.txt");
```

## Voir également

`Gestionnaire onClipEvent`, `loadVariables (méthode MovieClip.loadVariables)`

# onDragOut (gestionnaire MovieClip.onDragOut)

```
onDragOut = function() {}
```

Appelé lorsque l'utilisateur appuie sur le bouton de la souris et que le pointeur se déplace hors de l'objet. Vous devez définir une fonction qui s'exécute lorsque le gestionnaire d'événements est appelé. Vous pouvez définir la fonction sur le scénario ou dans un fichier de classe qui étend la classe MovieClip ou est lié à un symbole dans la bibliothèque.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

### Exemple

L'exemple suivant définit une fonction pour la méthode `onDragOut` qui transmet une instruction `trace()` au panneau de sortie.

```
my_mc.onDragOut = function () {
  trace ("onDragOut called");
}
```

### Voir également

`onDragOver (gestionnaire MovieClip.onDragOver)`

## onDragOver (gestionnaire MovieClip.onDragOver)

```
onDragOver = function() {}
```

Appelé lorsque l'utilisateur fait glisser le pointeur hors du clip, puis sur le clip. Vous devez définir une fonction qui s'exécute lorsque le gestionnaire d'événements est appelé. Vous pouvez définir la fonction sur le scénario ou dans un fichier de classe qui étend la classe MovieClip ou est lié à un symbole dans la bibliothèque.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

### Exemple

L'exemple suivant définit une fonction pour la méthode `onDragOver` qui transmet une instruction `trace()` au panneau de sortie.

```
my_mc.onDragOver = function () {
  trace ("onDragOver called");
}
```

### Voir également

`onDragOut (gestionnaire MovieClip.onDragOut)`

## onEnterFrame (gestionnaire MovieClip.onEnterFrame)

```
onEnterFrame = function() {}
```

Appelé à plusieurs reprises à la cadence du fichier SWF. La fonction que vous affectez au gestionnaire d'événement `onEnterFrame` est traitée avant tout autre code ActionScript lié aux images affectées.

Vous devez définir une fonction qui s'exécute lorsque le gestionnaire d'événements est appelé. Vous pouvez définir la fonction sur le scénario ou dans un fichier de classe qui étend la classe MovieClip ou est lié à un symbole dans la bibliothèque.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

### Exemple

L'exemple suivant définit une fonction pour le gestionnaire d'événement `onEnterFrame` qui transmet une instruction `trace()` au panneau de sortie.

```
my_mc.onEnterFrame = function () {
  trace ("onEnterFrame called");
}
```

## onKeyDown (gestionnaire MovieClip.onKeyDown)

```
onKeyDown = function() {}
```

Appelé lorsqu'un clip reçoit le focus d'entrée et que l'utilisateur appuie sur une touche. Le gestionnaire d'événements `onKeyDown` est appelé sans paramètre. Vous pouvez utiliser les méthodes `Key.getAscii()` et `Key.getCode()` pour déterminer sur quelle touche l'utilisateur a appuyé. Vous devez définir une fonction qui s'exécute lorsque le gestionnaire d'événements est appelé. Vous pouvez définir la fonction sur le scénario ou dans un fichier de classe qui étend la classe MovieClip ou est lié à un symbole dans la bibliothèque.

Le gestionnaire d'événements `onKeyDown` fonctionne uniquement si le focus d'entrée du clip est activé et défini. D'abord, la propriété `MovieClip.focusEnabled` doit être définie sur `true` pour le clip. Ensuite, le clip doit recevoir le focus. Pour ce faire, utilisez `Selection.setFocus()` ou paramétrez la touche Tab pour naviguer jusqu'au clip.

Si vous utilisez `Selection.setFocus()`, vous devez transmettre le chemin pour le clip à `Selection.setFocus()`. Les autres éléments peuvent aisément reprendre le focus lorsqu'un utilisateur déplace la souris.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

### Exemple

L'exemple suivant définit une fonction pour la méthode `onKeyDown()` qui transmet une instruction `trace()` au panneau de sortie. Crée un clip appelé my_mc et ajoute le code ActionScript suivant au fichier FLA ou AS :

```
my_mc.onKeyDown = function () {
  trace ("key was pressed");
}
```

Le clip doit avoir le focus pour que le gestionnaire d'événements `onKeyDown` fonctionne. Ajoutez le code ActionScript pour définir le focus d'entrée :

```
my_mc.tabEnabled = true;
my_mc.focusEnabled = true;
Selection.setFocus(my_mc);
```

Lorsque l'utilisateur appuie sur une touche, `key was pressed` s'affiche dans le panneau de sortie. Cependant, cette situation ne se produit pas lorsque vous déplacez la souris, dans la mesure où le clip perd le focus. Par conséquent, vous devez utiliser `Key.onKeyDown` dans la plupart des cas.

### Voir également

`getAscii (méthode Key.getAscii)`, `getCode (méthode Key.getCode)`, `onKeyDown (écouteur d'événement Key.onKeyDown)`, `focusEnabled (propriété MovieClip.focusEnabled)`, `onKeyUp (gestionnaire MovieClip.onKeyUp)`, `setFocus (méthode Selection.setFocus)`

## onKeyUp (gestionnaire MovieClip.onKeyUp)

```
onKeyUp = function() {}
```

Appelé lorsqu'une touche est relâchée. Le gestionnaire d'événements `onKeyUp` est appelé sans paramètre. Vous pouvez utiliser les méthodes `Key.getAscii()` et `Key.getCode()` pour déterminer quelle touche a été utilisée. Vous devez définir une fonction qui s'exécute lorsque le gestionnaire d'événements est appelé. Vous pouvez définir la fonction sur le scénario ou dans un fichier de classe qui étend la classe MovieClip ou est lié à un symbole dans la bibliothèque.

Le gestionnaire d'événements `onKeyUp` fonctionne uniquement si le focus d'entrée du clip est activé et défini. D'abord, la propriété `MovieClip.focusEnabled` doit être définie sur `true` pour le clip. Ensuite, le clip doit recevoir le focus. Pour ce faire, utilisez `Selection.setFocus()` ou paramétrez la touche Tab pour naviguer jusqu'au clip.

Si vous utilisez `Selection.setFocus()`, vous devez transmettre le chemin pour le clip à `Selection.setFocus()`. Les autres éléments peuvent aisément reprendre le focus lorsque l'utilisateur déplace la souris.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

### Exemple

L'exemple suivant définit une fonction pour la méthode `onKeyUp` qui transmet une instruction `trace()` au panneau de sortie.

```
my_mc.onKeyUp = function () {
```

```
  trace ("onKey called");
}
```

L'exemple suivant définit le focus d'entrée :

```
my_mc.focusEnabled = true;
Selection.setFocus(my_mc);
```

### Voir également

`getAscii (méthode Key.getAscii)`, `getCode (méthode Key.getCode)`, `onKeyDown (écouteur d'événement Key.onKeyDown)`, `focusEnabled (propriété MovieClip.focusEnabled)`, `onKeyDown (gestionnaire MovieClip.onKeyDown)`, `setFocus (méthode Selection.setFocus)`

## onKillFocus (gestionnaire MovieClip.onKillFocus)

```
onKillFocus = function(newFocus:Object) {}
```

Appelé lorsqu'un clip perd le focus clavier. La méthode `onKillFocus` reçoit un paramètre, `newFocus` : il s'agit d'un objet qui représente le nouvel objet recevant le focus. Si aucun objet ne reçoit le focus, `newFocus` contient la valeur `null`.

Vous devez définir une fonction qui s'exécute lorsque le gestionnaire d'événements est appelé. Vous pouvez définir la fonction sur le scénario ou dans un fichier de classe qui étend la classe MovieClip ou est lié à un symbole dans la bibliothèque.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

### Paramètres

*`newFocus`*`:Object` - Objet qui reçoit le focus clavier.

### Exemple

L'exemple suivant signale des informations sur le clip qui perd le focus et l'occurrence qui le reçoit. Deux clips, appelés `my_mc` et `other_mc`, sont sur la scène. Ajoutez le code ActionScript suivant à votre document AS ou FLA :

```
my_mc.onRelease = Void;
other_mc.onRelease = Void;
my_mc.onKillFocus = function(newFocus) {
  trace("onKillFocus called, new focus is: "+newFocus);
};
```

Sélectionnez les deux occurrences avec la touche de tabulation pour afficher des informations dans le panneau de sortie.

### Voir également

onSetFocus (gestionnaire MovieClip.onSetFocus)

## onLoad (gestionnaire MovieClip.onLoad)

```
onLoad = function() {}
```

Appelé lorsque le clip est instancié et apparaît dans le scénario. Vous devez définir une fonction qui s'exécute lorsque le gestionnaire d'événements est appelé. Vous pouvez définir la fonction sur le scénario ou dans un fichier de classe qui étend la classe MovieClip ou est lié à un symbole dans la bibliothèque.

Vous utilisez ce gestionnaire uniquement avec des clips disposant d'un symbole associé à une classe dans la bibliothèque. Si vous souhaitez qu'un gestionnaire d'événements soit appelé lors du chargement d'un clip spécifique, vous devez utiliser `onClipEvent(load)` ou la classe MovieClipLoader à la place de ce gestionnaire ; par exemple, lorsque vous utilisez `MovieClip.loadMovie()` pour charger un fichier SWF de manière dynamique. Contrairement à `MovieClip.onLoad`, les autres gestionnaires sont appelés lors du chargement d'un clip.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

### Exemple

Cet exemple indique comment utiliser le gestionnaire d'événements `onLoad` dans une définition de classe ActionScript 2.0 qui étend la classe MovieClip. Tout d'abord, créez un fichier de classe appelé Oval.as et définissez une méthode de classe appelée `onLoad()`. Assurez-vous ensuite que le fichier de classe figure dans le chemin de classe approprié, comme dans l'exemple suivant :

```
// contents of Oval.as
class Oval extends MovieClip{
  public function onLoad () {
    trace ("onLoad called");
  }
}
```

Ensuite, créez un symbole de clip dans votre bibliothèque et appelez-le Oval. Cliquez avec le bouton droit de la souris (pour afficher le menu contextuel) sur le symbole dans le panneau Bibliothèque et sélectionnez Liaison... dans le menu contextuel. Cliquez sur l'option Exporter pour ActionScript et saisissez `Oval` dans l'identifiant et les champs de classe ActionScript 2.0. Maintenez l'option Exporter dans la première image activée, puis cliquez sur OK.

En troisième lieu, passez à la première image de votre fichier et entrez le code suivant dans le panneau Actions :

```
var myOval:Oval = Oval(attachMovie("Oval","Oval_1",1));
```

Enfin, créez une animation de test, ce qui renvoie normalement le texte « onLoad appelé ».

### Voir également

`loadMovie (méthode MovieClip.loadMovie)`, `Gestionnaire onClipEvent`, `MovieClipLoader`

## onMouseDown (gestionnaire MovieClip.onMouseDown)

```
onMouseDown = function() {}
```

Appelé lorsque vous appuyez sur le bouton de la souris. Vous devez définir une fonction qui s'exécute lorsque le gestionnaire d'événements est appelé. Vous pouvez définir la fonction sur le scénario ou dans un fichier de classe qui étend la classe MovieClip ou est lié à un symbole dans la bibliothèque.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

### Exemple

L'exemple suivant définit une fonction pour la méthode `onMouseDown()` qui transmet une instruction `trace()` au panneau de sortie.

```
my_mc.onMouseDown = function () {
  trace ("onMouseDown called");
}
```

## onMouseMove (gestionnaire MovieClip.onMouseMove)

```
onMouseMove = function() {}
```

Appelé lorsque la souris bouge. Vous devez définir une fonction qui s'exécute lorsque le gestionnaire d'événements est appelé. Vous pouvez définir la fonction sur le scénario ou dans un fichier de classe qui étend la classe MovieClip ou est lié à un symbole dans la bibliothèque.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

### Exemple

L'exemple suivant définit une fonction pour la méthode `onMouseMove()` qui transmet une instruction `trace()` au panneau de sortie.

```
my_mc.onMouseMove = function () {
  trace ("onMouseMove called");
}
```

## onMouseUp (gestionnaire MovieClip.onMouseUp)

```
onMouseUp = function() {}
```

Appelé lorsque vous relâchez le bouton de la souris. Vous devez définir une fonction qui s'exécute lorsque le gestionnaire d'événements est appelé. Vous pouvez définir la fonction sur le scénario ou dans un fichier de classe qui étend la classe MovieClip ou est lié à un symbole dans la bibliothèque.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

### Exemple

L'exemple suivant définit une fonction pour la méthode `onMouseUp()` qui transmet une instruction `trace()` au panneau de sortie.

```
my_mc.onMouseUp = function () {
  trace ("onMouseUp called");
}
```

## onPress (gestionnaire MovieClip.onPress)

```
onPress = function() {}
```

Appelé lorsque l'utilisateur clique sur le bouton de la souris quand le pointeur est placé sur un clip. Vous devez définir une fonction qui s'exécute lorsque le gestionnaire d'événements est appelé. Vous pouvez définir la fonction dans la bibliothèque.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

### Exemple

L'exemple suivant définit une fonction pour la méthode `onPress()` qui transmet une instruction `trace()` au panneau de sortie.

```
my_mc.onPress = function () {
  trace ("onPress called");
}
```

## onRelease (gestionnaire MovieClip.onRelease)

```
onRelease = function() {}
```

Appelé lorsqu'un utilisateur relâche le bouton de la souris sur un clip. Vous devez définir une fonction qui s'exécute lorsque le gestionnaire d'événements est appelé. Vous pouvez définir la fonction sur le scénario ou dans un fichier de classe qui étend la classe MovieClip ou est lié à un symbole dans la bibliothèque.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

### Exemple

L'exemple suivant définit une fonction pour la méthode `onRelease()` qui transmet une instruction `trace()` au panneau de sortie.

```
my_mc.onRelease = function () {
  trace ("onRelease called");
}
```

## onReleaseOutside (gestionnaire MovieClip.onReleaseOutside)

```
onReleaseOutside = function() {}
```

Appelé lorsqu'un utilisateur appuie sur le bouton de la souris dans une zone de clip et le relâche ensuite en dehors de la zone de clip.

Vous devez définir une fonction qui s'exécute lorsque le gestionnaire d'événements est appelé. Vous pouvez définir la fonction sur le scénario ou dans un fichier de classe qui étend la classe MovieClip ou est lié à un symbole dans la bibliothèque.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

### Exemple

L'exemple suivant définit une fonction pour la méthode `onReleaseOutside()` qui transmet une instruction `trace()` au panneau de sortie.

```
my_mc.onReleaseOutside = function () {
  trace ("onReleaseOutside called");
}
```

## onRollOut (gestionnaire MovieClip.onRollOut)

```
onRollOut = function() {}
```

Appelé lorsqu'un utilisateur déplace le pointeur hors d'une zone de clip.

Vous devez définir une fonction qui s'exécute lorsque le gestionnaire d'événements est appelé. Vous pouvez définir la fonction sur le scénario ou dans un fichier de classe qui étend la classe MovieClip ou est lié à un symbole dans la bibliothèque.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

### Exemple

L'exemple suivant définit une fonction pour la méthode `onRollOut()` qui transmet une instruction `trace()` au panneau de sortie.

```
my_mc.onRollOut = function () {
```

```
  trace ("onRollOut called");
}
```

# onRollOver (gestionnaire MovieClip.onRollOver)

```
onRollOver = function() {}
```

Appelé lorsqu'un utilisateur déplace le pointeur sur une zone de clip.

Vous devez définir une fonction qui s'exécute lorsque le gestionnaire d'événements est appelé. Vous pouvez définir la fonction sur le scénario ou dans un fichier de classe qui étend la classe MovieClip ou est lié à un symbole dans la bibliothèque.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

## Exemple

L'exemple suivant définit une fonction pour la méthode `onRollOver()` qui transmet une instruction `trace()` au panneau de sortie.

```
my_mc.onRollOver = function () {
  trace ("onRollOver called");
}
```

# onSetFocus (gestionnaire MovieClip.onSetFocus)

```
onSetFocus = function(oldFocus:Object) {}
```

Appelé lorsqu'un clip reçoit le focus clavier. Le paramètre `oldFocus` est l'objet qui perd le focus. Par exemple, si l'utilisateur appuie sur la touche Tab pour déplacer le focus d'entrée d'un clip vers un champ de texte, `oldFocus` contient l'occurrence de clip.

Si aucun objet n'avait précédemment reçu le focus, le paramètre `oldFocus` contient une valeur nulle.

Vous devez définir une fonction qui s'exécute lorsque le gestionnaire d'événements est appelé. Vous pouvez définir la fonction sur le scénario ou dans un fichier de classe qui étend la classe MovieClip ou est lié à un symbole dans la bibliothèque.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

## Paramètres

*`oldFocus`*`:Object` - Objet qui perd le focus.

### Exemple

L'exemple suivant affiche des informations sur le clip qui reçoit le focus clavier et l'occurrence qui vient de le céder. Deux clips, appelés `my_mc` et `other_mc`, sont sur la scène. Ajoutez le code ActionScript suivant à votre document AS ou FLA :

```
my_mc.onRelease = Void;
other_mc.onRelease = Void;
my_mc.onSetFocus = function(oldFocus) {
  trace("onSetFocus called, previous focus was: "+oldFocus);
}
```

Sélectionnez les deux occurrences avec la touche de tabulation pour afficher des informations dans le panneau de sortie.

### Voir également

`onKillFocus (gestionnaire MovieClip.onKillFocus)`

## onUnload (gestionnaire MovieClip.onUnload)

```
onUnload = function() {}
```

Appelé dans la première image une fois la suppression du clip dans le scénario effectuée. Flash exécute les actions associées au gestionnaire d'événements `onUnload` avant de lier des actions à l'image affectée. Vous devez définir une fonction qui s'exécute lorsque le gestionnaire d'événements est appelé. Vous pouvez définir la fonction sur le scénario ou dans un fichier de classe qui étend la classe MovieClip ou lié à un symbole dans la bibliothèque.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

### Exemple

L'exemple suivant définit une fonction pour la méthode `MovieClip.onUnload()` qui transmet une instruction `trace()` au panneau de sortie.

```
my_mc.onUnload = function () {
  trace ("onUnload called");
}
```

# opaqueBackground (propriété MovieClip.opaqueBackground)

```
public opaqueBackground : Number
```

Couleur de l'arrière-plan opaque (non transparent) du, clip spécifiée par un nombre (une valeur RVB hexadécimale). Lorsque la valeur est `null` ou `undefined`, il n'y a aucun arrière-plan opaque. Pour les clips dont la propriété `cacheAsBitmap` est définie sur `true`, le paramètre `opaqueBackground` peut améliorer les performances de rendu.

L'amélioration des performances est particulièrement visible pour les clips qui comporteraient de nombreuses régions transparentes si `opaqueBackground` n'avaient pas été défini.

*Remarque :*La zone d'arrière-plan opaque ne fait l'objet d'aucune concordance lorsque la méthode `hitTest()` a un paramètre `shapeFlag` défini sur `true`.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

## Exemple

L'exemple suivant crée un contour triangulaire et défini la propriété `opaqueBackground` sur une couleur spécifique :

```
var triangle:MovieClip = this.createEmptyMovieClip("triangle",
  this.getNextHighestDepth());
triangle._x = triangle._y = 50;
triangle.lineStyle(3, 0xFFCC00);
triangle.lineTo(0, 30);
triangle.lineTo(50, 0);
triangle.lineTo(0, 0);
triangle.endFill();
triangle.opaqueBackground = 0xCCCCCC;
```

Si votre fichier SWF comprend un composant de la version 2, utilisez les composants de la version 2 de la classe DepthManager au lieu de la méthode `MovieClip.getNextHighestDepth()`, utilisée dans cet exemple.

## Voir également

`cacheAsBitmap (propriété MovieClip.cacheAsBitmap)`, `hitTest (méthode MovieClip.hitTest)`

# _parent (propriété MovieClip._parent)

```
public _parent : MovieClip
```

Référence au clip ou à l'objet contenant le clip ou l'objet actuel. L'objet actuel est celui qui fait référence à la propriété `_parent`. Utilisez la propriété `_parent` pour spécifier un chemin relatif vers les clips ou les objets situés au-dessus du clip ou de l'objet actuel.

Vous pouvez utiliser `_parent` pour remonter de plusieurs niveaux dans l'arborescence de la liste d'affichage, comme dans l'exemple suivant :

```
this._parent._parent._alpha = 20;
```

**Disponibilité :** ActionScript 1.0 ; Flash Player 5

## Exemple

L'exemple ci-dessous suit la référence d'un clip et de son scénario parent. Crée un clip avec le nom d'occurrence `my_mc`, et l'ajoute au scénario principal. Ajoutez le code ActionScript suivant à votre fichier FLA ou AS :

```
my_mc.onRelease = function() {
  trace("You clicked the movie clip: "+this);
  trace("The parent of "+this._name+" is: "+this._parent);
}
```

Lorsque vous cliquez sur le clip, les informations suivantes s'affichent dans le panneau de sortie :

```
You clicked the movie clip: _level0.my_mc
The parent of my_mc is: _level0
```

## Voir également

`_parent (propriété Button._parent)`, `_root, propriété`, `Fonction targetPath`, `_parent (propriété TextField._parent)`

# play (méthode MovieClip.play)

```
public play() : Void
```

Déplace la tête de lecture dans le scénario du clip.

Vous pouvez étendre les méthodes et les gestionnaires d'événements de la classe MovieClip en créant une sous-classe.

**Disponibilité :** ActionScript 1.0 ; Flash Player 5

### Exemple

Utilisez le code ActionScript suivant pour lire le scénario principal d'un fichier SWF. Le code ActionScript est destiné à un bouton de clip appelé `my_mc` dans le scénario principal :

```
stop();
my_mc.onRelease = function() {
  this._parent.play();
};
```

Utilisez le code ActionScript suivant pour lire le scénario d'un clip dans un fichier SWF. Le code ActionScript suivant est destiné à un bouton, appelé `my_btn` dans le scénario principal, permettant de lire un clip appelé `animation_mc`:

```
animation_mc.stop();
my_btn.onRelease = function(){
  animation_mc.play();
};
```

### Voir également

`Fonction play`, `gotoAndPlay (méthode MovieClip.gotoAndPlay)`, `Fonction gotoAndPlay`

## prevFrame (méthode MovieClip.prevFrame)

```
public prevFrame() : Void
```

Place la tête de lecture sur l'image précédente et l'arrête.

Vous pouvez étendre les méthodes et les gestionnaires d'événements de la classe MovieClip en créant une sous-classe.

**Disponibilité :** ActionScript 1.0 ; Flash Player 5

### Exemple

Dans l'exemple suivant, deux boutons de clip permettent de contrôler le scénario. Le bouton `prev_mc` déplace la tête de lecture sur l'image précédente alors que le bouton `next_mc` la déplace sur l'image suivante. Ajoutez du contenu à une série d'images du scénario et ajoutez le code ActionScript suivant sur l'image 1 du scénario :

```
stop();
prev_mc.onRelease = function() {
  var parent_mc:MovieClip = this._parent;
  if (parent_mc._currentframe>1) {
  parent_mc.prevFrame();
  } else {
  parent_mc.gotoAndStop(parent_mc._totalframes);
  }
};
```

```
next_mc.onRelease = function() {
  var parent_mc:MovieClip = this._parent;
  if (parent_mc._currentframe<parent_mc._totalframes) {
  parent_mc.nextFrame();
  } else {
  parent_mc.gotoAndStop(1);
  }
};
```

### Voir également

Fonction prevFrame

## _quality (propriété MovieClip._quality)

```
public _quality : String
```

Définit ou extrait la qualité du rendu appliqué à un fichier SWF. Les polices de périphérique sont toujours aliasées et ne sont donc pas affectées par la propriété `_quality`.

Vous pouvez définir `_quality` sur les valeurs suivantes :

| Valeur | Description | Anti-aliasing graphique | Lissage des bitmaps |
|---|---|---|---|
| `"LOW"` | Faible qualité de rendu. | Les graphiques ne sont pas anti-aliasés. | Les bitmaps ne sont pas lissés |
| `"MEDIUM"` | Qualité de rendu moyenne. Ce paramétrage convient aux animations qui ne contiennent pas de texte. | Les graphiques sont anti-aliasés selon une grille de 2 x 2 pixels. | Flash Player 8 : Les bitmaps sont lissés à partir du paramètre `smoothing` utilisé dans les appels `MovieClip.attachBitmap()` et `MovieClip.beginBitmapFill()`.<br>Flash Player version 6 et 7 : Les bitmaps ne sont pas lissés |

| Valeur | Description | Anti-aliasing graphique | Lissage des bitmaps |
|---|---|---|---|
| "HIGH" | Haute qualité de rendu. Il s'agit du paramètre de qualité de rendu par défaut de Flash. | Les graphiques sont anti-aliasés selon une grille de 4 x 4 pixels. | Flash Player 8 : Les bitmaps sont lissés à partir du paramètre smoothing utilisé dans les appels MovieClip.attachBitmap() et MovieClip.beginBitmapFill(). Flash Player version 6 et 7 : Les bitmaps sont lissés si le clip est statique. |
| "BEST" | Très haute qualité de rendu. | Les graphiques sont anti-aliasés selon une grille de 4 x 4 pixels. | Flash Player 8 : Les bitmaps sont lissés à partir du paramètre smoothing utilisé dans les appels MovieClip.attachBitmap() et MovieClip.beginBitmapFill(). Lorsque le paramètre smoothing est défini, le résultat se traduit par une plus grande qualité lorsque le clip est réduit en utilisant un algorithme de moyenne. Cela peut ralentir le rendu, mais permet par exemple de créer des vignettes de haute qualité pour des images de grande taille. Flash Player version 6 et 7 : Les bitmaps sont toujours lissés. |

*Remarque :* Bien que vous puissiez spécifier cette propriété pour un objet MovieClip, il s'agit également d'une propriété globale : il vous suffit donc de définir sa valeur sur_quality.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

### Exemple

Cet exemple définit la qualité de rendu d'un clip appelé `my_mc` sur `LOW`:

```
my_mc._quality = "LOW";
```

### Voir également

`_quality, propriété`

## removeMovieClip (méthode MovieClip.removeMovieClip)

```
public removeMovieClip() : Void
```

Supprime une occurrence de clip créée avec `duplicateMovieClip()`, `MovieClip.duplicateMovieClip()`, `MovieClip.createEmptyMovieClip()`, ou `MovieClip.attachMovie()`.

Cette méthode ne permet pas de supprimer un clip associé à une valeur négative de profondeur. Les clips créés avec l'outil de programmation reçoivent par défaut des valeurs négatives de profondeur. Pour supprimer un clip comportant une valeur négative de profondeur, appliquez tout d'abord la méthode `MovieClip.swapDepths()` pour convertir cette valeur en valeur positive.

*Remarque :* Si vous utilisez des composants de la version 2, n'utilisez pas cette méthode. Si vous placez un composant de la version 2 sur la scène ou dans la bibliothèque, la méthode `getNextHighestDepth()` renvoie parfois une valeur de profondeur de 1048676, qui est en dehors de la limite valide. Si vous utilisez des composants de la version 2, vous devez toujours utiliser les composants de la version 2 de la classe DepthManager.

*Remarque :* Si vous utilisez des composants version 2 et utilisez `MovieClip.getNextHighestDepth()` au lieu des composants version de la classe DepthManager pour affecter les valeurs de profondeur, `removeMovieClip()` risque d'échouer sans indication. Lorsqu'un composant version 2 est utilisé, la classe DepthManager réserve automatiquement les profondeurs disponibles, que ce soit la plus élevée (1048575) ou la plus basse (-16383), pour les curseurs et les info-bulles. Tout autre appel à `getNextHighestDepth()` renvoie 1048576, ce qui est en dehors des limites valides. La méthode `removeMovieClip()` échoue sans indication si elle détecte une valeur de profondeur en dehors de la plage valide. Si vous utilisez `getNextHighestDepth()` avec des composants version 2, vous pouvez utiliser `swapDepths()` pour affecter une valeur de profondeur valide ou utiliser `MovieClip.unloadMovie()` pour supprimer le contenu du clip. En outre, vous pouvez utiliser la classe DepthManager pour affecter des valeurs de profondeur au sein d'une plage valide.

Vous pouvez étendre les méthodes et les gestionnaires d'événements de la classe MovieClip en créant une sous-classe.

**Disponibilité :** ActionScript 1.0 ; Flash Player 5

### Exemple

A chaque fois que vous cliquez sur un bouton dans l'exemple suivant, vous liez une occurrence de clip à la scène à une position choisie au hasard. Lorsque vous cliquez sur une occurrence de clip, vous supprimez cette occurrence du fichier SWF.

```
function randRange(min:Number, max:Number):Number {
  var randNum:Number = Math.round(Math.random()*(max-min))+min;
  return randNum;
}
var bugNum:Number = 0;
addBug_btn.onRelease = addBug;
function addBug() {
  var thisBug:MovieClip = this._parent.attachMovie("bug_id",
  "bug"+bugNum+"_mc", bugNum,
{_x:randRange(50, 500), _y:randRange(50, 350)});
  thisBug.onRelease = function() {
  this.removeMovieClip();
  };
  bugNum++;
}
```

### Voir également

Fonction duplicateMovieClip, createEmptyMovieClip (méthode MovieClip.createEmptyMovieClip), duplicateMovieClip (méthode MovieClip.duplicateMovieClip), attachMovie (méthode MovieClip.attachMovie), swapDepths (méthode MovieClip.swapDepths)

## _rotation (propriété MovieClip._rotation)

```
public _rotation : Number
```

Spécifie la rotation du clip, en degrés, à partir de son orientation d'origine. Les valeurs comprises entre 0 et 180 représentent la rotation selon le sens horaire ; les valeurs comprises entre 0 et -180 représentent la rotation selon le sens anti-horaire. Les valeurs non comprises dans cette plage sont ajoutées à 360 ou soustraites de 360 pour obtenir une valeur comprise dans la plage ; par exemple, l'instruction `my_mc._rotation = 450` est la même que `my_mc._rotation = 90`.

**Disponibilité :** ActionScript 1.0 ; Flash Player 4

### Exemple

L'exemple suivant crée une occurrence de clip `triangle` de façon dynamique. Lorsque vous exécutez un fichier SWF, cliquez sur le clip pour le faire pivoter.

```
this.createEmptyMovieClip("triangle", this.getNextHighestDepth());

triangle.beginFill(0x0000FF, 100);
triangle.moveTo(100, 100);
triangle.lineTo(100, 150);
triangle.lineTo(150, 100);
triangle.lineTo(100, 100);

triangle.onMouseUp= function() {
  this._rotation += 15;
};
```

La méthode `MovieClip.getNextHighestDepth()` utilisée dans cet exemple nécessite Flash Player 7 ou une version ultérieure. Si votre fichier SWF comprend un composant de la version 2, utilisez les composants de la version 2 de la classe DepthManager au lieu de la méthode `MovieClip.getNextHighestDepth()`.

### Voir également

`_rotation (propriété Button._rotation)`, `_rotation (propriété TextField._rotation)`

## scale9Grid (propriété MovieClip.scale9Grid)

```
public scale9Grid : Rectangle
```

La zone rectangulaire qui définit les neuf zones de redimensionnement du clip. Si elle est définie sur `null`, l'ensemble du clip est redimensionné normalement, si nécessaire.

Lorsqu'une propriété `scale9Grid` est définie pour un clip, ce clip se voit appliquer une grille de neuf régions, en fonction du rectangle `scale9Grid` qui définit la zone centrale de la grille. Cette grille comporte huit autres zones :

- La zone située dans le coin supérieur gauche, en dehors du rectangle.
- La zone située au-dessus du rectangle
- La zone située dans le coin supérieur droit, en dehors du rectangle.
- La zone située à gauche du rectangle
- La zone située à droite du rectangle
- La zone située dans le coin inférieur gauche, en dehors du rectangle.
- La zone située en dessous du rectangle
- La zone située dans le coin inférieur droit, en dehors du rectangle.

Les huit zones entourant la partie centrale (définie par le rectangle) peuvent être conçues comme un cadre qui bénéficie de règles spécifiques de redimensionnement du clip.

Lorsque la propriété `scale9Grid` est définie et qu'un clip est redimensionné, l'ensemble du texte et tous les clips enfants sont redimensionnés normalement, quelle que soit leur position dans les zones de la grille scale9. Pour les autres types d'objet, les règles suivantes s'appliquent :

- Tout le contenu de la zone centrale est redimensionné normalement.
- Le contenu figurant éventuellement dans les angles n'est redimensionné que lorsque le facteur de redimensionnement de la zone centrale est de 0.
- Tout contenu figurant dans les zones du haut et du bas est redimensionné uniquement à l'horizontale. Le contenu figurant dans les zones de droite et de gauche se redimensionne uniquement à la verticale.
- Tous les remplissages (ce qui inclut les bitmaps, les vidéos et les dégradés) sont étirés pour remplir leur forme.

En cas de rotation d'un clip, les redimensionnements suivants sont normaux (et la propriété `scale9Grid` est ignorée).

Par exemple, considérez le clip suivant et un rectangle qui lui est appliqué en tant que propriété `scale9Grid` :


Le clip.


Le rectangle rouge représente la zone couverte par `scale9Grid`.

Lorsque le clip est redimensionné ou étiré, les objets placés dans le rectangle se redimensionnent normalement. Par contre, les objets situés en dehors du rectangle sont redimensionnés selon les règles de `scale9Grid` :

| | |
|---|---|
| Redimensionné à 75% : |   |
| Redimensionné à 50% : |   |
| Redimensionné à 25 % : |  |
| Etiré à l'horizontal à 150 % : |   |

L'une des principales utilisations de `scale9Grid` consiste à définir un composant où les lignes de bordure conservent la même largeur lorsque le composant est redimensionné.

Dans l'environnement auteur Macromedia Flash, vous pouvez activer les repères pour la *mise à l'échelle à 9 découpes* d'un symbole de clip dans une bibliothèque. Cela vous permet de déterminer graphiquement le paramètre `scale9grid` pour l'objet. Lorsque vous définissez la mise à l'échelle à 9 découpes d'un symbole, la propriété `scale9grid` de toute occurrence de ce symbole est automatiquement définie. Dans le cas d'un symbole qui a une mise à l'échelle à 9 découpes activée, lorsque vous créez le fichier SWF, une courbe qui s'étend sur plus d'une zone de la grille de mise à l'échelle à 9 découpes est divisée en courbes séparées pour chaque zone de la grille. Prenez par exemple une courbe dans un symbole de clip pour lequel une mise à l'échelle à 9 découpes est activée et la même courbe dans un symbole de clip pour lequel une mise à l'échelle à 9 découpes n'est *pas* activée :

| | |
|---|---|
| Symbole avec mise à l'échelle à 9 découpes activée : |  |
| Symbole sans mise à l'échelle à 9 découpes activée : |  |

Lorsque Flash crée le fichier SWF, la courbe dans le premier clip illustré est divisée en trois courbes. Ce n'est pas le cas dans le second clip, pour lequel la mise à l'échelle à 9 découpes n'est pas activée. Même si vous définissez le paramètre `scale9Grid` du second clip sur un rectangle correspondant au paramètre `scale9Grid` du premier clip, lorsque vous redimensionnez ces clips, les résultats diffèrent en raison de la manière dont Flash divise les courbes dans le premier clip :

| | |
|---|---|
| Symbole avec mise à l'échelle à 9 découpes activée |  |
| Symbole sans mise à l'échelle à 9 découpes activée |  |

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

## Exemple

L'exemple suivant crée un clip qui contient une ligne de 20 pixels (formant la bordure) et un remplissage en dégradé. Le clip se redimensionne par rapport à la position de la souris et, en fonction de la propriété `scale9Grid` définie pour le clip, l'épaisseur de la ligne de 20 pixels ne varie pas lorsque le clip est redimensionné (bien que le dégradé du clip *soit* redimensionné) :

```
import flash.geom.Rectangle;
import flash.geom.Matrix;
```

```
this.createEmptyMovieClip("my_mc", this.getNextHighestDepth());

var grid:Rectangle = new Rectangle(20, 20, 260, 260);
my_mc.scale9Grid = grid ;

my_mc._x = 50;
my_mc._y = 50;

function onMouseMove()
{
  my_mc._width = _xmouse;
  my_mc._height = _ymouse;
}

my_mc.lineStyle(20, 0xff3333, 100);
var gradient_matrix:Matrix = new Matrix();
gradient_matrix.createGradientBox(15, 15, Math.PI, 10, 10);
my_mc.beginGradientFill("radial", [0xffff00, 0x0000ff],
       [100, 100], [0, 0xFF], gradient_matrix,
       "reflect", "RGB", 0.9);
my_mc.moveTo(0, 0);
my_mc.lineTo(0, 300);
my_mc.lineTo(300, 300);
my_mc.lineTo(300, 0);
my_mc.lineTo(0, 0);
my_mc.endFill();
```

### Voir également

Rectangle (flash.geom.Rectangle)

## scrollRect (propriété MovieClip.scrollRect)

```
public scrollRect : Object
```

La propriété `scrollRect` permet de parcourir rapidement le contenu du clip et d'ouvrir une fenêtre plus grande pour afficher davantage de contenu. Les champs texte et le contenu défilent beaucoup plus vite dans la mesure où la copie au niveau des pixels permet de faire défiler les données sans avoir à générer de nouveau tout le clip à partir de données vectorielles. Pour afficher le gain de performances, utilisez `scrollRect` en combinaison avec un clip dont le paramètre `cacheAsBitmap` est défini sur `true`.

Le clip est recadré et défile selon des paramètres de décalage de largeur, hauteur et défilement spécifiques. Les propriétés `scrollRect` sont stockées dans l'espace de coordonnées du clip et sont redimensionnées comme le reste du clip. Les extrémités des angles de la fenêtre recadrée du clip à faire défiler servent de coordonnées d'origine (0, 0) et de point de référence (`scrollWidth`, `scrollHeight`). Ces points ne sont pas centrés autour de l'origine, mais utilisent le coin supérieur gauche comme origine. Le défilement des clips se fait toujours par pixels entiers. Si le clip subit une rotation de 90 degrés et que vous le faites défiler vers la gauche et la droite (en définissant la propriété `scrollRect.x`), il défile vers le haut et le bas.

Si le clip est associé à l'objet flash.geom.Rectangle, le clip est recadré pour obtenir une taille donnée et défile.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

## Exemple

L'exemple suivant définit une hiérarchie MovieClip (en appelant la fonction `setUpContainer()`), puis il définit un nouveau Rectangle en tant que propriété `scrollRect`.

```
import flash.geom.Rectangle;
var container:MovieClip = setUpContainer();
var window:Rectangle = new Rectangle(0, 0, 100, 40);
container.scrollRect = window;

function setUpContainer():MovieClip {
  var mc:MovieClip = this.createEmptyMovieClip("container",
  this.getNextHighestDepth());
  mc._x = 50;
  mc._y = 50;
  mc.opaqueBackground = 0xCCCCCC;

  var content:MovieClip = mc.createEmptyMovieClip("content",
  mc.getNextHighestDepth());
  var colors:Array = [0xFF0000, 0x0000FF];
  var alphas:Array = [100, 100];
  var ratios:Array = [0, 0xFF];
  var matrix:Object = {a:150, b:0, c:0, d:0, e:150, f:0, g:150, h:150,
  i:1};
  content.beginGradientFill("linear", colors, alphas, ratios, matrix);
  content.lineTo(300, 0);
  content.lineTo(300, 300);
  content.lineTo(0, 300);
  content.lineTo(0, 0);
  content.endFill();
  content._rotation = -90;

  mc.onEnterFrame = function() {
    this.content._y += 1;
  }
```

```
  return mc;
}
```

La fonction `setUpContainer()` effectue les opérations suivantes :

- Crée un clip appelé `container`
- Crée un clip appelé `content` dans un `container`
- Trace une forme en dégradé à l'intérieur du clip `content`
- Renvoie une référence au clip `container`

Si votre fichier SWF comprend un composant de la version 2, utilisez les composants de la version 2 de la classe DepthManager au lieu de la méthode `MovieClip.getNextHighestDepth()`, utilisée dans cet exemple.

# setMask (méthode MovieClip.setMask)

```
public setMask(mc:Object) : Void
```

Définit le clip du paramètre `mc` comme étant un masque qui révèle le clip appelant.

La méthode `setMask()` permet à plusieurs clips d'image au contenu multicouche complexe de faire office de masques (ce qui est possible avec des calques de masque). Si un clip masqué dispose de polices de périphérique, celles-ci sont tracées, et non masquées. Vous ne pouvez pas définir un clip comme étant son propre masque, par exemple, `my_mc.setMask(my_mc)`.

Si vous créez un calque de masque contenant un clip, puis que vous appliquez la méthode `setMask()` à celui-ci, l'appel `setMask()` est prioritaire : cette action n'est pas réversible. Par exemple, vous pouvez disposer d'un clip dans d'un calque de masque appelé `UIMask` masquant un autre calque qui contient un autre clip appelé `UIMaskee`. Si, pendant la lecture du fichier SWF, vous appelez `UIMask.setMask(UIMaskee)`, à partir de ce moment, `UIMask` est masqué par `UIMaskee`.

Pour annuler un masque créé avec ActionScript, transmettez la valeur `null` à la méthode `setMask()`. Le code suivant annule le masque sans affecter le calque de masque dans le scénario.

```
UIMask.setMask(null);
```

Vous pouvez étendre les méthodes et les gestionnaires d'événements de la classe MovieClip en créant une sous-classe.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

## Paramètres

*`mc`*`:Object` - Nom d'occurrence du clip à transformer en masque. Il peut s'agir d'une chaîne ou d'un clip.

### Exemple

Le code suivant utilise le clip `circleMask_mc` pour masquer le clip `theMaskee_mc` :

```
theMaskee_mc.setMask(circleMask_mc);
```

## _soundbuftime (propriété MovieClip._soundbuftime)

```
public _soundbuftime : Number
```

Spécifie le nombre de secondes pendant lequel les sons sont chargés en mémoire tampon avant d'être diffusés en continu.

*Remarque :* Bien que vous puissiez spécifier cette propriété pour un objet MovieClip, il s'agit en fait d'une propriété globale qui s'applique à l'ensemble des sons qui ont été chargés. Il vous suffit donc de définir sa valeur sur `_soundbuftime`. La définition de cette propriété pour un objet MovieClip définit en fait la propriété globale.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

### Voir également

`_soundbuftime, propriété`

## startDrag (méthode MovieClip.startDrag)

```
public startDrag([lockCenter:Boolean], [left:Number], [top:Number],
  [right:Number], [bottom:Number]) : Void
```

Permet à l'utilisateur de faire glisser le clip spécifié. Le clip reste déplaçable jusqu'à son arrêt explicite par un appel de `MovieClip.stopDrag()`, ou jusqu'à ce qu'un autre clip soit déplaçable. Vous ne pouvez déplacer qu'un clip à la fois.

Vous pouvez étendre les méthodes et les gestionnaires d'événements de la classe MovieClip en créant une sous-classe.

**Disponibilité :** ActionScript 1.0 ; Flash Player 5

### Paramètres

*`lockCenter`*`:Boolean` [facultatif] - Valeur booléenne spécifiant si le clip à déplacer doit être verrouillé au centre de la position de la souris (`true`), ou verrouillé au point où l'utilisateur a cliqué sur le clip en premier lieu (`false`).

*`left`*`:Number` [facultatif] - Valeur relative aux coordonnées du parent du clip qui spécifie un rectangle de délimitation pour le clip.

*`top`*`:Number` [facultatif] - Valeur relative aux coordonnées du parent du clip qui spécifie un rectangle de délimitation pour le clip.

*right*:Number [facultatif] - Valeur relative aux coordonnées du parent du clip qui spécifie un rectangle de délimitation pour le clip.

*bottom*:Number [facultatif] - Valeur relative aux coordonnées du parent du clip qui spécifie un rectangle de délimitation pour le clip.

### Exemple

L'exemple suivant crée une occurrence déplaçable de clip appelée mc_1:

```
this.createEmptyMovieClip("mc_1", 1);

with (mc_1) {
  lineStyle(1, 0xCCCCCC);
  beginFill(0x4827CF);
  moveTo(0, 0);
  lineTo(80, 0);
  lineTo(80, 60);
  lineTo(0, 60);
  lineTo(0, 0);
  endFill();
}

mc_1.onPress = function() {
  this.startDrag();
};
mc_1.onRelease = function() {
  this.stopDrag();
};
```

### Voir également

_droptarget (propriété MovieClip._droptarget), Fonction startDrag, stopDrag (méthode MovieClip.stopDrag)

## stop (méthode MovieClip.stop)

```
public stop() : Void
```

Arrête le clip en cours de lecture.

Vous pouvez étendre les méthodes et les gestionnaires d'événements de la classe MovieClip en créant une sous-classe.

**Disponibilité :** ActionScript 1.0 ; Flash Player 5

### Exemple

L'exemple suivant indique comment arrêter un clip appelé aMovieClip:

```
aMovieClip.stop();
```

### Voir également

Fonction stop

## stopDrag (méthode MovieClip.stopDrag)

```
public stopDrag() : Void
```

Termine une méthode `MovieClip.startDrag()`. Un clip déplaçable via cette méthode reste déplaçable jusqu'à ce qu'une méthode `stopDrag()` soit ajoutée, ou jusqu'à ce qu'un autre clip devienne déplaçable. Vous ne pouvez déplacer qu'un seul clip à la fois.

Vous pouvez étendre les méthodes et les gestionnaires d'événements de la classe MovieClip en créant une sous-classe.

**Disponibilité :** ActionScript 1.0 ; Flash Player 5

### Exemple

L'exemple suivant crée une occurrence déplaçable de clip appelée `mc_1`:

```
this.createEmptyMovieClip("mc_1", 1);

with (mc_1) {
  lineStyle(1, 0xCCCCCC);
  beginFill(0x4827CF);
  moveTo(0, 0);
  lineTo(80, 0);
  lineTo(80, 60);
  lineTo(0, 60);
  lineTo(0, 0);
  endFill();
}

mc_1.onPress = function() {
  this.startDrag();
};
mc_1.onRelease = function() {
  this.stopDrag();
};
```

### Voir également

_droptarget (propriété MovieClip._droptarget), startDrag (méthode MovieClip.startDrag), Fonction stopDrag

# swapDepths (méthode MovieClip.swapDepths)

```
public swapDepths(target:Object) : Void
```

Intervertit l'empilement, ou le niveau de profondeur (ordre z), de ce clip avec le clip spécifié par le paramètre `target` ou avec le clip qui occupe actuellement le niveau de profondeur spécifié dans le paramètre `target`. Les deux clips doivent avoir le même clip parent. La permutation du niveau de profondeur des clips revient à déplacer un clip devant ou derrière l'autre. Si l'appel de cette méthode provoque l'interpolation d'un clip, l'interpolation est arrêtée.

Vous pouvez étendre les méthodes et les gestionnaires d'événements de la classe MovieClip en créant une sous-classe.

**Disponibilité :** ActionScript 1.0 ; Flash Player 5

## Paramètres

*`target`*`:Object` - Ce paramètre peut prendre deux formes :

- Un nombre spécifiant le niveau de profondeur du clip.
- Une chaîne spécifiant l'occurrence de clip dont la profondeur est permutée avec le clip pour lequel la méthode est appliquée. Les deux clips doivent avoir le même clip parent.

## Exemple

L'exemple suivant permet de permuter l'ordre des deux occurrences de clip. Superposez deux occurrences de clip, appelées `myMC1_mc` et `myMC2_mc`, sur la scène et ajoutez ensuite le script suivant au scénario parent :

```
myMC1_mc.onRelease = function() {
  this.swapDepths(myMC2_mc);
};
myMC2_mc.onRelease = function() {
  this.swapDepths(myMC1_mc);
};
```

## Voir également

`_level`, propriété, `getDepth` (méthode MovieClip.getDepth), `getInstanceAtDepth` (méthode MovieClip.getInstanceAtDepth), `getNextHighestDepth` (méthode MovieClip.getNextHighestDepth)

# tabChildren (propriété MovieClip.tabChildren)

```
public tabChildren : Boolean
```

Détermine si les enfants d'un clip sont inclus dans l'ordre de tabulation automatique. Si la propriété `tabChildren` est définie sur `undefined` ou `true`, les enfants d'un clip sont inclus dans l'ordre de tabulation automatique. Si la valeur de `tabChildren` est `false`, les enfants d'un clip ne sont pas inclus dans l'ordre de tabulation automatique. La valeur par défaut est `undefined`.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

## Exemple

Un widget de zone de liste, créé en tant que clip comportant plusieurs éléments, dans une interface utilisateur. L'utilisateur peut cliquer sur les différents éléments pour les sélectionner, ce qui signifie que chaque élément est représenté par un bouton. Cependant, seule la zone de liste doit pouvoir être sélectionnée avec la touche de tabulation. Les éléments figurant dans la zone de liste doivent être exclus de l'ordre de tabulation. Pour ce faire, vous définissez la propriété `tabChildren` de la zone de liste sur `false`.

La propriété `tabChildren` reste sans effet lorsque la propriété `tabIndex` est utilisée ; la propriété `tabChildren` affecte uniquement l'ordre de tabulation automatique.

L'exemple suivant désactive l'ordre de tabulation de tous les clips enfants au sein d'un clip parent appelé `menu_mc`:

```
menu_mc.onRelease = function(){};
menu_mc.menu1_mc.onRelease = function(){};
menu_mc.menu2_mc.onRelease = function(){};
menu_mc.menu3_mc.onRelease = function(){};
menu_mc.menu4_mc.onRelease = function(){};

menu_mc.tabChildren = false;
```

Modifiez la dernière ligne de code ci-dessous, de façon à inclure les occurrences de clip enfant de `menu_mc` dans l'ordre de tabulation automatique :

```
menu_mc.tabChildren = true;
```

## Voir également

`tabIndex (propriété Button.tabIndex)`, `tabEnabled (propriété MovieClip.tabEnabled)`, `tabIndex (propriété MovieClip.tabIndex)`, `tabIndex (propriété TextField.tabIndex)`

# tabEnabled (propriété MovieClip.tabEnabled)

```
public tabEnabled : Boolean
```

Spécifie si le clip est inclus dans l'ordre de tabulation automatique. La valeur par défaut est `undefined`.

Si la propriété `tabEnabled` est définie sur `undefined`, l'objet n'est inclus dans l'ordre de tabulation automatique que si au moins un gestionnaire de clip, comme `MovieClip.onRelease`, est défini. Si la propriété `tabEnabled` est définie sur `true`, l'objet est inclus dans l'ordre de tabulation automatique. Si la propriété `tabIndex` est également définie sur une valeur, l'objet est également inclus dans l'ordre de tabulation personnalisé.

Si la propriété `tabEnabled` est définie sur `false`, l'objet n'est pas inclus dans l'ordre de tabulation automatique ou personnalisé, même si la propriété `tabIndex` est définie. Cependant, si la valeur de `MovieClip.tabChildren` est définie sur `true`, vous pouvez toujours inclure les enfants du clip dans l'ordre de tabulation automatique même si la valeur de `tabEnabled` est définie sur `false`.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

## Exemple

L'exemple suivant exclut `myMC2_mc` de l'ordre de tabulation automatique :

```
myMC1_mc.onRelease = function() {};
myMC2_mc.onRelease = function() {};
myMC3_mc.onRelease = function() {};
myMC2_mc.tabEnabled = false;
```

## Voir également

`onRelease (gestionnaire MovieClip.onRelease)`, `tabEnabled (propriété Button.tabEnabled)`, `tabChildren (propriété MovieClip.tabChildren)`, `tabIndex (propriété MovieClip.tabIndex)`, `tabEnabled (propriété TextField.tabEnabled)`

# tabIndex (propriété MovieClip.tabIndex)

```
public tabIndex : Number
```

Permet de personnaliser l'ordre de tabulation des objets dans un clip. La propriété `tabIndex` est `undefined` par défaut. Vous pouvez définir la propriété `tabIndex` sur un bouton, un clip ou sur une occurrence de champ de texte.

Si un objet contenu dans un fichier SWF contient une propriété `tabIndex`, l'ordre de tabulation automatique est désactivé : l'ordre de tabulation est alors calculé à partir des propriétés `tabIndex` des objets contenus dans le fichier SWF. L'ordre de tabulation personnalisé inclut uniquement des objets dotés de propriétés `tabIndex`.

La propriété `tabIndex` peut être un entier positif. Les objets sont triés selon leurs propriétés `tabIndex`, par ordre croissant. Un objet ayant une valeur `tabIndex` de `1` précède un objet `tabIndex` d'une valeur de `2`. L'ordre de tabulation personnalisé ignore les relations hiérarchiques des objets contenus dans un fichier SWF. Tous les objets du fichier SWF dotés de propriétés `tabIndex` sont placés dans l'ordre de tabulation. N'appliquez pas la même valeur `tabIndex` à plusieurs objets.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

### Exemple

Le code ActionScript suivant met en place un ordre de tabulation personnalisé pour trois occurrences de clip.

```
myMC1_mc.onRelease = function() {};
myMC2_mc.onRelease = function() {};
myMC3_mc.onRelease = function() {};
myMC1_mc.tabIndex = 2;
myMC2_mc.tabIndex = 1;
myMC3_mc.tabIndex = 3;
```

### Voir également

`tabIndex (propriété Button.tabIndex)`, `tabIndex (propriété TextField.tabIndex)`

## _target (propriété MovieClip._target)

```
public _target : Chaîne [lecture seule]
```

Renvoie le chemin cible de l'occurrence de clip, en notation avec barre oblique. Utilisez la fonction `eval()` pour convertir le chemin cible en notation avec point.

**Disponibilité :** ActionScript 1.0 ; Flash Player 4

### Exemple

L'exemple suivant affiche les chemins cibles des occurrences de clip dans un fichier SWF, en notation à barre oblique aussi bien que point.

```
for (var i in this) {
  if (typeof (this[i]) == "movieclip") {
  trace("name: " + this[i]._name + ",\t target: " + this[i]._target + ",\t
  target(2):"
       + eval(this[i]._target));
  }
}
```

# _totalframes (propriété MovieClip._totalframes)

```
public _totalframes : Nombre [lecture seule]
```

Renvoie le nombre total d'images dans l'occurrence de clip spécifiée par le paramètre `MovieClip`.

**Disponibilité :** ActionScript 1.0 ; Flash Player 4

## Exemple

Dans l'exemple suivant, deux boutons de clip permettent de contrôler le scénario. Le bouton `prev_mc` déplace la tête de lecture vers l'image précédente alors que le bouton `next_mc` la déplace vers l'image suivante. Ajoutez du contenu à une série d'images du scénario et ajoutez le code ActionScript suivant sur l'image 1 du scénario :

```
stop();
prev_mc.onRelease = function() {
  var parent_mc:MovieClip = this._parent;
  if (parent_mc._currentframe>1) {
  parent_mc.prevFrame();
  } else {
  parent_mc.gotoAndStop(parent_mc._totalframes);
  }
};
next_mc.onRelease = function() {
  var parent_mc:MovieClip = this._parent;
  if (parent_mc._currentframe<parent_mc._totalframes) {
  parent_mc.nextFrame();
  } else {
  parent_mc.gotoAndStop(1);
  }
};
```

# trackAsMenu (propriété MovieClip.trackAsMenu)

```
public trackAsMenu : Boolean
```

Valeur booléenne indiquant si d'autres boutons ou clips peuvent recevoir des événements de relâchement de souris. La propriété `trackAsMenu` vous permet de créer des menus. Vous pouvez définir la propriété `trackAsMenu` sur n'importe quel bouton ou objet de clip. Si la propriété `trackAsMenu` n'existe pas, la valeur du comportement par défaut devient `false`.

Vous pouvez modifier la propriété `trackAsMenu` à tout moment ; le clip modifié accepte immédiatement le nouveau comportement.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

### Exemple

L'exemple suivant définit la propriété `trackAsMenu` pour trois clips de la scène. Cliquez sur un clip et relâchez le bouton de la souris sur un autre clip pour déterminer l'occurrence recevant l'événement.

```
myMC1_mc.trackAsMenu = true;
myMC2_mc.trackAsMenu = true;
myMC3_mc.trackAsMenu = false;

myMC1_mc.onRelease = clickMC;
myMC2_mc.onRelease = clickMC;
myMC3_mc.onRelease = clickMC;

function clickMC() {
  trace("you clicked the "+this._name+" movie clip.");
};
```

### Voir également

`trackAsMenu (propriété Button.trackAsMenu)`

## transform (propriété MovieClip.transform)

```
public transform : Transform
```

Un objet avec des propriétés se rapportant à la matrice d'un clip, à la transformation des couleurs et aux limites des pixels. Les propriétés spécifiques, telles que matrix, colorTransform et trois propriétés en lecture seule (`concatenatedMatrix`, `concatenatedColorTransform`, et `pixelBounds`) sont décrites dans la section relative à la classe Transform.

Chacune des propriétés de l'objet transform constitue un objet. Ceci est important dans la mesure où la seule façon de définir de nouvelles valeurs pour les objets matrix ou colorTransform consiste à créer un objet et à le copier dans la propriété transform.matrix ou transform.colorTransform.

Par exemple, pour augmenter la valeur `tx` d'une matrice de clip, vous devez copier l'intégralité de l'objet matrix, modifier la propriété `tx` du nouvel objet, puis copier le nouvel objet dans la propriété matrix de l'objet transform :

```
var myMatrix:Object = myDisplayObject.transform.matrix;
myMatrix.tx += 10;
myDisplayObject.transform.matrix = myMatrix;
```

Vous ne pouvez pas définir directement la propriété `tx`. Le code suivant n'a pas d'effet sur `myDisplayObject`: `myDisplayObject.transform.matrix.tx += 10;`

Vous pouvez copier un objet transform et l'associer à la propriété transform d'un autre clip. Par exemple, le code suivant copie l'objet transform dans son intégralité, de `myOldDisplayObj` à `myNewDisplayObj` :

```
myNewDisplayObj.transform = myOldDisplayObj.transform;
```

Le nouveau clip, `myNewDisplayObj`, dispose désormais des mêmes valeurs que l'ancien clip pour sa matrice, pour la transformation de couleurs et les limites de pixels.`myOldDisplayObj`.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

## Exemple

L'exemple suivant montre comment utiliser la propriété `transform` d'un clip pour accéder et modifier l'emplacement du clip en utilisant le positionnement de matrice.

```
import flash.geom.Matrix;

var rect:MovieClip = createRectangle(20, 80, 0xFF0000);

var translateMatrix:Matrix = new Matrix();
translateMatrix.translate(10, 0);

rect.onPress = function() {
  var tmpMatrix:Matrix = this.transform.matrix;
  tmpMatrix.concat(translateMatrix);
  this.transform.matrix = tmpMatrix;
}

function createRectangle(width:Number, height:Number, color:Number,
  scope:MovieClip):MovieClip {
  scope = (scope == undefined) ? this : scope;
  var depth:Number = scope.getNextHighestDepth();
  var mc:MovieClip = scope.createEmptyMovieClip("mc_" + depth, depth);
  mc.beginFill(color);
  mc.lineTo(0, height);
  mc.lineTo(width, height);
  mc.lineTo(width, 0);
  mc.lineTo(0, 0);
  return mc;
}
```

Si votre fichier SWF comprend un composant de la version 2, utilisez les composants de la version 2 de la classe DepthManager au lieu de la méthode `MovieClip.getNextHighestDepth()`, utilisée dans cet exemple.

## Voir également

`Transform (flash.geom.Transform)`

# unloadMovie (méthode MovieClip.unloadMovie)

```
public unloadMovie() : Void
```

Supprime le contenu d'une occurrence de clip. Les propriétés de l'occurrence et les gestionnaires de clip restent inchangés.

Pour supprimer l'occurrence, y compris ses propriétés et les gestionnaires de clip, utilisez `MovieClip.removeMovieClip()`.

Vous pouvez étendre les méthodes et les gestionnaires d'événements de la classe MovieClip en créant une sous-classe.

**Disponibilité :** ActionScript 1.0 ; Flash Player 5

## Exemple

L'exemple suivant purge une occurrence de clip appelée `box` lorsqu'un utilisateur clique sur le clip `box` :

```
this.createEmptyMovieClip("box", 1);

with (box) {
  lineStyle(1, 0xCCCCCC);
  beginFill(0x4827CF);
  moveTo(0, 0);
  lineTo(80, 0);
  lineTo(80, 60);
  lineTo(0, 60);
  lineTo(0, 0);
  endFill();
}

box.onRelease = function() {
    box.unloadMovie();
};
```

## Voir également

`removeMovieClip (méthode MovieClip.removeMovieClip)`, `attachMovie (méthode MovieClip.attachMovie)`, `loadMovie (méthode MovieClip.loadMovie)`, `Fonction unloadMovie`, `Fonction unloadMovieNum`

# _url (propriété MovieClip._url)

```
public _url : Chaîne [lecture seule]
```

Récupère l'URL du fichier SWF, JPEG, GIF ou PNG ayant servi à télécharger le clip.

**Disponibilité :** ActionScript 1.0 ; Flash Player 4

## Exemple

L'exemple suivant affiche l'URL de l'image chargée dans l'occurrence `image_mc` dans le panneau de sortie.

```
this.createEmptyMovieClip("image_mc", 1);
var mclListener:Object = new Object();
mclListener.onLoadInit = function(target_mc:MovieClip) {
  trace("_url: "+target_mc._url);
};
var image_mcl:MovieClipLoader = new MovieClipLoader();
image_mcl.addListener(mclListener);
image_mcl.loadClip("http://www.macromedia.com/images/shared/product_boxes/
  112x112/box_studio_112x112.jpg", image_mc);
```

L'exemple suivant associe l'objet `menu_cm` de ContextMenu au clip `image_mc`. L'objet `menu_cm` contient un élément de menu personnalisé appelé `View Image in Browser` qui a une fonction associée appelée `viewImage()`.

```
var menu_cm:ContextMenu = new ContextMenu();
menu_cm.customItems.push(new ContextMenuItem("View Image in Browser...",
  viewImage));
this.createEmptyMovieClip("image_mc", this.getNextHighestDepth());
var mclListener:Object = new Object();
mclListener.onLoadInit = function(target_mc:MovieClip) {
  target_mc.menu = menu_cm;
};
var image_mcl:MovieClipLoader = new MovieClipLoader();
image_mcl.addListener(mclListener);
image_mcl.loadClip("photo1.jpg", image_mc);

function viewImage(target_mc:MovieClip, obj:Object) {
  getURL(target_mc._url, "_blank");
}
```

Lorsque vous cliquez avec le bouton droit de la souris (Windows) ou en appuyant sur la touche Contrôle (Macintosh) sur l'image pendant l'exécution, sélectionnez View Image in Browser dans le menu contextuel pour ouvrir l'image dans une fenêtre de navigateur.

La classe MovieClipLoader utilisée dans ces exemples nécessite Flash Player 7 ou une version ultérieure. La méthode `MovieClip.getNextHighestDepth()` utilisée dans ces exemples nécessite Flash Player 7 ou une version ultérieure. Si votre fichier SWF comprend un composant de la version 2, utilisez les composants de la version 2 de la classe DepthManager au lieu de la méthode `MovieClip.getNextHighestDepth()`.

# useHandCursor (propriété MovieClip.useHandCursor)

```
public useHandCursor : Boolean
```

Valeur booléenne indiquant si le curseur en forme de main apparaît lorsque la souris passe sur un clip. La valeur par défaut de la propriété `useHandCursor` est `true`. Si la propriété `useHandCursor` est définie sur `true`, le curseur en forme de main utilisé pour les boutons s'affiche lorsque la souris survole un clip. Si la propriété `useHandCursor` est définie sur `false`, le pointeur flèche est utilisé.

Vous pouvez modifier la propriété `useHandCursor` à tout moment ; le clip modifié accepte immédiatement le nouveau comportement du curseur. La propriété `useHandCursor` peut être extraite d'un objet prototype.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

## Exemple

L'exemple suivant définit la propriété `useHandCursor` pour deux clips appelés `myMC1_mc` et `myMC2_mc`. Cette propriété doit être définie sur `true` pour l'une des occurrences et sur `false` pour l'autre. Vous pouvez constater que les deux occurrences peuvent toujours recevoir des événements.

```
myMC1_mc.onRelease = traceMC;
myMC2_mc.onRelease = traceMC;
myMC2_mc.useHandCursor = false;

function traceMC() {
  trace("you clicked: "+this._name);
};
```

# _visible (propriété MovieClip._visible)

```
public _visible : Boolean
```

Valeur booléenne indiquant si le clip est visible. Les clips qui ne sont pas visibles (propriété_`visible` définie sur `false`) sont désactivés. Par exemple, vous ne pouvez pas cliquer sur le bouton d'un clip dont la propriété `_visible` est définie sur `false`.

**Disponibilité :** ActionScript 1.0 ; Flash Player 4

### Exemple

L'exemple suivant définit la propriété `_visible` pour deux clips appelés `myMC1_mc` et `myMC2_mc`. Cette propriété doit être définie sur `true` pour l'une des occurrences et sur `false` pour l'autre. Vous pouvez alors constater qu'il est impossible de cliquer sur l'occurrence `myMC1_mc` lorsque la propriété `_visible` est définie sur `false`.

```
myMC1_mc.onRelease = function() {
  trace(this._name+"._visible = false");
  this._visible = false;
};
myMC2_mc.onRelease = function() {
  trace(this._name+"._alpha = 0");
  this._alpha = 0;
};
```

### Voir également

`_visible (propriété Button._visible)`, `_visible (propriété TextField._visible)`

## _width (propriété MovieClip._width)

```
public _width : Number
```

Largeur du clip, en pixels.

**Disponibilité :** ActionScript 1.0 ; Flash Player 4

### Exemple

L'exemple de code suivant affiche la hauteur et la largeur d'un clip dans le panneau de sortie :

```
this.createEmptyMovieClip("triangle", this.getNextHighestDepth());

triangle.beginFill(0x0000FF, 100);
triangle.moveTo(100, 100);
triangle.lineTo(100, 150);
triangle.lineTo(150, 100);
triangle.lineTo(100, 100);

trace(triangle._name + " = " + triangle._width + " X " + triangle._height +
  " pixels");
```

La méthode `MovieClip.getNextHighestDepth()` utilisée dans cet exemple nécessite Flash Player 7 ou une version ultérieure. Si votre fichier SWF comprend un composant de la version 2, utilisez les composants de la version 2 de la classe DepthManager au lieu de la méthode `MovieClip.getNextHighestDepth()`.

### Voir également

_height (propriété MovieClip._height)

## _x (propriété MovieClip._x)

```
public _x : Number
```

Entier qui définit la coordonnée *x* d'un clip par rapport aux coordonnées locales du clip parent. Si un clip se trouve dans le scénario principal, son système de coordonnées se réfère alors au coin supérieur gauche de la scène : (0, 0). Si le clip est imbriqué dans un autre clip subissant des transformations, le clip se trouve dans le système de coordonnées local du clip qui l'encadre. Ainsi, dans le cas d'un clip qui a effectué une rotation à 90 ° en sens anti-horaire, les enfants du clip héritent d'un système de coordonnées ayant effectué une rotation à 90 ° en sens anti-horaire. Les coordonnées du clip renvoient à la position du point d'alignement.

**Disponibilité :** ActionScript 1.0 ; Flash Player 3

### Exemple

L'exemple suivant associe un clip portant l'identifiant de liaison `cursor_id` à un fichier SWF. Le clip est appelé `cursor_mc`, et permet de remplacer le pointeur de souris par défaut. Le code ActionScript suivant définit les coordonnées actuelles de l'occurrence de clip en fonction de l'emplacement actuel du pointeur de la souris.

```
this.attachMovie("cursor_id", "cursor_mc", this.getNextHighestDepth(),
  {_x:_xmouse, _y:_ymouse});
Mouse.hide();
var mouseListener:Object = new Object();
mouseListener.onMouseMove = function() {
  cursor_mc._x = _xmouse;
  cursor_mc._y = _ymouse;
  updateAfterEvent();
};
Mouse.addListener(mouseListener);
```

La méthode `MovieClip.getNextHighestDepth()` utilisée dans cet exemple nécessite Flash Player 7 ou une version ultérieure. Si votre fichier SWF comprend un composant de la version 2, utilisez les composants de la version 2 de la classe DepthManager au lieu de la méthode `MovieClip.getNextHighestDepth()`.

### Voir également

_xscale (propriété MovieClip._xscale), _y (propriété MovieClip._y), _yscale (propriété MovieClip._yscale)

# _xmouse (propriété MovieClip._xmouse)

```
public _xmouse : Nombre [lecture seule]
```

Renvoie la coordonnée *x* de la position de la souris.

**Disponibilité :** ActionScript 1.0 ; Flash Player 5

## Exemple

L'exemple suivant renvoie les coordonnées actuelles *x* et *y* de la souris sur la scène (`_level0`) et par rapport à un clip appelé `my_mc` figurant également sur la scène :

```
this.createTextField("mouse_txt", this.getNextHighestDepth(), 0, 0, 150,
  66);
mouse_txt.html = true;
mouse_txt.multiline = true;
var row1_str:String = " \t<b>_xmouse\t</b><b>_ymouse</b>";
my_mc.onMouseMove = function() {
  mouse_txt.htmlText = "<textformat tabStops='[50,100]'>";
  mouse_txt.htmlText += row1_str;
  mouse_txt.htmlText += "<b>_level0</b>\t"+_xmouse+"\t"+_ymouse;
  mouse_txt.htmlText += "<b>my_mc</b>\t"+this._xmouse+"\t"+this._ymouse;
  mouse_txt.htmlText += "</textformat>";
};
```

La méthode `MovieClip.getNextHighestDepth()` utilisée dans cet exemple nécessite Flash Player 7 ou une version ultérieure. Si votre fichier SWF comprend un composant de la version 2, utilisez les composants de la version 2 de la classe DepthManager au lieu de la méthode `MovieClip.getNextHighestDepth()`.

## Voir également

`Souris`, `_ymouse (propriété MovieClip._ymouse)`

# _xscale (propriété MovieClip._xscale)

```
public _xscale : Number
```

Détermine le redimensionnement horizontal du clip (*`percentage`*) tel qu'il est appliqué à partir du point d'alignement du clip. Le point d'alignement par défaut est (0,0).

Le redimensionnement du système de coordonnées local affecte les paramètres des propriétés `_x` et `_y` définis en pixels. Par exemple, si le clip parent est redimensionné à 50 %, le paramétrage de la propriété `_x` déplace un objet dans le clip selon un nombre de pixels réduit de moitié par rapport à celui qui serait appliqué si le clip était défini sur 100 %.

**Disponibilité :** ActionScript 1.0 ; Flash Player 4

## Exemple

L'exemple suivant crée un clip appelé `box_mc` pendant l'exécution. L'API de dessin permet de dessiner un cadre dans cette occurrence, puis lorsque la souris survole cette zone, le clip est redimensionné à l'horizontale et à la verticale. Lorsque la souris quitte la zone de l'occurrence, les dimensions précédentes sont rétablies.

```
this.createEmptyMovieClip("box_mc", 1);
box_mc._x = 100;
box_mc._y = 100;
with (box_mc) {
  lineStyle(1, 0xCCCCCC);
  beginFill(0xEEEEEE);
  moveTo(0, 0);
  lineTo(80, 0);
  lineTo(80, 60);
  lineTo(0, 60);
  lineTo(0, 0);
  endFill();
};
box_mc.onRollOver = function() {
  this._x -= this._width/2;
  this._y -= this._height/2;
  this._xscale = 200;
  this._yscale = 200;
};
box_mc.onRollOut = function() {
  this._xscale = 100;
  this._yscale = 100;
  this._x += this._width/2;
  this._y += this._height/2;
};
```

## Voir également

_width (propriété MovieClip._width), _x (propriété MovieClip._x), _y (propriété MovieClip._y), _yscale (propriété MovieClip._yscale)

# _y (propriété MovieClip._y)

```
public _y : Number
```

Définit la coordonnée *y* d'un clip par rapport aux coordonnées locales du clip parent. Si un clip se trouve dans le scénario principal, son système de coordonnées se réfère alors au coin supérieur gauche de la scène : (0,0). Si le clip est imbriqué dans un autre clip subissant des transformations, le clip se trouve dans le système de coordonnées local du clip qui l'encadre. Ainsi, dans le cas d'un clip qui a effectué une rotation à 90 ° en sens anti-horaire, les enfants du clip héritent d'un système de coordonnées ayant effectué une rotation à 90 ° en sens anti-horaire. Les coordonnées du clip renvoient à la position du point d'alignement.

**Disponibilité :** ActionScript 1.0 ; Flash Player 3

## Exemple

L'exemple suivant associe un clip à l'identifiant de liaison `cursor_id` à un fichier SWF. Le clip est appelé `cursor_mc` et permet de remplacer le pointeur de souris par défaut. Le code ActionScript suivant définit les coordonnées actuelles de l'occurrence de clip en fonction de l'emplacement actuel du pointeur de la souris.

```
this.attachMovie("cursor_id", "cursor_mc", this.getNextHighestDepth(),
  {_x:_xmouse, _y:_ymouse});
Mouse.hide();
var mouseListener:Object = new Object();
mouseListener.onMouseMove = function() {
  cursor_mc._x = _xmouse;
  cursor_mc._y = _ymouse;
  updateAfterEvent();
};
Mouse.addListener(mouseListener);
```

La méthode `MovieClip.getNextHighestDepth()` utilisée dans cet exemple nécessite Flash Player 7 ou une version ultérieure. Si votre fichier SWF comprend un composant de la version 2, utilisez les composants de la version 2 de la classe DepthManager au lieu de la méthode `MovieClip.getNextHighestDepth()`.

## Voir également

`_x (propriété MovieClip._x)`, `_xscale (propriété MovieClip._xscale)`, `_yscale (propriété MovieClip._yscale)`

# _ymouse (propriété MovieClip._ymouse)

```
public _ymouse : Nombre [lecture seule]
```

Renvoie la coordonnée *y* de la position de la souris.

**Disponibilité :** ActionScript 1.0 ; Flash Player 5

## Exemple

L'exemple suivant renvoie les coordonnées actuelles *x* et *y* de la souris sur la scène (`_level0`) et par rapport à un clip appelé `my_mc` figurant également sur la scène.

```
this.createTextField("mouse_txt", this.getNextHighestDepth(), 0, 0, 150,
  66);
mouse_txt.html = true;
mouse_txt.multiline = true;
var row1_str:String = " \t<b>_xmouse\t</b><b>_ymouse</b>";
my_mc.onMouseMove = function() {
  mouse_txt.htmlText = "<textformat tabStops='[50,100]'>";
  mouse_txt.htmlText += row1_str;
  mouse_txt.htmlText += "<b>_level0</b>\t"+_xmouse+"\t"+_ymouse;
  mouse_txt.htmlText += "<b>my_mc</b>\t"+this._xmouse+"\t"+this._ymouse;
  mouse_txt.htmlText += "</textformat>";
};
```

La méthode `MovieClip.getNextHighestDepth()` utilisée dans cet exemple nécessite Flash Player 7 ou une version ultérieure. Si votre fichier SWF comprend un composant de la version 2, utilisez les composants de la version 2 de la classe DepthManager au lieu de la méthode `MovieClip.getNextHighestDepth()`.

## Voir également

`Souris`, `_xmouse (propriété MovieClip._xmouse)`

# _yscale (propriété MovieClip._yscale)

```
public _yscale : Number
```

Détermine le redimensionnement vertical du clip (*`percentage`*) tel qu'il est appliqué à partir du point d'alignement du clip. Le point d'alignement par défaut est (0,0).

Le redimensionnement du système de coordonnées local affecte les paramètres des propriétés `_x` et `_y`, définis en pixels. Par exemple, si le clip parent est redimensionné à 50 %, le paramétrage de la propriété `_x` déplace un objet dans le clip selon un nombre de pixels réduit de moitié par rapport à celui qui serait appliqué si le clip était défini sur 100 %.

**Disponibilité :** ActionScript 1.0 ; Flash Player 4

### Exemple

L'exemple suivant crée un clip appelé `box_mc` pendant l'exécution. L'API de dessin permet de dessiner un cadre dans cette occurrence, puis lorsque la souris survole cette zone, le clip est redimensionné à l'horizontale et à la verticale. Lorsque la souris quitte la zone de l'occurrence, les dimensions précédentes sont rétablies.

```
this.createEmptyMovieClip("box_mc", 1);
box_mc._x = 100;
box_mc._y = 100;
with (box_mc) {
  lineStyle(1, 0xCCCCCC);
  beginFill(0xEEEEEE);
  moveTo(0, 0);
  lineTo(80, 0);
  lineTo(80, 60);
  lineTo(0, 60);
  lineTo(0, 0);
  endFill();
};
box_mc.onRollOver = function() {
  this._x -= this._width/2;
  this._y -= this._height/2;
  this._xscale = 200;
  this._yscale = 200;
};
box_mc.onRollOut = function() {
  this._xscale = 100;
  this._yscale = 100;
  this._x += this._width/2;
  this._y += this._height/2;
};
```

### Voir également

`_height (propriété MovieClip._height)`, `_x (propriété MovieClip._x)`, `_xscale (propriété MovieClip._xscale)`, `_y (propriété MovieClip._y)`

# MovieClipLoader

```
Object
  |
  +-MovieClipLoader
```

```
public class MovieClipLoader
extends Object
```

Cette classe permet d'implémenter des rappels d'écouteur qui fournissent des informations d'état lors du chargement des fichiers SWF, JPEG, GIF et PNG dans les clips. Pour utiliser les fonctions de MovieClipLoader, utilisez `MovieClipLoader.loadClip()` à la place de `loadMovie()` ou `MovieClip.loadMovie()` pour charger des fichiers SWF.

Après avoir appelé la commande `MovieClipLoader.loadClip()` les événements suivants se produisent dans l'ordre répertorié :

- Lorsque les premiers octets du fichier téléchargé ont été écrits sur le disque, l'écouteur `MovieClipLoader.onLoadStart` est appelé.
- Si vous avez implémenté l'écouteur `MovieClipLoader.onLoadProgress`, il est appelé pendant le processus de chargement.
  *Remarque :* Vous pouvez appeler `MovieClipLoader.getProgress()` à tout moment au cours du processus de chargement.
- Lorsque le fichier téléchargé a été entièrement écrit sur le disque, l'écouteur `MovieClipLoader.onLoadComplete` est appelé.
- Une fois les actions de la première image du fichier téléchargé exécutées, l'écouteur `MovieClipLoader.onLoadInit` est appelé.

Lorsque `MovieClipLoader.onLoadInit` a été appelé, vous pouvez définir les propriétés, utiliser les méthodes ou encore interagir avec l'animation chargée.

Si le chargement du fichier ne s'effectue pas entièrement, l'écouteur `MovieClipLoader.onLoadError` est appelé.

**Disponibilité :** ActionScript 1.0 ; Flash Player 7

## Résumé des propriétés

*Propriétés héritées de la classe Object*

```
constructeur (propriété Object.constructor), __proto__ (Object.__proto__
property), prototype (Object.prototype, propriété), __resolve
(Object.__resolve, propriété)
```

## Résumé des événements

| Evénement | Description |
|---|---|
| `onLoadComplete = function([target_mc:MovieClip], [httpStatus:Number]) {}` | Appelé lorsque le fichier qui a été chargé avec `MovieClipLoader.loadClip()` a fini son téléchargement. |
| `onLoadError = function(target_mc:MovieClip, errorCode:String, [httpStatus:Number]) {}` | Appelé lorsque le chargement d'un fichier chargé avec `MovieClipLoader.loadClip()` a échoué. |
| `onLoadInit = function([target_mc:MovieClip]) {}` | Appelé une fois les actions de la première image du clip chargé exécutées. |
| `onLoadProgress = function([target_mc:MovieClip], loadedBytes:Number, totalBytes:Number) {}` | Appelé chaque fois que le contenu est écrit sur le disque dur au cours du processus de chargement (c'est-à-dire entre `MovieClipLoader.onLoadStart` et `MovieClipLoader.onLoadComplete`). |
| `onLoadStart = function([target_mc:MovieClip]) {}` | Appelé lorsqu'un appel de `MovieClipLoader.loadClip()` a commencé à charger un fichier. |

## Résumé des constructeurs

| Signature | Description |
|---|---|
| `MovieClipLoader()` | Crée un objet MovieClipLoader que vous pouvez utiliser pour implémenter un certain nombre d'écouteurs qui sont chargés de répondre aux événements pendant le téléchargement d'un fichier SWF, JPEG, GIF ou PNG. |

## Résumé de la méthode

| Modificateurs | Signature | Description |
|---|---|---|
| | `addListener(listener:Object) : Boolean` | Enregistre un objet de façon à ce qu'il reçoive des notifications lors d'un appel du gestionnaire d'événements `MovieClipLoader`. |
| | `getProgress(target:Object) : Object` | Renvoie le nombre d'octets chargés et le nombre total d'octets pour un fichier chargé à l'aide de `MovieClipLoader.loadClip()` ; pour les animations compressées, renvoie le nombre d'octets compressés. |
| | `loadClip(url:String, target:Object) : Boolean` | Charge un fichier SWF, JPEG, JPEG progressif, GIF non animé ou PNG dans un clip Flash Player lors de la lecture de l'animation d'origine. |
| | `removeListener(listener:Object) : Boolean` | Supprime l'écouteur utilisé pour recevoir la notification de l'appel du gestionnaire d'événements `MovieClipLoader`. |
| | `unloadClip(target:Object) : Boolean` | Supprime un clip chargé via `MovieClipLoader.loadClip()`. |

*Méthodes héritées de la classe Object*

`addProperty (méthode Object.addProperty)`, `hasOwnProperty (méthode Object.hasOwnProperty)`, `isPropertyEnumerable (méthode Object.isPropertyEnumerable)`, `isPrototypeOf (méthode Object.isPrototypeOf)`, `registerClass (méthode Object.registerClass)`, `toString (méthode Object.toString)`, `unwatch (méthode Object.unwatch)`, `valueOf (méthode Object.valueOf)`, `watch (méthode Object.watch)`

# addListener (méthode MovieClipLoader.addListener)

`public addListener(listener:Object) : Boolean`

Enregistre un objet de façon à ce qu'il reçoive des notifications lors d'un appel du gestionnaire d'événements `MovieClipLoader`.

**Disponibilité :** ActionScript 1.0 ; Flash Player 7

## Paramètres

*`listener`*`:Object` - Objet chargé de détecter la notification de rappel provenant des gestionnaires d'événements `MovieClipLoader`.

## Valeur renvoyée

`Boolean` - Valeur booléenne. Renvoie `true` si l'écouteur a été établi avec succès ; sinon `false`.

## Exemple

L'exemple suivant charge une image dans un clip appelé `image_mc`. L'occurrence de clip est pivotée et centrée sur la scène. Un trait entoure ensuite le périmètre de la scène et du clip.

```
this.createEmptyMovieClip("image_mc", this.getNextHighestDepth());
var mclListener:Object = new Object();
mclListener.onLoadInit = function(target_mc:MovieClip) {
  target_mc._x = Stage.width/2-target_mc._width/2;
  target_mc._y = Stage.height/2-target_mc._width/2;
  var w:Number = target_mc._width;
  var h:Number = target_mc._height;
  target_mc.lineStyle(4, 0x000000);
  target_mc.moveTo(0, 0);
  target_mc.lineTo(w, 0);
  target_mc.lineTo(w, h);
  target_mc.lineTo(0, h);
  target_mc.lineTo(0, 0);
  target_mc._rotation = 3;
};
var image_mcl:MovieClipLoader = new MovieClipLoader();
image_mcl.addListener(mclListener);
image_mcl.loadClip("http://www.helpexamples.com/flash/images/image1.jpg",
  image_mc);
```

Si votre fichier SWF comprend un composant de la version 2, utilisez les composants de la version 2 de la classe DepthManager au lieu de la méthode `MovieClip.getNextHighestDepth()`, utilisée dans cet exemple.

## Voir également

`onLoadComplete (écouteur d'événement MovieClipLoader.onLoadComplete)`, `onLoadError (écouteur d'événement MovieClipLoader.onLoadError)`, `onLoadInit (écouteur d'événement MovieClipLoader.onLoadInit)`, `onLoadProgress (écouteur d'événement MovieClipLoader.onLoadProgress)`, `onLoadStart (écouteur d'événement MovieClipLoader.onLoadStart)`, `removeListener (méthode MovieClipLoader.removeListener)`

# getProgress (méthode MovieClipLoader.getProgress)

`public getProgress(target:Object) : Object`

Renvoie le nombre d'octets chargés et le nombre total d'octets pour un fichier chargé à l'aide de `MovieClipLoader.loadClip()` ; pour les animations compressées, renvoie le nombre d'octets compressés. La méthode `getProgress` permet de demander ces informations de façon explicite, au lieu (ou en plus) d'écrire une fonction d'écouteur `MovieClipLoader.onLoadProgress`.

**Disponibilité :** ActionScript 1.0 ; Flash Player 7

## Paramètres

*`target`*`:Object` - Ffichier SWF, JPEG, GIF ou PNG chargé avec `MovieClipLoader.loadClip()`.

## Valeur renvoyée

`Object` - Objet ayant deux propriétés de type entier : `bytesLoaded` et `bytesTotal`.

## Exemple

L'exemple suivant illustre l'utilisation de la méthode `getProgress()`. Au lieu d'utiliser cette méthode, vous créerez généralement un objet d'écoute pour écouter l'événement `onLoadProgress`. Remarquez également que le premier appel synchrone de `getProgress()` peut renvoyer le nombre d'octets chargés et le nombre total d'octets du *conteneur*, mais pas les valeurs pour l'objet demandé à l'extérieur.

```
var container:MovieClip = this.createEmptyMovieClip("container",
  this.getNextHighestDepth());
var image:MovieClip = container.createEmptyMovieClip("image",
  container.getNextHighestDepth());

var mcLoader:MovieClipLoader = new MovieClipLoader();
var listener:Object = new Object();
listener.onLoadProgress = function(target:MovieClip, bytesLoaded:Number,
  bytesTotal:Number):Void {
  trace(target + ".onLoadProgress with " + bytesLoaded + " bytes of " +
  bytesTotal);
}
mcLoader.addListener(listener);
mcLoader.loadClip("http://www.w3.org/Icons/w3c_main.png", image);

var interval:Object = new Object();
interval.id = setInterval(checkProgress, 100, mcLoader, image, interval);
```

```
function checkProgress(mcLoader:MovieClipLoader, image:MovieClip,
  interval:Object):Void {
  trace(">> checking progress now with : " + interval.id);
  var progress:Object = mcLoader.getProgress(image);
  trace("bytesLoaded: " + progress.bytesLoaded + " bytesTotal: " +
  progress.bytesTotal);
  if(progress.bytesLoaded == progress.bytesTotal) {
    clearInterval(interval.id);
  }
}
```

Si votre fichier SWF comprend un composant de la version 2, utilisez les composants de la version 2 de la classe DepthManager au lieu de la méthode `MovieClip.getNextHighestDepth()`, utilisée dans cet exemple.

### Voir également

`loadClip (méthode MovieClipLoader.loadClip)`, `onLoadProgress (écouteur d'événement MovieClipLoader.onLoadProgress)`

## loadClip (méthode MovieClipLoader.loadClip)

```
public loadClip(url:String, target:Object) : Boolean
```

Charge un fichier SWF, JPEG, JPEG progressif, GIF non animé ou PNG dans un clip Flash Player lors de la lecture de l'animation d'origine. SI vous chargez un GIF animé, seule la première image est affichée. L'utilisation de cette méthode vous permet d'afficher plusieurs fichiers SWF simultanément, puis de basculer entre les fichiers SWF sans charger un autre document HTML.

L'utilisation de la méthode `loadClip()` au lieu de `loadMovie()` ou `MovieClip.loadMovie()` présente un certain nombre d'avantages. Les gestionnaires suivants sont implémentés par l'intermédiaire d'un objet écouteur. Vous activez l'écouteur en l'enregistrant avec la classe MovieClipLoader en utilisant `MovieClipLoader.addListener(listenerObject)`.

- Le gestionnaire `MovieClipLoader.onLoadStart` est invoqué lorsque le chargement commence.
- Le gestionnaire `MovieClipLoader.onLoadError` est appelé si le clip ne peut pas être chargé.
- Le gestionnaire `MovieClipLoader.onLoadProgress` est appelé lors de la progression du processus de chargement.
- Le gestionnaire `MovieClipLoader.onLoadComplete` est appelé lorsqu'un fichier termine son chargement, mais avant la mise à disposition des méthodes et des propriétés du clip qui vient d'être chargé. Ce gestionnaire est appelé avant le gestionnaire `onLoadInit`.

- Le gestionnaire `MovieClipLoader.onLoadInit` est appelé une fois les actions de la première image du clip exécutées, de manière à ce que vous puissiez commencer à manipuler le clip chargé. Ce gestionnaire est appelé après le gestionnaire `onLoadComplete`. Dans la plupart des cas, utilisez le gestionnaire `onLoadInit`.

Un fichier SWF ou une image chargé(e) dans un clip hérite des propriétés position, rotation et scale (échelle) du clip. Vous pouvez utiliser le chemin cible du clip pour cibler l'animation chargée.

Vous pouvez utiliser la méthode `loadClip()` pour charger un ou plusieurs fichiers dans un clip ou niveau unique ; les objets écouteurs MovieClipLoader reçoivent les paramètres d'occurrence de clip cible en cours de chargement. Sinon, vous pouvez également créer un objet MovieClipLoader différent pour chaque fichier que vous chargez.

Utilisez `MovieClipLoader.unloadClip()` pour supprimer des animations ou des images chargées à l'aide de cette méthode ou pour annuler une opération de chargement en cours.

`MovieClipLoader.getProgress()` et `MovieClipLoaderListener.onLoadProgress` ne signale pas les valeurs actuelles de `bytesLoaded` et de `bytesTotal` dans le lecteur de programmation lorsque les fichiers sont locaux. Lorsque vous utilisez la fonctionnalité Testeur de bande passante dans l'environnement de programmation, `MovieClipLoader.getProgress()` et `MovieClipLoaderListener.onLoadProgress` signalent le téléchargement à la vitesse de téléchargement réelle, et non selon la valeur de la bande passante réduite fournie par le testeur de bande passante.

Lorsque vous utilisez cette méthode, tenez compte du modèle de sécurité Flash Player.

Pour Flash Player 8 :

- Le chargement n'est pas permis si le clip appelant est dans le sandbox local avec système de fichier et que le clip chargé provient d'un sandbox réseau.
- Le chargement n'est pas autorisé si le fichier SWF appelant est sur un sandbox réseau et que le clip à charger est local.
- L'accès au sandbox réseau à partir d'un sandbox local approuvé ou de réseau local nécessite une autorisation du site au travers d'un fichier de régulation interdomaine.
- Les clips dans le sandbox local avec système de fichier ne peuvent pas inscrire des clips dans le sandbox local avec accès au réseau (l'inverse est également rendu impossible).

Flash Player 7 et versions ultérieures :

- Les sites peuvent autoriser un accès interdomaine à une ressource au travers d'un fichier de régulation interdomaines.
- La programmation entre les fichiers SWF est limitée par le domaine d'origine des fichiers SWF. Utilisez la méthode `System.security.allowDomain()` pour ajuster ces restrictions.

Pour plus d'informations, voir les sections suivantes :

- Chapitre 17, « Fonctionnement de la sécurité », dans *Formation à ActionScript 2.0 dans Flash*
- Le livre blanc sur la sécurité de Flash Player 8 disponible à l'adresse : http://www.macromedia.com/go/fp8_security
- Le livre blanc sur les API relatif à la sécurité de Flash Player 8 disponible à l'adresse : http://www.macromedia.com/go/fp8_security_apis

**Disponibilité :** ActionScript 1.0 ; Flash Player 7

## Paramètres

*url*:String - URL absolue ou relative du fichier SWF, JPEG, GIF ou PNG à charger. Un chemin relatif doit être relatif au fichier SWF au niveau 0. Les URL absolues doivent inclure la référence de protocole, telle que http:// ou file:///. Les noms de fichier ne doivent pas inclure les spécifications de lecteur de disque.

*target*:Object - Chemin cible d'un clip, ou entier spécifiant le niveau de Flash Player devant recevoir le fichier à charger. Le clip cible est remplacé par le fichier SWF chargé ou l'image.

## Valeur renvoyée

Boolean - Valeur booléenne. Renvoie true si la requête d'URL a été envoyée avec succès ; sinon false.

## Exemple

L'exemple suivant indique comment utiliser la méthode MovieClipLoader.loadClip() en créant un gestionnaire pour l'événement onLoadInit puis en exécutant la requête.

Le code suivant doit être placé directement dans une action d'image d'un scénario ou collé dans une classe développant l'objet MovieClip. Ce code attend également une image appelée YourImage.jpg dans le même répertoire que le fichier SWF compilé.

```
var container:MovieClip = createEmptyMovieClip("container",
  getNextHighestDepth());
var mcLoader:MovieClipLoader = new MovieClipLoader();
mcLoader.addListener(this);
mcLoader.loadClip("YourImage.jpg", container);

function onLoadInit(mc:MovieClip) {
  trace("onLoadInit: " + mc);
}
```

Si votre fichier SWF comprend un composant de la version 2, utilisez les composants de la version 2 de la classe DepthManager au lieu de la méthode MovieClip.getNextHighestDepth(), utilisée dans cet exemple.

### Voir également

onLoadInit (écouteur d'événement MovieClipLoader.onLoadInit)

## MovieClipLoader, constructeur

```
public MovieClipLoader()
```

Crée un objet MovieClipLoader que vous pouvez utiliser pour implémenter un certain nombre d'écouteurs qui sont chargés de répondre aux événements pendant le téléchargement d'un fichier SWF, JPEG, GIF ou PNG.

**Disponibilité :** ActionScript 1.0 ; Flash Player 7

### Exemple

Voir la section `MovieClipLoader.loadClip()`.

### Voir également

addListener (méthode MovieClipLoader.addListener), loadClip (méthode MovieClipLoader.loadClip)

## onLoadComplete (écouteur d'événement MovieClipLoader.onLoadComplete)

onLoadComplete = function([target_mc:MovieClip], [httpStatus:Number]) {}

Appelé lorsque le fichier qui a été chargé avec `MovieClipLoader.loadClip()` a fini son téléchargement. Appelle cet écouteur sur un objet d'écoute que vous ajoutez à l'aide de `MovieClipLoader.addListener()`. L'écouteur d'événement `onLoadComplete` est transmis par Flash Player à votre code, mais il n'est pas nécessaire d'implémenter tous les paramètres dans la fonction d'écouteur. La valeur `target_mc` identifie le clip pour lequel cet appel est effectué. Cette identification est particulièrement utile si plusieurs fichiers sont chargés avec le même jeu d'écouteurs.

Dans Flash Player 8, cet écouteur peut renvoyer un code d'état HTTP. Si Flash Player ne peut pas obtenir le code d'état du serveur, ou si Flash Player ne peut pas communiquer avec le serveur, la valeur par défaut (0) est transmise à votre code ActionScript. Une valeur de 0 peut être générée dans n'importe quel lecteur (par exemple, si une URL mal formulée est requise), et une valeur de 0 est toujours générée par le module Flash Player lorsqu'il est exécuté dans les navigateurs suivants, qui ne peuvent pas transmettre les codes d'état HTTP du serveur à Flash Player : Netscape, Mozilla, Safari, Opera et Internet Explorer pour Macintosh.

Il est important de comprendre la différence entre `MovieClipLoader.onLoadComplete` et `MovieClipLoader.onLoadInit`. L'événement `onLoadComplete` est appelé lorsque le chargement du fichier SWF, JPEG, GIF ou PNG est terminé, mais avant l'initialisation de l'application. À ce stade, il est impossible d'accéder aux méthodes et propriétés du clip chargé ; c'est la raison pour laquelle vous ne pouvez pas appeler de fonction, vous déplacer vers une image spécifique, etc. Dans la plupart des situations, il est préférable d'utiliser l'événement `onLoadInit`, qui est appelé une fois le contenu chargé et entièrement initialisé.

**Disponibilité :** ActionScript 1.0 ; Flash Player 7

## Paramètres

*`target_mc`*`:MovieClip` [facultatif] - Clip chargé par la méthode `MovieClipLoader.loadClip()`.

*`httpStatus`*`:Number` [facultatif] - (Flash Player 8 seulement) Code d'état HTTP renvoyé par le serveur. Par exemple, un code d'état de 404 indique que le serveur n'a trouvé aucune correspondance pour l'URL requise. Pour plus d'informations sur les codes d'état HTTP, consultez les sections 10.4 et 10.5 de la spécification HTTP à l'adresse ftp://ftp.isi.edu/in-notes/rfc2616.txt.

## Exemple

L'exemple suivant crée un clip, une nouvelle occurrence `MovieClipLoader` et un écouteur d'événement anonyme qui écoute l'événement `onLoadComplete` mais attend qu'un événement `onLoadInit` interagisse avec les propriétés de l'élément chargé.

```
var loadListener:Object = new Object();

loadListener.onLoadComplete = function(target_mc:MovieClip,
  httpStatus:Number):Void {
  trace(">> loadListener.onLoadComplete()");
  trace(">> ============================");
  trace(">> target_mc._width: " + target_mc._width); // 0
  trace(">> httpStatus: " + httpStatus);
}

loadListener.onLoadInit = function(target_mc:MovieClip):Void {
  trace(">> loadListener.onLoadInit()");
  trace(">> ==========================");
  trace(">> target_mc._width: " + target_mc._width); // 315
}

var mcLoader:MovieClipLoader = new MovieClipLoader();
mcLoader.addListener(loadListener);
```

```
var mc:MovieClip = this.createEmptyMovieClip("mc",
  this.getNextHighestDepth());
mcLoader.loadClip("http://www.w3.org/Icons/w3c_main.png", mc);
```

Si votre fichier SWF comprend un composant de la version 2, utilisez les composants de la version 2 de la classe DepthManager au lieu de la méthode `MovieClip.getNextHighestDepth()`, utilisée dans cet exemple.

### Voir également

`addListener (méthode MovieClipLoader.addListener)`, `loadClip (méthode MovieClipLoader.loadClip)`, `onLoadStart (écouteur d'événement MovieClipLoader.onLoadStart)`, `onLoadError (écouteur d'événement MovieClipLoader.onLoadError)`, `onLoadInit (écouteur d'événement MovieClipLoader.onLoadInit)`

## onLoadError (écouteur d'événement MovieClipLoader.onLoadError)

```
onLoadError = function(target_mc:MovieClip, errorCode:String,
  [httpStatus:Number]) {}
```

Appelé lorsque le chargement d'un fichier chargé avec `MovieClipLoader.loadClip()` a échoué. Cet écouteur peut être invoqué pour différentes raisons ; par exemple, si le serveur est indisponible, le fichier introuvable ou si une violation de sécurité se produit.

Appelle cet écouteur sur un objet d'écoute que vous ajoutez à l'aide de `MovieClipLoader.addListener()`.

La valeur de `target_mc` identifie le clip pour lequel cet appel est effectué. Ce paramètre est particulièrement utile si vous chargez plusieurs fichiers avec le même jeu d'écouteurs.

Pour le paramètre `errorCode`, la chaîne `"URLNotFound"` est renvoyée si l'écouteur `MovieClipLoader.onLoadStart` ou `MovieClipLoader.onLoadComplete` n'a pas été appelé ; par exemple si le serveur est indisponible ou si le fichier est introuvable. La chaîne `"LoadNeverCompleted"` est renvoyée si l'écouteur `MovieClipLoader.onLoadStart` a été appelé, mais que l'écouteur `MovieClipLoader.onLoadComplete` n'a pas été appelé ; par exemple si le chargement a été interrompu en raison d'un encombrement du serveur, d'une panne de serveur, etc.

Dans Flash Player 8, cet écouteur peut renvoyer un code d'état HTTP dans le paramètre `httpStatus`. Si Flash Player ne peut pas obtenir un code d'état du serveur, ou si Flash Player ne peut pas communiquer avec le serveur, la valeur par défaut (0) est transmise à votre code ActionScript. Une valeur de 0 peut être générée dans n'importe quel lecteur (par exemple, si une URL mal formulée est requise), et une valeur de 0 est toujours générée par le module Flash Player lorsqu'il est exécuté dans les navigateurs suivants, qui ne peuvent pas transmettre les codes d'état HTTP du serveur vers Flash Player : Netscape, Mozilla, Safari, Opera, et Internet Explorer pour Macintosh. Une valeur de 0 peut également être générée si le lecteur n'a pas essayé d'effectuer la demande d'URL pour réaliser le chargement. Ceci peut se produire parce que la requête viole les règles sandbox de sécurité pour le fichier SWF.

**Disponibilité :** ActionScript 1.0 ; Flash Player 7

## Paramètres

*`target_mc`*`:MovieClip` - Clip chargé par la méthode `MovieClipLoader.loadClip()`.

*`errorCode`*`:String` - Chaîne qui précise les raisons de l'échec, soit `"URLNotFound"` soit `"LoadNeverCompleted"`.

*`httpStatus`*`:Number` [facultatif] - (Flash Player 8 seulement) Code d'état HTTP renvoyé par le serveur. Par exemple, un code d'état de 404 indique que le serveur n'a trouvé aucune correspondance pour l'URL requise. Pour plus d'informations sur les codes d'état HTTP, consultez les sections 10.4 et 10.5 de la spécification HTTP à l'adresse ftp://ftp.isi.edu/in-notes/rfc2616.txt.

## Exemple

L'exemple suivant affiche des informations dans le panneau de sortie lorsqu'une image ne se charge pas. L'URL utilisée dans cet exemple est fictive ; remplacez-la par une URL valide.

```
var loadListener:Object = new Object();

loadListener.onLoadError = function(target_mc:MovieClip, errorCode:String,
  httpStatus:Number) {
  trace(">> loadListener.onLoadError()");
  trace(">> ==========================");
  trace(">> errorCode: " + errorCode);
  trace(">> httpStatus: " + httpStatus);
}

var mcLoader:MovieClipLoader = new MovieClipLoader();
mcLoader.addListener(loadListener);

var mc:MovieClip = this.createEmptyMovieClip("mc",
  this.getNextHighestDepth());
mcLoader.loadClip("http://www.fakedomain.com/images/bad_hair_day.jpg", mc);
```

Si votre fichier SWF comprend un composant de la version 2, utilisez les composants de la version 2 de la classe DepthManager au lieu de la méthode `MovieClip.getNextHighestDepth()`, utilisée dans cet exemple.

### Voir également

`addListener (méthode MovieClipLoader.addListener)`, `loadClip (méthode MovieClipLoader.loadClip)`, `onLoadStart (écouteur d'événement MovieClipLoader.onLoadStart)`, `onLoadComplete (écouteur d'événement MovieClipLoader.onLoadComplete)`

## onLoadInit (écouteur d'événement MovieClipLoader.onLoadInit)

```
onLoadInit = function([target_mc:MovieClip]) {}
```

Appelé une fois les actions de la première image du clip chargé exécutées. Lorsque cet écouteur a été appelé, vous pouvez définir les propriétés, utiliser les méthodes ou encore interagir avec l'animation chargée. Appelle cet écouteur sur un objet d'écoute que vous ajoutez à l'aide de `MovieClipLoader.addListener()`.

La valeur `target_mc` identifie le clip pour lequel cet appel est effectué. Ce paramètre est particulièrement utile si vous chargez plusieurs fichiers avec le même jeu d'écouteurs.

**Disponibilité :** ActionScript 1.0 ; Flash Player 7

### Paramètres

*`target_mc`*`:MovieClip` [facultatif] - Clip chargé par la méthode `MovieClipLoader.loadClip()`.

### Exemple

L'exemple suivant charge une image dans une occurrence de clip appelée `image_mc`. Les événements `onLoadInit` et `onLoadComplete` permettent de déterminer le temps de chargement de l'image. Les informations s'affichent dans le champ texte appelé `timer_txt`.

```
this.createEmptyMovieClip("image_mc", this.getNextHighestDepth());
var mclListener:Object = new Object();
mclListener.onLoadStart = function(target_mc:MovieClip) {
  target_mc.startTimer = getTimer();
};
mclListener.onLoadComplete = function(target_mc:MovieClip) {
  target_mc.completeTimer = getTimer();
};
mclListener.onLoadInit = function(target_mc:MovieClip) {
  var timerMS:Number = target_mc.completeTimer-target_mc.startTimer;
```

```
  target_mc.createTextField("timer_txt", target_mc.getNextHighestDepth(),
  0, target_mc._height,
target_mc._width, 22);
  target_mc.timer_txt.text = "loaded in "+timerMS+" ms.";
};
var image_mcl:MovieClipLoader = new MovieClipLoader();
image_mcl.addListener(mclListener);
image_mcl.loadClip("http://www.helpexamples.com/flash/images/image1.jpg",
  image_mc);
```

L'exemple suivant permet de s'assurer qu'une animation a été chargée dans un clip créé lors de l'exécution. L'URL utilisée dans cet exemple n'est là que pour faire une démonstration ; remplacez-la par une URL valide.

```
this.createEmptyMovieClip("tester_mc", 1);
var mclListener:Object = new Object();
mclListener.onLoadInit = function(target_mc:MovieClip) {
  trace("movie loaded");
}
var image_mcl:MovieClipLoader = new MovieClipLoader();
image_mcl.addListener(mclListener);
image_mcl.loadClip("http://www.yourserver.com/your_movie.swf", tester_mc);
```

Si votre fichier SWF comprend un composant de la version 2, utilisez les composants de la version 2 de la classe DepthManager au lieu de la méthode `MovieClip.getNextHighestDepth()`, utilisée dans cet exemple.

### Voir également

`addListener (méthode MovieClipLoader.addListener)`, `loadClip (méthode MovieClipLoader.loadClip)`, `onLoadStart (écouteur d'événement MovieClipLoader.onLoadStart)`

## onLoadProgress (écouteur d'événement MovieClipLoader.onLoadProgress)

```
onLoadProgress = function([target_mc:MovieClip], loadedBytes:Number,
  totalBytes:Number) {}
```

Ecouteur : appelé à chaque fois que le contenu est écrit sur le disque dur au cours du processus de chargement (c'est-à-dire, entre `MovieClipLoader.onLoadStart` et `MovieClipLoader.onLoadComplete`). Appelle cet écouteur sur un objet d'écoute que vous ajoutez à l'aide de `MovieClipLoader.addListener()`. Vous pouvez utiliser cette méthode pour afficher les informations sur la progression du téléchargement, à l'aide des paramètres `loadedBytes` et `totalBytes`.

La valeur `target_mc` identifie le clip pour lequel cet appel est effectué. Cela est particulièrement utile lorsque vous chargez plusieurs fichiers avec le même jeu d'écouteurs.

*Remarque :* Si vous tentez d'utiliser `onLoadProgress` en mode test sur un fichier local résidant sur votre disque dur, il ne fonctionne pas correctement car, en mode test, Flash Player charge intégralement les fichiers locaux.

**Disponibilité :** ActionScript 1.0 ; Flash Player 7

## Paramètres

*`target_mc`*`:MovieClip` [facultatif] - Clip chargé par la méthode `MovieClipLoader.loadClip()`.

*`loadedBytes`*`:Number` - Nombre d'octets chargés lorsque l'écouteur a été appelé.

*`totalBytes`*`:Number` - Nombre total d'octets dans le fichier chargé.

## Exemple

L'exemple suivant crée un clip, une nouvelle occurrence `MovieClipLoader` et un écouteur d'événement anonyme. Il affiche périodiquement la progression d'un chargement et envoie un signal lorsque le chargement est terminé et que l'actif est disponible pour ActionScript.

```
var container:MovieClip = this.createEmptyMovieClip("container",
  this.getNextHighestDepth());
var mcLoader:MovieClipLoader = new MovieClipLoader();
var listener:Object = new Object();
listener.onLoadProgress = function(target:MovieClip, bytesLoaded:Number,
  bytesTotal:Number):Void {
  trace(target + ".onLoadProgress with " + bytesLoaded + " bytes of " +
  bytesTotal);
}
listener.onLoadInit = function(target:MovieClip):Void {
  trace(target + ".onLoadInit");
}
mcLoader.addListener(listener);
mcLoader.loadClip("http://www.w3.org/Icons/w3c_main.png", container);
```

Si votre fichier SWF comprend un composant de la version 2, utilisez les composants de la version 2 de la classe DepthManager au lieu de la méthode `MovieClip.getNextHighestDepth()`, utilisée dans cet exemple.

## Voir également

`addListener (méthode MovieClipLoader.addListener)`, `loadClip (méthode MovieClipLoader.loadClip)`, `getProgress (méthode MovieClipLoader.getProgress)`

# onLoadStart (écouteur d'événement MovieClipLoader.onLoadStart)

```
onLoadStart = function([target_mc:MovieClip]) {}
```

Appelé lorsqu'un appel à `MovieClipLoader.loadClip()` a commencé à charger un fichier. Appelle cet écouteur sur un objet d'écoute que vous ajoutez à l'aide de `MovieClipLoader.addListener()`.

La valeur `target_mc` identifie le clip pour lequel cet appel est effectué. Ce paramètre est particulièrement utile si vous chargez plusieurs fichiers avec le même jeu d'écouteurs.

**Disponibilité :** ActionScript 1.0 ; Flash Player 7

## Paramètres

*`target_mc`*`:MovieClip` [facultatif] - Clip chargé par la méthode `MovieClipLoader.loadClip()`.

## Exemple

L'exemple suivant charge une image dans une occurrence de clip appelée `image_mc`. Les événements `onLoadInit` et `onLoadComplete` permettent de déterminer le temps de chargement de l'image. Les informations s'affichent dans le champ texte appelé `timer_txt`.

```
this.createEmptyMovieClip("image_mc", this.getNextHighestDepth());
var mclListener:Object = new Object();
mclListener.onLoadStart = function(target_mc:MovieClip) {
  target_mc.startTimer = getTimer();
};
mclListener.onLoadComplete = function(target_mc:MovieClip) {
  target_mc.completeTimer = getTimer();
};
mclListener.onLoadInit = function(target_mc:MovieClip) {
  var timerMS:Number = target_mc.completeTimer-target_mc.startTimer;
  target_mc.createTextField("timer_txt", target_mc.getNextHighestDepth(),
  0, target_mc._height,
target_mc._width, 22);
  target_mc.timer_txt.text = "loaded in "+timerMS+" ms.";
};
var image_mcl:MovieClipLoader = new MovieClipLoader();
image_mcl.addListener(mclListener);
image_mcl.loadClip("http://www.helpexamples.com/flash/images/image1.jpg",
  image_mc);
```

Si votre fichier SWF comprend un composant de la version 2, utilisez les composants de la version 2 de la classe DepthManager au lieu de la méthode `MovieClip.getNextHighestDepth()`, utilisée dans cet exemple.

### Voir également

addListener (méthode MovieClipLoader.addListener), loadClip (méthode MovieClipLoader.loadClip), onLoadError (écouteur d'événement MovieClipLoader.onLoadError), onLoadInit (écouteur d'événement MovieClipLoader.onLoadInit), onLoadComplete (écouteur d'événement MovieClipLoader.onLoadComplete)

## removeListener (méthode MovieClipLoader.removeListener)

```
public removeListener(listener:Object) : Boolean
```

Supprime l'écouteur utilisé pour recevoir la notification de l'appel du gestionnaire d'événements `MovieClipLoader`. Aucun autre message en cours de chargement ne sera reçu.

**Disponibilité :** ActionScript 1.0 ; Flash Player 7

### Paramètres

*`listener`*`:Object` - Objet écouteur ajouté à l'aide de `MovieClipLoader.addListener()`.

### Valeur renvoyée

`Boolean` - Valeur booléenne. Renvoie `true` si l'écouteur a été supprimé avec succès ; sinon `false`.

### Exemple

L'exemple suivant charge une image dans un clip et permet à l'utilisateur de démarrer et d'arrêter le processus de chargement à l'aide de deux boutons appelés `start_button` et `stop_button`. Lorsque l'utilisateur lance ou arrête la progression, des informations s'affichent dans le panneau de sortie.

```
this.createEmptyMovieClip("image_mc", this.getNextHighestDepth());
var mclListener:Object = new Object();
mclListener.onLoadStart = function(target_mc:MovieClip) {
  trace("\t onLoadStart");
};
mclListener.onLoadComplete = function(target_mc:MovieClip) {
  trace("\t onLoadComplete");
};
mclListener.onLoadError = function(target_mc:MovieClip, errorCode:String) {
  trace("\t onLoadError: "+errorCode);
};
mclListener.onLoadInit = function(target_mc:MovieClip) {
  trace("\t onLoadInit");
  start_button.enabled = true;
```

```
  stop_button.enabled = false;
};
var image_mcl:MovieClipLoader = new MovieClipLoader();
//
start_button.clickHandler = function() {
  trace("Starting...");
  start_button.enabled = false;
  stop_button.enabled = true;
  //
  image_mcl.addListener(mclListener);
  image_mcl.loadClip("http://www.helpexamples.com/flash/images/
  image1.jpg", image_mc);
};
stop_button.clickHandler = function() {
  trace("Stopping...");
  start_button.enabled = true;
  stop_button.enabled = false;
  //
  image_mcl.removeListener(mclListener);
};
stop_button.enabled = false;
```

Si votre fichier SWF comprend un composant de la version 2, utilisez les composants de la version 2 de la classe DepthManager au lieu de la méthode `MovieClip.getNextHighestDepth()`, utilisée dans cet exemple.

### Voir également

`addListener (méthode MovieClipLoader.addListener)`

## unloadClip (méthode MovieClipLoader.unloadClip)

```
public unloadClip(target:Object) : Boolean
```

Supprime un clip chargé via `MovieClipLoader.loadClip()`. Si vous appelez cette commande lors du chargement d'une animation, `MovieClipLoader.onLoadError` est appelé.

**Disponibilité :** ActionScript 1.0 ; Flash Player 7

### Paramètres

*`target`*`:Object` - Chaîne ou entier transmis à l'appel correspondant à `my_mcl.loadClip()`.

### Valeur renvoyée

`Boolean` - Valeur booléenne. Renvoie `true` si le clip a été supprimé avec succès ; sinon `false`.

### Exemple

L'exemple suivant charge une image dans un clip appelé `image_mc`. Lorsque vous cliquez sur le clip, ce dernier est supprimé et des informations s'affichent dans le panneau de sortie.

```
this.createEmptyMovieClip("image_mc", this.getNextHighestDepth());
var mclListener:Object = new Object();
mclListener.onLoadInit = function(target_mc:MovieClip) {
  target_mc._x = 100;
  target_mc._y = 100;
  target_mc.onRelease = function() {
  trace("Unloading clip...");
  trace("\t name: "+target_mc._name);
  trace("\t url: "+target_mc._url);
  image_mcl.unloadClip(target_mc);
  };
};
var image_mcl:MovieClipLoader = new MovieClipLoader();
image_mcl.addListener(mclListener);
image_mcl.loadClip("http://www.helpexamples.com/flash/images/image1.jpg",
  image_mc);
```

Si votre fichier SWF comprend un composant de la version 2, utilisez les composants de la version 2 de la classe DepthManager au lieu de la méthode `MovieClip.getNextHighestDepth()`, utilisée dans cet exemple.

### Voir également

`loadClip (méthode MovieClipLoader.loadClip)`, `onLoadError (écouteur d'événement MovieClipLoader.onLoadError)`

# NetConnection

```
Object
  |
  +-NetConnection
```

`public dynamic class` *NetConnection*
`extends Object`

La classe NetConnection permet de lire des fichiers FLV en flux continu à partir d'un lecteur local ou d'une adresse HTTP.

*Remarque :* Cette classe est également prise en charge dans Flash Player 6 lorsqu'elle est utilisée de concert avec Flash Communication Server. Pour plus d'informations, consultez la documentation relative à Flash Communication Server.

**Disponibilité :** ActionScript 1.0 ; Flash Player 7

### Résumé des propriétés

*Propriétés héritées de la classe Object*

| |
|---|
| constructeur (propriété Object.constructor), __proto__ (Object.__proto__ property), prototype (Object.prototype, propriété), __resolve (Object.__resolve, propriété) |

### Résumé des constructeurs

| Signature | Description |
|---|---|
| `NetConnection()` | Crée un objet NetConnection que vous pouvez utiliser en conjonction avec un objet NetStream pour lire les fichiers vidéo locaux en diffusion continue (FLV). |

### Résumé de la méthode

| Modificateurs | Signature | Description |
|---|---|---|
| | `connect(targetURI:String) : Boolean` | Ouvre une connexion locale permettant de lire les fichiers vidéo (FLV) à partir d'une adresse HTTP ou du système local de fichiers. |

*Méthodes héritées de la classe Object*

| |
|---|
| addProperty (méthode Object.addProperty), hasOwnProperty (méthode Object.hasOwnProperty), isPropertyEnumerable (méthode Object.isPropertyEnumerable), isPrototypeOf (méthode Object.isPrototypeOf), registerClass (méthode Object.registerClass), toString (méthode Object.toString), unwatch (méthode Object.unwatch), valueOf (méthode Object.valueOf), watch (méthode Object.watch) |

## connect (méthode NetConnection.connect)

`public connect(targetURI:String) : Boolean`

Ouvre une connexion locale permettant de lire les fichiers vidéo (FLV) à partir d'une adresse HTTP ou du système local de fichiers.

Lorsque vous utilisez cette méthode, prenez en considération le modèle de sécurité de Flash Player et les considérations de sécurité suivantes :

- Par défaut l'accès entre les sandbox est refusé. Le site peut autoriser l'accès à une ressource via un fichier de régulation interdomaine.
- Un site peut refuser l'accès à une ressource en ajoutant une logique d'application ActionScript côté serveur dans Flash Communication Server.

- Pour Flash Player 8, la méthode NetConnection.connect() n'est pas autorisée si le fichier SWF appelant est dans le sandbox local avec système de fichier.

Pour plus d'informations, voir les sections suivantes :

- Chapitre 17, « Fonctionnement de la sécurité », dans *Formation à ActionScript 2.0 dans Flash*
- Le livre blanc sur la sécurité de Flash Player 8 disponible à l'adresse : http://www.macromedia.com/go/fp8_security
- Le livre blanc sur les API relatif à la sécurité de Flash Player 8 disponible à l'adresse : http://www.macromedia.com/go/fp8_security_apis

**Disponibilité :** ActionScript 1.0 ; Flash Player 7

## Paramètres

*`targetURI`*`:String` - Pour ce paramètre, vous devez transmettre `null`.

## Valeur renvoyée

`Boolean` - Si la valeur est `false`, la connexion a échoué et n'est pas utilisable. Si la valeur est `true`, la connexion n'a pas échoué lorsque la méthode `connect()` est appelée, mais cela ne garantit pas le succès de l'opération.

## Exemple

L'exemple suivant ouvre une connexion pour lire le fichier video2.flv. Sélectionnez Nouvelle vidéo dans le menu d'options du panneau Bibliothèque pour créer un nouvel objet vidéo et appelez cette occurrence `my_video`.

```
var connection_nc:NetConnection = new NetConnection();
connection_nc.connect(null);
var stream_ns:NetStream = new NetStream(connection_nc);
my_video.attachVideo(stream_ns);
stream_ns.play("video2.flv");
```

## Voir également

`NetStream`

## NetConnection, constructeur

```
public NetConnection()
```

Crée un objet NetConnection que vous pouvez utiliser en conjonction avec un objet NetStream pour lire les fichiers vidéo locaux en diffusion continue (FLV). Après avoir créé l'objet NetConnection, utilisez `NetConnection.connect()` pour établir la véritable connexion.

La lecture des fichiers FLV externes offre plusieurs avantages par rapport à l'intégration de vidéo dans un document Flash : performances et gestion de la mémoire améliorées, indépendance des cadences vidéo et Flash. La classe NetConnection permet de lire des fichiers FLV en flux continu à partir d'un lecteur local ou d'une adresse HTTP.

**Disponibilité :** ActionScript 1.0 ; Flash Player 7

### Exemple

Consultez l'exemple relatif à `NetConnection.connect()`.

### Voir également

`connect (méthode NetConnection.connect)`, `attachVideo (méthode Video.attachVideo)`, `NetStream`

# NetStream

```
Object
  |
  +-NetStream
```

```
public dynamic class NetStream
extends Object
```

La classe NetStream fournit des méthodes et des propriétés permettant de lire des fichiers Flash Video (FLV) à partir du système de fichiers local ou d'une adresse HTTP. Vous utilisez un objet NetStream pour transmettre de la vidéo en continu via un objet NetConnection. La lecture des fichiers FLV externes offre plusieurs avantages par rapport à l'intégration de vidéo dans un document Flash : performances et gestion de la mémoire améliorées, indépendance des cadences vidéo et Flash. Cette classe fournit un certain nombre de méthodes et de propriétés que vous pouvez utiliser pour suivre la progression du chargement et de la lecture du fichier, et pour permettre à l'utilisateur de contrôler la lecture (arrêt, pause, etc.).

**Disponibilité :** ActionScript 1.0 ; Flash Player 7

## Résumé des propriétés

| Modificateurs | Propriété | Description |
|---|---|---|
| | `bufferLength:Number` [lecture seule] | Nombre de secondes de données enregistrées dans la mémoire tampon. |
| | `bufferTime:Number` [lecture seule] | Nombre de secondes affectées à la mémoire tampon par `NetStream.setBufferTime()`. |
| | `bytesLoaded:Number` [lecture seule] | Nombre d'octets de données ayant été chargés dans le lecteur. |
| | `bytesTotal:Number` [lecture seule] | Taille totale, en octets, du fichier chargé dans le lecteur. |
| | `currentFps:Number` [lecture seule] | Nombre d'images affichées par seconde. |
| | `time:Number` [lecture seule] | Position de la tête de lecture, en secondes. |

*Propriétés héritées de la classe Object*

`constructeur (propriété Object.constructor), __proto__ (Object.__proto__ property), prototype (Object.prototype, propriété), __resolve (Object.__resolve, propriété)`

## Résumé des événements

| Evénement | Description |
|---|---|
| `onMetaData = function(infoObject:Object) {}` | Invoqué lorsque Flash Player reçoit une description intégrée dans le fichier FLV lu. |
| `onStatus = function(infoObject:Object) {}` | Appelé à chaque changement d'état ou chaque fois qu'une erreur est publiée pour l'objet NetStream. |

## Résumé des constructeurs

| Signature | Description |
|---|---|
| `NetStream(connection:NetConnection)` | Crée un flux de diffusion en continu qui permet de lire des fichiers FLV à l'aide de l'objet NetConnection spécifié. |

### Résumé de la méthode

| Modificateurs | Signature | Description |
|---|---|---|
| | `close() : Void` | Arrête la lecture des données du flux de diffusion en continu, définit la propriété `NetStream.time` sur 0 et met le flux de diffusion en continu à disposition. |
| | `pause([flag:Boolean]) : Void` | Interrompt ou reprend la lecture d'un flux. |
| | `play(name:Object, start:Number, len:Number, reset:Object) : Void` | Commence la lecture d'un fichier vidéo externe (FLV). |
| | `seek(offset:Number) : Void` | Recherche l'image-clé la plus proche du nombre de secondes spécifié à partir du début du flux. |
| | `setBufferTime(bufferTime:Number) : Void` | Spécifie la durée de la mise en mémoire tampon des messages avant de commencer à afficher le flux. |

*Méthodes héritées de la classe Object*

`addProperty (méthode Object.addProperty), hasOwnProperty (méthode Object.hasOwnProperty), isPropertyEnumerable (méthode Object.isPropertyEnumerable), isPrototypeOf (méthode Object.isPrototypeOf), registerClass (méthode Object.registerClass), toString (méthode Object.toString), unwatch (méthode Object.unwatch), valueOf (méthode Object.valueOf), watch (méthode Object.watch)`

## bufferLength (propriété NetStream.bufferLength)

`public bufferLength : Nombre [lecture seule]`

Le nombre de secondes de données enregistrées dans la mémoire tampon. Vous pouvez utiliser cette propriété conjointement avec `NetStream.bufferTime` pour estimer le niveau de remplissage de la mémoire tampon, par exemple, pour permettre à un utilisateur qui attend la fin du chargement des données dans la mémoire tampon de consulter le compte rendu.

**Disponibilité :** ActionScript 1.0 ; Flash Player 7

### Exemple

L'exemple suivant crée de façon dynamique un champ texte qui donne des informations sur le nombre de secondes d'affichages en mémoire tampon. Ce champ donne également la longueur de tampon définie pour la vidéo et le pourcentage de cette valeur qui est utilisée.

```
this.createTextField("buffer_txt", this.getNextHighestDepth(), 10, 10, 300,
  22);
buffer_txt.html = true;

var connection_nc:NetConnection = new NetConnection();
connection_nc.connect(null);
var stream_ns:NetStream = new NetStream(connection_nc);
stream_ns.setBufferTime(3);
my_video.attachVideo(stream_ns);
stream_ns.play("video1.flv");

var buffer_interval:Number = setInterval(checkBufferTime, 100, stream_ns);
function checkBufferTime(my_ns:NetStream):Void {
  var bufferPct:Number = Math.min(Math.round(my_ns.bufferLength/
  my_ns.bufferTime 100), 100);
  var output_str:String = "<textformat tabStops='[100,200]'>";
  output_str += "Length: "+my_ns.bufferLength+"\t"+"Time:
  "+my_ns.bufferTime+"\t"+"Buffer:"+bufferPct+"%";
  output_str += "</textformat>";
  buffer_txt.htmlText = output_str;
}
```

Si votre fichier SWF comprend un composant de la version 2, utilisez les composants de la version 2 de la classe DepthManager au lieu de la méthode `MovieClip.getNextHighestDepth()`, utilisée dans cet exemple.

### Voir également

`bufferTime (propriété NetStream.bufferTime)`, `bytesLoaded (propriété NetStream.bytesLoaded)`

## bufferTime (propriété NetStream.bufferTime)

```
public bufferTime : Nombre [lecture seule]
```

Le nombre de secondes affectées à la mémoire tampon par `NetStream.setBufferTime()`. La valeur par défaut est 0,1 (un dixième de seconde). Pour déterminer le nombre de secondes actuellement dans la mémoire tampon, utilisez `NetStream.bufferLength`.

**Disponibilité :** ActionScript 1.0 ; Flash Player 7

### Exemple

L'exemple suivant crée de façon dynamique un champ texte qui donne des informations sur le nombre de secondes actuellement en mémoire tampon. Ce champ donne également la longueur de tampon définie pour la vidéo et le pourcentage de cette valeur qui est utilisée.

```
this.createTextField("buffer_txt", this.getNextHighestDepth(), 10, 10, 300,
  22);
```

```
buffer_txt.html = true;

var connection_nc:NetConnection = new NetConnection();
connection_nc.connect(null);
var stream_ns:NetStream = new NetStream(connection_nc);
stream_ns.setBufferTime(3);
my_video.attachVideo(stream_ns);
stream_ns.play("video1.flv");

var buffer_interval:Number = setInterval(checkBufferTime, 100, stream_ns);
function checkBufferTime(my_ns:NetStream):Void {
  var bufferPct:Number = Math.min(Math.round(my_ns.bufferLength/
  my_ns.bufferTime 100), 100);
  var output_str:String = "<textformat tabStops='[100,200]'>";
  output_str += "Length: "+my_ns.bufferLength+"\t"+"Time:
  "+my_ns.bufferTime+"\t"+"Buffer:"+bufferPct+"%";
  output_str += "</textformat>";
  buffer_txt.htmlText = output_str;
}
```

Si votre fichier SWF comprend un composant de la version 2, utilisez les composants de la version 2 de la classe DepthManager au lieu de la méthode `MovieClip.getNextHighestDepth()`, utilisée dans cet exemple.

### Voir également

`setBufferTime (méthode NetStream.setBufferTime)`, `time (propriété NetStream.time)`, `bufferLength (propriété NetStream.bufferLength)`

## bytesLoaded (propriété NetStream.bytesLoaded)

```
public bytesLoaded : Nombre [lecture seule]
```

Le nombre d'octets de données ayant été chargés dans le lecteur. Vous pouvez utiliser cette méthode conjointement avec `NetStream.bytesTotal` pour estimer le niveau de remplissage de la mémoire tampon, par exemple, pour permettre à un utilisateur qui attend la fin du chargement des données dans la mémoire tampon de consulter le compte rendu.

**Disponibilité :** ActionScript 1.0 ; Flash Player 7

### Exemple

L'exemple suivant crée une barre de progression avec l'API de dessin, ainsi que les propriétés `bytesLoaded` et `bytesTotal` qui affichent la progression du chargement de video1.flv dans l'occurrence d'objet vidéo appelé `my_video`. Un champ texte appelé `loaded_txt` est créé de façon dynamique pour afficher également des informations sur la progression du processus de chargement.

```
var connection_nc:NetConnection = new NetConnection();
connection_nc.connect(null);
var stream_ns:NetStream = new NetStream(connection_nc);
my_video.attachVideo(stream_ns);
stream_ns.play("video1.flv");

this.createTextField("loaded_txt", this.getNextHighestDepth(), 10, 10, 160,
  22);
this.createEmptyMovieClip("progressBar_mc", this.getNextHighestDepth());
progressBar_mc.createEmptyMovieClip("bar_mc",
  progressBar_mc.getNextHighestDepth());
with (progressBar_mc.bar_mc) {
  beginFill(0xFF0000);
  moveTo(0, 0);
  lineTo(100, 0);
  lineTo(100, 10);
  lineTo(0, 10);
  lineTo(0, 0);
  endFill();
  _xscale = 0;
}
progressBar_mc.createEmptyMovieClip("stroke_mc",
  progressBar_mc.getNextHighestDepth());
with (progressBar_mc.stroke_mc) {
  lineStyle(0, 0x000000);
  moveTo(0, 0);
  lineTo(100, 0);
  lineTo(100, 10);
  lineTo(0, 10);
  lineTo(0, 0);
}

var loaded_interval:Number = setInterval(checkBytesLoaded, 500, stream_ns);
function checkBytesLoaded(my_ns:NetStream) {
  var pctLoaded:Number = Math.round(my_ns.bytesLoaded/my_ns.bytesTotal
  100);
  loaded_txt.text = Math.round(my_ns.bytesLoaded/1000)+" of
  "+Math.round(my_ns.bytesTotal/1000)+" KB loaded ("+pctLoaded+"%)";
  progressBar_mc.bar_mc._xscale = pctLoaded;
  if (pctLoaded>=100) {
  clearInterval(loaded_interval);
  }
}
```

Si votre fichier SWF comprend un composant de la version 2, utilisez les composants de la version 2 de la classe DepthManager au lieu de la méthode `MovieClip.getNextHighestDepth()`, utilisée dans cet exemple.

### Voir également

bytesTotal (propriété NetStream.bytesTotal), bufferLength (propriété NetStream.bufferLength)

## bytesTotal (propriété NetStream.bytesTotal)

public bytesTotal : Nombre [lecture seule]

La taille totale, en octets, du fichier chargé dans le lecteur.

**Disponibilité :** ActionScript 1.0 ; Flash Player 7

### Exemple

L'exemple suivant crée une barre de progression avec l'API de dessin, ainsi que les propriétés bytesLoaded et bytesTotal qui affichent la progression du chargement de video1.flv dans l'occurrence d'objet vidéo appelé my_video. Un champ texte appelé loaded_txt est créé de façon dynamique pour afficher également des informations sur la progression du processus de chargement.

```
var connection_nc:NetConnection = new NetConnection();
connection_nc.connect(null);
var stream_ns:NetStream = new NetStream(connection_nc);
my_video.attachVideo(stream_ns);
stream_ns.play("video1.flv");

this.createTextField("loaded_txt", this.getNextHighestDepth(), 10, 10, 160,
  22);
this.createEmptyMovieClip("progressBar_mc", this.getNextHighestDepth());
progressBar_mc.createEmptyMovieClip("bar_mc",
  progressBar_mc.getNextHighestDepth());
with (progressBar_mc.bar_mc) {
  beginFill(0xFF0000);
  moveTo(0, 0);
  lineTo(100, 0);
  lineTo(100, 10);
  lineTo(0, 10);
  lineTo(0, 0);
  endFill();
  _xscale = 0;
}
progressBar_mc.createEmptyMovieClip("stroke_mc",
  progressBar_mc.getNextHighestDepth());
with (progressBar_mc.stroke_mc) {
  lineStyle(0, 0x000000);
  moveTo(0, 0);
  lineTo(100, 0);
  lineTo(100, 10);
  lineTo(0, 10);
```

```
  lineTo(0, 0);
}

var loaded_interval:Number = setInterval(checkBytesLoaded, 500, stream_ns);
function checkBytesLoaded(my_ns:NetStream) {
  var pctLoaded:Number = Math.round(my_ns.bytesLoaded/my_ns.bytesTotal
  100);
  loaded_txt.text = Math.round(my_ns.bytesLoaded/1000)+" of
  "+Math.round(my_ns.bytesTotal/1000)+" KB loaded ("+pctLoaded+"%)";
  progressBar_mc.bar_mc._xscale = pctLoaded;
  if (pctLoaded>=100) {
  clearInterval(loaded_interval);
  }
}
```

Si votre fichier SWF comprend un composant de la version 2, utilisez les composants de la version 2 de la classe DepthManager au lieu de la méthode `MovieClip.getNextHighestDepth()`, utilisée dans cet exemple.

### Voir également

`bytesLoaded (propriété NetStream.bytesLoaded)`, `bufferTime (propriété NetStream.bufferTime)`

## close (méthode NetStream.close)

```
public close() : Void
```

Arrête la lecture des données du flux de diffusion en continu, définit la propriété `NetStream.time` sur 0 et met le flux de diffusion en continu à disposition. Cette commande supprime également la copie locale d'un fichier FLV téléchargé via HTTP. Même si Flash Player supprime la copie locale du fichier FLV qu'il crée, une copie de la vidéo peut subsister dans le répertoire du cache du navigateur. Si une prévention totale de la mise en cache ou du stockage local du fichier FLV est requise, utilisez Flash Communication Server MX.

**Disponibilité :** ActionScript 1.0 ; Flash Player 7

### Exemple

La fonction `onDisconnect()` suivante ferme une connexion et supprime la copie temporaire de video1.flv, qui a été stockée sur le disque local lorsque vous cliquez sur le bouton nommé `close_btn` :

```
var connection_nc:NetConnection = new NetConnection();
connection_nc.connect(null);
var stream_ns:NetStream = new NetStream(connection_nc);
my_video.attachVideo(stream_ns);
stream_ns.play("video1.flv");
```

```
close_btn.onRelease = function(){
  stream_ns.close();
};
```

### Voir également

pause (méthode NetStream.pause), play (méthode NetStream.play)

## currentFps (propriété NetStream.currentFps)

```
public currentFps : Nombre [lecture seule]
```

Le nombre d'images affichées par seconde. Si vous exportez des fichiers FLV afin de les lire sur un certain nombre de systèmes, vous pouvez vérifier cette valeur pendant le test afin de vous aider à déterminer la compression à appliquer lors de l'exportation du fichier.

**Disponibilité :** ActionScript 1.0 ; Flash Player 7

### Exemple

L'exemple suivant crée un champ texte qui affiche le nombre actuel d'images par seconde affichées par video1.flv.

```
var connection_nc:NetConnection = new NetConnection();
connection_nc.connect(null);
var stream_ns:NetStream = new NetStream(connection_nc);
my_video.attachVideo(stream_ns);
stream_ns.play("video1.flv");

this.createTextField("fps_txt", this.getNextHighestDepth(), 10, 10, 50,
  22);
fps_txt.autoSize = true;
var fps_interval:Number = setInterval(displayFPS, 500, stream_ns);
function displayFPS(my_ns:NetStream) {
  fps_txt.text = "currentFps (frames per second):
  "+Math.floor(my_ns.currentFps);
}
```

Si votre fichier SWF comprend un composant de la version 2, utilisez les composants de la version 2 de la classe DepthManager au lieu de la méthode `MovieClip.getNextHighestDepth()`, utilisée dans cet exemple.

## NetStream, constructeur

```
public NetStream(connection:NetConnection)
```

Crée un flux de diffusion en continu qui permet de lire des fichiers FLV à l'aide de l'objet NetConnection spécifié.

**Disponibilité :** ActionScript 1.0 ; Flash Player 7

### Paramètres

*connection*:NetConnection - Objet NetConnection.

### Exemple

Le code suivant construit un nouvel objet NetConnection, `connection_nc`, et l'utilise pour construire un nouvel objet NetStream appelé `stream_ns`. Sélectionnez Nouvelle vidéo dans le menu d'options du panneau Bibliothèque pour créer une occurrence d'objet vidéo et appelez cette occurrence `my_video`.

```
var connection_nc:NetConnection = new NetConnection();
connection_nc.connect(null);
var stream_ns:NetStream = new NetStream(connection_nc);
my_video.attachVideo(stream_ns);
stream_ns.play("video1.flv");
```

### Voir également

NetConnection, attachVideo (méthode Video.attachVideo)

## onMetaData (gestionnaire NetStream.onMetaData)

```
onMetaData = function(infoObject:Object) {}
```

Invoqué lorsque Flash Player reçoit une description intégrée dans le fichier FLV lu.

L'utilitaire Flash Video Exporter (version 1.1 ou supérieure) intègre la durée de la vidéo, la date de création, les débits et d'autres informations dans le fichier vidéo. Différents codeurs vidéo intègrent différents jeux de métadonnées.

Ce gestionnaire est déclenché après un appel à la méthode `NetStream.play()`, mais avant que la tête de lecture vidéo ait avancé.

Dans la plupart des cas, la valeur de durée intégrée dans les métadonnées FLV se rapproche de la durée réelle, mais n'est pas exacte. En d'autres termes, elle ne correspond pas toujours à la valeur de la propriété `NetStream.time` lorsque la tête de lecture est à la fin du flux vidéo.

**Disponibilité :** ActionScript 1.0 ; Flash Player 7

### Paramètres

*infoObject*:Object - Objet comprenant une propriété pour chaque élément de métadonnées.

### Exemple

Le code donné dans cet exemple commence par la création de nouveaux objets NetConnection et NetStream. Il définit ensuite le gestionnaire `onMetaData()` pour l'objet NetStream. Le gestionnaire passe dans toutes les propriétés nommées dans l'objet infoObject reçu et imprime le nom et la valeur de la propriété.

```
var nc:NetConnection = new NetConnection();
nc.connect(null);
var ns:NetStream = new NetStream(nc);

ns.onMetaData = function(infoObject:Object) {
  for (var propName:String in infoObject) {
    trace(propName + " = " + infoObject[propName]);
  }
};

ns.play("http://www.helpexamples.com/flash/video/water.flv");
```

Les informations suivantes s'affichent alors :

```
canSeekToEnd = true
 videocodecid = 4
 framerate = 15
 videodatarate = 400
 height = 215
 width = 320
 duration = 7.347
```

La liste des propriétés peut varier selon le logiciel utilisé pour coder le fichier FLV.

### Voir également

`time (propriété NetStream.time)`, `play (méthode NetStream.play)`, `NetConnection`

## onStatus (gestionnaire NetStream.onStatus)

```
onStatus = fonction(infoObject:Objet) {}
```

Appelé à chaque changement d'état ou à chaque fois qu'une erreur est publiée pour l'objet NetStream. Si vous souhaitez répondre à ce gestionnaire d'événements, vous devez créer une fonction pour traiter l'objet d'informations.

L'objet d'informations a une propriété de code contenant une chaîne qui décrit le résultat du gestionnaire onStatus et une propriété de niveau contenant une chaîne qui est soit « état » soit « erreur ».

Flash propose également, outre ce gestionnaire onStatus, une « super » fonction appelée `System.onStatus`. Si onStatus est invoqué pour un objet particulier et qu'aucune fonction n'est affectée pour y répondre, Flash traite une fonction affectée à `System.onStatus` si elle existe.

Les événements suivants vous indiquent lorsque certaines activités NetStream ont lieu.

| Propriété du code | Propriété de niveau | Signification |
|---|---|---|
| `NetStream.Buffer.Empty` | état | Les données ne sont pas reçues suffisamment rapidement pour remplir le tampon. Le flux de données est interrompu tant que la mémoire tampon n'est pas rechargée : une fois l'opération terminée, un message `NetStream.Buffer.Full` est envoyé et la lecture du flux reprend. |
| `NetStream.Buffer.Full` | état | La mémoire tampon est pleine et la lecture du flux commence. |
| `NetStream.Buffer.Flush` | état | Le flux de données est terminé et le tampon restant va être vidé. |
| `NetStream.Play.Start` | état | La lecture a repris. |
| `NetStream.Play.Stop` | état | La lecture s'est arrêtée. |
| `NetStream.Play.StreamNotFound` | erreur | Le fichier FLV transmis à la méthode `play()` n'a pas été trouvé. |

| Propriété du code | Propriété de niveau | Signification |
|---|---|---|
| `NetStream.Seek.InvalidTime` | erreur | Pour une vidéo chargée avec un chargement progressif, l'utilisateur a essayé de rechercher ou de lire au-delà des données vidéo déjà chargées, ou après la fin de la vidéo lorsque le fichier a été totalement chargé. La propriété `message.details` contient un code de temps qui indique la dernière position valide de recherche utilisateur. |
| `NetStream.Seek.Notify` | état | L'opération de recherche est terminée. |

Si vous recevez systématiquement des erreurs concernant la mémoire tampon, vous devriez essayer de changer la mémoire tampon via la méthode `NetStream.setBufferTime()`.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

## Paramètres

*`infoObject`*`:Object` - Paramètre défini selon le message d'état ou un message d'erreur.

## Exemple

L'exemple suivant affiche les données relatives au flux dans le panneau de sortie :

```
var connection_nc:NetConnection = new NetConnection();
connection_nc.connect(null);
var stream_ns:NetStream = new NetStream(connection_nc);
my_video.attachVideo(stream_ns);
stream_ns.play("video1.flv");
stream_ns.onStatus = function(infoObject:Object) {
     trace("NetStream.onStatus called: ("+getTimer()+" ms)");
     for (var prop in infoObject) {
       trace("\t"+prop+":\t"+infoObject[prop]);
     }
     trace("");
};
```

### Voir également

`setBufferTime (méthode NetStream.setBufferTime)`, `onStatus (gestionnaire System.onStatus)`

## pause (méthode NetStream.pause)

`public pause([flag:Boolean]) : Void`

Interrompt ou reprend la lecture d'un flux.

Lorsque vous appelez cette méthode pour la première fois (sans envoyer de paramètre), la lecture est interrompue ; au prochain appel, la lecture reprend. Vous avez également la possibilité de lier cette méthode au bouton sur lequel l'utilisateur appuie pour interrompre ou reprendre la lecture.

**Disponibilité :** ActionScript 1.0 ; Flash Player 7

### Paramètres

*`flag`*`:Boolean` [facultatif] - Valeur booléenne permettant de spécifier si vous souhaitez interrompre la lecture (`true`) ou la reprendre (`false`). Si vous omettez ce paramètre, `NetStream.pause()` permet de procéder au basculement : lorsque cette méthode est appelée pour la première fois sur un flux spécifique, elle interrompt la lecture et, lors de l'appel suivant, la reprend.

### Exemple

Les exemples ci-dessous illustrent diverses façons d'utiliser cette méthode :

```
my_ns.pause(); // pauses play first time issued
my_ns.pause(); // resumes play
my_ns.pause(false); // no effect, play continues
my_ns.pause(); // pauses play
```

### Voir également

`close (méthode NetStream.close)`, `play (méthode NetStream.play)`

## play (méthode NetStream.play)

`public play(name:Object, start:Number, len:Number, reset:Object) : Void`

Commence la lecture d'un fichier vidéo externe (FLV). Pour afficher les données vidéo, vous devez appeler une méthode `Video.attachVideo()` ; la lecture des données audio transmises en continu avec la vidéo, ou d'un fichier FLV contenant uniquement des données audio, s'effectue automatiquement.

Si vous souhaitez contrôler la partie audio associée à un fichier FLV, vous pouvez utiliser `MovieClip.attachAudio()` pour ajouter le son à un clip ; vous pouvez ensuite créer un objet Sound pour contrôler certains aspects du son. Pour plus d'informations, consultez `MovieClip.attachAudio()`.

Si le fichier FLV est introuvable, le gestionnaire d'événements `NetStream.onStatus` est appelé. Si vous souhaitez interrompre un flux en cours de lecture, utilisez `NetStream.close()`.

Vous pouvez lire les fichiers FLV locaux stockés dans le même répertoire que le fichier SWF ou dans un sous-répertoire ; vous ne pouvez pas naviguer vers un répertoire de niveau supérieur. Par exemple, si le fichier SWF se trouve dans un répertoire nommé /training, et si vous souhaitez lire une vidéo stockée dans le répertoire /training/videos, utilisez la syntaxe suivante :

```
my_ns.play("videos/videoName.flv");
```

Pour lire une vidéo stockée dans le répertoire /training, utilisez la syntaxe suivante :

```
my_ns.play("videoName.flv");
```

Lorsque vous utilisez cette méthode, tenez compte du modèle de sécurité Flash Player.

Pour Flash Player 8 :

- NetStream.play() n'est pas autorisé si le fichier SWF appelant est dans le sandbox local avec système de fichier et que la ressource est dans un sandbox non local.
- L'accès au sandbox réseau à partir d'un sandbox local approuvé ou de réseau local nécessite une autorisation du site au travers d'un fichier de régulation interdomaine.

Pour plus d'informations, voir les sections suivantes :

- Chapitre 17, « Fonctionnement de la sécurité », dans *Formation à ActionScript 2.0 dans Flash*
- Le livre blanc sur la sécurité de Flash Player 8 disponible à l'adresse :http://www.macromedia.com/go/fp8_security
- Le livre blanc sur les API relatif à la sécurité de Flash Player 8 disponible à l'adresse : http://www.macromedia.com/go/fp8_security_apis

**Disponibilité :** ActionScript 1.0 ; Flash Player 7

## Paramètres

*`name`*`:Object` - Nom du fichier FLV à lire, entre guillemets. Les formats http:// et file:// sont pris en charge ; l'emplacement file:// est toujours relatif par rapport à l'emplacement du fichier SWF.

*`start`*`:Number` -

*len*:Number -

*reset*:Object -

### Exemple

L'exemple ci-dessous illustre diverses façons d'utiliser la commande `NetStream.play()`. Vous pouvez lire un fichier qui est sur l'ordinateur d'un utilisateur. Le répertoire joe_user est un sous-répertoire du répertoire où le fichier SWF est enregistré. Vous pouvez également lire un fichier sur un serveur.

```
// Play a file that is on the user's computer.
my_ns.play("file://joe_user/flash/videos/lectureJune26.flv");

// Play a file on a server.
my_ns.play("http://someServer.someDomain.com/flash/video/orientation.flv");
```

### Voir également

`attachAudio (méthode MovieClip.attachAudio)`, `close (méthode NetStream.close)`, `onStatus (gestionnaire NetStream.onStatus)`, `pause (méthode NetStream.pause)`, `attachVideo (méthode Video.attachVideo)`

## seek (méthode NetStream.seek)

```
public seek(offset:Number) : Void
```

Recherche l'image-clé la plus proche du nombre de secondes spécifié à partir du début du flux. La lecture du flux reprend lorsqu'elle atteint l'emplacement spécifié dans le flux.

**Disponibilité :** ActionScript 1.0 ; Flash Player 7

### Paramètres

*offset*:Number - Valeur de temps approximative, en secondes, à atteindre dans un fichier FLV. La tête de lecture se place sur l'image clé de la vidéo qui est la plus proche de *numberOfSeconds*.

- Pour revenir au début du flux de diffusion, transmettez 0 pour *numberOfSeconds*.
- Pour effectuer une recherche en partant du début du flux, transmettez le nombre de secondes à appliquer. Par exemple, pour placer la tête de lecture 15 secondes après le début, utilisez *my_ns* `.seek(15)`.
- Pour faire une recherche par rapport à la position actuelle, transmettez *my_ns* `.time + n` ou *my_ns* `.time - n`, pour rechercher *n* secondes vers l'avant ou l'arrière, par rapport à la position actuelle. Par exemple, pour revenir en arrière de 20 secondes par rapport à la position actuelle, utilisez `my_ns.seek(my_ns.time - 20)`.

L'emplacement exact de recherche diffère selon le nombre d'images par seconde sélectionnées lors de l'exportation. Par conséquent, si la même vidéo est exportée à 6 ips et 30 ips, la recherche portera sur deux points différents si vous utilisez, par exemple, `my_ns.seek(15)` pour les deux objets vidéo.

### Exemple

L'exemple ci-dessous illustre diverses façons d'utiliser la commande `NetStream.seek()`. Vous pouvez revenir au début du flux, aller à un emplacement à 30 secondes du début du flux et revenir en arrière de trois minutes par rapport à l'emplacement actuel :

```
// Return to the beginning of the stream
my_ns.seek(0);

// Move to a location 30 seconds from the beginning of the stream
my_ns.seek(30);

// Move backwards three minutes from current location
my_ns.seek(my_ns.time - 180);
```

### Voir également

, `time (propriété NetStream.time)`

## setBufferTime (méthode NetStream.setBufferTime)

```
public setBufferTime(bufferTime:Number) : Void
```

Spécifie la durée de la mise en mémoire tampon des messages avant de commencer à afficher le flux. Par exemple, si vous voulez vous assurer que la lecture du flux soit ininterrompue au cours des 15 premières secondes, définissez *`numberOfSeconds`* sur 15 ; Flash commence la lecture du flux uniquement 15 secondes après la mise en mémoire tampon des données.

**Disponibilité :** ActionScript 1.0 ; Flash Player 7

### Paramètres

*`bufferTime`*`:Number` - Nombre de secondes de données à placer en mémoire tampon avant que Flash n'affiche les données. La valeur par défaut est 0,1 (un dixième de seconde).

### Exemple

Consultez l'exemple relatif à `NetStream.bufferLength`.

### Voir également

`bufferLength (propriété NetStream.bufferLength)`, `bufferTime (propriété NetStream.bufferTime)`

# time (propriété NetStream.time)

```
public time : Nombre [lecture seule]
```

La position de la tête de lecture, en secondes.

**Disponibilité :** ActionScript 1.0 ; Flash Player 7

## Exemple

L'exemple suivant affiche la position actuelle de la tête de lecture dans un champ texte créé de façon dynamique, appelé `time_txt`. Sélectionnez Nouvelle vidéo dans le menu d'options du panneau Bibliothèque pour créer une occurrence d'objet vidéo et appelez cette occurrence `my_video`. Créez un objet vidéo appelé my_video. Ajoutez le code ActionScript suivant à votre fichier FLA ou AS :

```
var connection_nc:NetConnection = new NetConnection();
connection_nc.connect(null);
var stream_ns:NetStream = new NetStream(connection_nc);
my_video.attachVideo(stream_ns);
stream_ns.play("video1.flv");
//
stream_ns.onStatus = function(infoObject:Object) {
  statusCode_txt.text = infoObject.code;
};

this.createTextField("time_txt", this.getNextHighestDepth(), 10, 10, 100,
  22);
time_txt.text = "LOADING";

var time_interval:Number = setInterval(checkTime, 500, stream_ns);
function checkTime(my_ns:NetStream) {
  var ns_seconds:Number = my_ns.time;
  var minutes:Number = Math.floor(ns_seconds/60);
  var seconds = Math.floor(ns_seconds%60);
  if (seconds<10) {
  seconds = "0"+seconds;
  }
  time_txt.text = minutes+":"+seconds;
}
```

Si votre fichier SWF comprend un composant de la version 2, utilisez les composants de la version 2 de la classe DepthManager au lieu de la méthode `MovieClip.getNextHighestDepth()`, utilisée dans cet exemple.

## Voir également

`bufferLength (propriété NetStream.bufferLength)`, `bytesLoaded (propriété NetStream.bytesLoaded)`

# Number

```
Object
  |
  +-Number
```

```
public class Number
extends Object
```

La classe Number est une enveloppe simple dédiée au type de données Number. Vous pouvez manipuler des valeurs numériques primitives à l'aide des méthodes et des propriétés associées à la classe Number. Cette classe est identique à la classe JavaScript Number.

Les propriétés de la classe Number sont statiques, ce qui signifie qu'il n'est pas nécessaire de disposer d'un objet pour les utiliser ; par conséquent, il n'est pas nécessaire d'utiliser le constructeur.

L'exemple suivant appelle la méthode `toString()` de la classe Number, qui renvoie la chaîne `1234`:

```
var myNumber:Number = new Number(1234);
myNumber.toString();
```

L'exemple suivant affecte la valeur de la propriété `MIN_VALUE` à une variable déclarée sans l'utilisation du constructeur :

```
var smallest:Number = Number.MIN_VALUE;
```

**Disponibilité :** ActionScript 1.0 ; Flash Player 5

## Résumé des propriétés

| Modificateurs | Propriété | Description |
|---|---|---|
| `static` | `MAX_VALUE:Number` | Nombre représentable le plus élevé (double précision conformément à IEEE-754). |
| `static` | `MIN_VALUE:Number` | Nombre représentable le plus faible (double précision conformément à IEEE-754). |
| `static` | `NaN:Number` | Valeur IEEE-754 représentant Not A Number (`NaN`). |
| `static` | `NEGATIVE_INFINITY:Number` | Spécifie la valeur IEEE-754 représentant l'infini négatif. |
| `static` | `POSITIVE_INFINITY:Number` | Spécifie la valeur IEEE-754 représentant l'infini positif. |

*Propriétés héritées de la classe Object*

| |
|---|
| constructeur (propriété Object.constructor), __proto__ (Object.__proto__ property), prototype (Object.prototype, propriété), __resolve (Object.__resolve, propriété) |

### Résumé des constructeurs

| Signature | Description |
|---|---|
| `Number(num:Object)` | Crée un objet Number. |

### Résumé de la méthode

| Modificateurs | Signature | Description |
|---|---|---|
| | `toString(radix:Number) : String` | Renvoie une chaîne spécifiant l'objet Number spécifié (*myNumber*). |
| | `valueOf() : Number` | Renvoie le type de valeurs primitif de l'objet Number spécifié. |

*Méthodes héritées de la classe Object*

| |
|---|
| addProperty (méthode Object.addProperty), hasOwnProperty (méthode Object.hasOwnProperty), isPropertyEnumerable (méthode Object.isPropertyEnumerable), isPrototypeOf (méthode Object.isPrototypeOf), registerClass (méthode Object.registerClass), toString (méthode Object.toString), unwatch (méthode Object.unwatch), valueOf (méthode Object.valueOf), watch (méthode Object.watch) |

# MAX_VALUE (propriété Number.MAX_VALUE)

```
public static MAX_VALUE : Number
```

Nombre représentable le plus élevé (double précision conformément à IEEE-754). La valeur de ce nombre est d'environ 1,79e+308.

**Disponibilité :** ActionScript 1.0 ; Flash Player 5

## Exemple

L'ActionScript suivant affiche les nombres présentables les plus grands et les plus faibles dans le panneau de sortie.

```
trace("Number.MIN_VALUE = "+Number.MIN_VALUE);
trace("Number.MAX_VALUE = "+Number.MAX_VALUE);
```

Ce code affiche les valeurs suivantes :

```
Number.MIN_VALUE = 4.94065645841247e-324
Number.MAX_VALUE = 1.79769313486232e+308
```

# MIN_VALUE (propriété Number.MIN_VALUE)

```
public static MIN_VALUE : Number
```

Nombre représentable le plus faible (comportant deux décimales conformément à IEEE-754). La valeur de ce nombre est d'environ 5e-324.

**Disponibilité :** ActionScript 1.0 ; Flash Player 5

## Exemple

Le code ActionScript suivant affiche les nombres représentables le plus élevé et le plus faible dans le panneau de sortie.

```
trace("Number.MIN_VALUE = "+Number.MIN_VALUE);
trace("Number.MAX_VALUE = "+Number.MAX_VALUE);
```

Ce code affiche les valeurs suivantes :

```
Number.MIN_VALUE = 4.94065645841247e-324
Number.MAX_VALUE = 1.79769313486232e+308
```

# NaN (propriété Number.NaN)

```
public static NaN : Number
```

Valeur IEEE-754 ne représentant pas une valeur numérique (`NaN`).

**Disponibilité :** ActionScript 1.0 ; Flash Player 5

## Voir également

# NEGATIVE_INFINITY (propriété Number.NEGATIVE_INFINITY)

```
public static NEGATIVE_INFINITY : Number
```

Spécifie la valeur IEEE-754 représentant l'infini négatif. La valeur de cette propriété est identique à celle de la constante `-Infinity`.

L'infini négatif est une valeur numérique spéciale renvoyée lorsqu'une opération mathématique ou une fonction renvoie une valeur négative supérieure à celle pouvant être représentée.

**Disponibilité :** ActionScript 1.0 ; Flash Player 5

### Exemple

Cet exemple compare le résultat de la division des valeurs suivantes.

```
var posResult:Number = 1/0;
if (posResult == Number.POSITIVE_INFINITY) {
  trace("posResult = "+posResult); // output: posResult = Infinity
}
var negResult:Number = -1/0;
if (negResult == Number.NEGATIVE_INFINITY) {
  trace("negResult = "+negResult); // output: negResult = -Infinity
```

## Number, constructeur

```
public Number(num:Object)
```

Crée un nouvel objet Number. Le constructeur `new Number` est surtout utilisé en tant qu'espace réservé. Un objet Number n'est pas identique à la fonction `Number()` qui convertit un paramètre en valeur primitive.

**Disponibilité :** ActionScript 1.0 ; Flash Player 5

### Paramètres

*num*`:Object` - Valeur numérique de l'objet Number en cours de création, ou valeur à convertir en nombre. La valeur par défaut est de 0 si la valeur `value` n'est pas fournie.

### Exemple

Le code suivant crée de nouveaux objets Number :

```
var n1:Number = new Number(3.4);
var n2:Number = new Number(-10);
```

### Voir également

`toString (méthode Number.toString)`, `valueOf (méthode Number.valueOf)`

## POSITIVE_INFINITY (propriété Number.POSITIVE_INFINITY)

```
public static POSITIVE_INFINITY : Number
```

Spécifie la valeur IEEE-754 représentant l'infini positif. La valeur de cette propriété est identique à celle de la constante `Infinity`.

L'infini positif est une valeur numérique spéciale renvoyée lorsqu'une opération mathématique ou une fonction renvoie une valeur supérieure à celle pouvant être représentée.

**Disponibilité :** ActionScript 1.0 ; Flash Player 5

### Exemple

Cet exemple compare le résultat de la division des valeurs suivantes.

```
var posResult:Number = 1/0;
if (posResult == Number.POSITIVE_INFINITY) {
  trace("posResult = "+posResult); // output: posResult = Infinity
}
var negResult:Number = -1/0;
if (negResult == Number.NEGATIVE_INFINITY) {
  trace("negResult = "+negResult); // output: negResult = -Infinity
```

## toString (méthode Number.toString)

```
public toString(radix:Number) : String
```

Renvoie la représentation sous la forme d'une chaîne spécifiant l'objet Number spécifié (*myNumber*).

**Disponibilité :** ActionScript 1.0 ; Flash Player 5

### Paramètres

*radix*:Number - Spécifie la base numérique (de 2 à 36) à appliquer pour la conversion nombre vers chaîne. Si vous omettez le paramètre `radix`, la valeur par défaut est de 10.

### Valeur renvoyée

`String` - Chaîne.

### Exemple

L'exemple suivant utilise 2 et 8 pour le paramètre `radix` et renvoie une chaîne qui contient la représentation correspondante du numéro 9 :

```
var myNumber:Number = new Number(9);
trace(myNumber.toString(2)); // output: 1001
trace(myNumber.toString(8)); // output: 11
```

L'exemple suivant renvoie une valeur hexadécimale.

```
var r:Number = new Number(250);
var g:Number = new Number(128);
var b:Number = new Number(114);
var rgb:String = "0x"+ r.toString(16)+g.toString(16)+b.toString(16);
trace(rgb);
// output: rgb:0xFA8072 (Hexadecimal equivalent of the color 'salmon')
```

## valueOf (méthode Number.valueOf)

```
public valueOf() : Number
```

Renvoie le type de valeurs primitif de l'objet Number spécifié.

**Disponibilité :** ActionScript 1.0 ; Flash Player 5

### Valeur renvoyée

`Number` - Chaîne.

### Exemple

L'exemple suivant a pour résultat la valeur primitive de l'objet `numSocks`.

```
var numSocks = new Number(2);
trace(numSocks.valueOf()); // output: 2
```

# Object

```
Object
```

`public class` *Object*

La classe Object forme la racine de la hiérarchie de classes ActionScript. Cette classe contient un petit sous-ensemble des fonctions fournies par la classe Object de JavaScript.

**Disponibilité :** ActionScript 1.0 ; Flash Player 5

### Résumé des propriétés

| Modificateurs | Propriété | Description |
|---|---|---|
| | `constructor:Object` | Référence à la fonction constructeur pour une occurrence d'objet donnée. |
| | `__proto__:Object` | Fait référence à la propriété `prototype` de la classe (ActionScript 2.0) ou de la fonction constructeur (ActionScript 1.0) utilisée pour créer l'objet. |
| `static` | `prototype:Object` | Une référence à la superclasse d'un objet classe ou fonction. |
| | `__resolve:Object` | Référence à une fonction définie par l'utilisateur qui est appelée si le code ActionScript fait référence à une propriété ou une méthode non définie. |

## Résumé des constructeurs

| Signature | Description |
|---|---|
| `Object()` | Crée un objet Object et stocke une référence à la méthode constructeur de l'objet dans la propriété `constructor` de ce dernier. |

## Résumé de la méthode

| Modificateurs | Signature | Description |
|---|---|---|
| | `addProperty(name:String, getter:Function, setter:Function) : Boolean` | Crée une propriété de lecture/définition. |
| | `hasOwnProperty(name:String) : Boolean` | Indique si la propriété spécifiée d'un objet est définie. |
| | `isPropertyEnumerable(name:String) : Boolean` | Indique si la propriété spécifiée existe et est énumérable. |
| | `isPrototypeOf(theClass:Object) : Boolean` | Indique si une occurrence de la classe Object figure dans la chaîne de prototypes de l'objet spécifié en tant qu'argument. |
| `static` | `registerClass(name:String, theClass:Function) : Boolean` | Associe un symbole de clip à une classe d'objet ActionScript. |
| | `toString() : String` | Convertit l'objet spécifié en chaîne et renvoie cette dernière. |
| | `unwatch(name:String) : Boolean` | Supprime un point d'observation créé par `Object.watch()`. |
| | `valueOf() : Object` | Renvoie la valeur primitive de l'objet spécifié. |
| | `watch(name:String, callback:Function, [userData:Object]) : Boolean` | Enregistre un gestionnaire d'événements à appeler lorsque la propriété spécifiée d'un objet ActionScript change. |

# addProperty (méthode Object.addProperty)

```
public addProperty(name:String, getter:Function, setter:Function) : Boolean
```

Crée une propriété de lecture/définition. Lorsque Flash lit une propriété de lecture/définition, il appelle la fonction de lecture `get` et la valeur renvoyée par la fonction devient la valeur de `name`. Lorsque Flash écrit une propriété de lecture/définition, il appelle la fonction de définition `set` et transmet la nouvelle valeur comme paramètre. Si une propriété portant le nom donné existe déjà, la nouvelle propriété la remplace.

Une fonction de « lecture » est une fonction sans paramètre. La valeur renvoyée peut être de n'importe quel type. Son type peut changer d'une invocation à l'autre. La valeur renvoyée est considérée comme la valeur actuelle de la propriété.

Une fonction de « définition » est une fonction qui prend un paramètre, qui correspond à la nouvelle valeur de la propriété. Par exemple, si la propriété `x` est affectée par l'instruction `x = 1`, le paramètre `1` du numéro de type est transmis à la fonction de définition. La valeur renvoyée par la fonction de définition est ignorée.

Vous pouvez ajouter des propriétés de lecture/définition à des objets prototypes. Dans ce cas, toutes les occurrences d'objet qui héritent de l'objet prototype héritent de la propriété de lecture/définition. Cela permet d'ajouter une propriété de lecture/définition à un emplacement, au niveau de l'objet prototype, et de la propager à toutes les occurrences d'une classe, tout comme lorsque vous ajoutez des méthodes à des objets prototypes. Si une fonction de lecture/définition est invoquée pour une propriété de lecture/définition dans un objet prototype hérité, la référence transmise à la fonction de lecture/définition sera l'objet originellement référencé et non l'objet prototype.

En cas d'appel incorrect, `Object.addProperty()` risque d'échouer et de provoquer une erreur. Le tableau suivant décrit les erreurs qui risquent de se produire :

| Situation d'erreur | Que se passe-t-il ? |
|---|---|
| Le nom de propriété `name` n'est pas valide. Par exemple une chaîne vide. | Renvoie `false` et la propriété n'est pas ajoutée. |
| L'objet de définition `getter` n'est pas un objet de fonction valide. | Renvoie `false` et la propriété n'est pas ajoutée. |
| L'objet de définition `setter` n'est pas un objet de fonction valide. | Renvoie `false` et la propriété n'est pas ajoutée. |

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

## Paramètres

*name*:String - Chaîne ; nom de la propriété d'objet à créer.

*getter*:Function - Fonction appelée pour récupérer la valeur de la propriété ; ce paramètre est un objet Function.

*setter*:Function - Fonction appelée pour définir la valeur de la propriété ; ce paramètre est un objet Function. Si vous transmettez la valeur `null` pour ce paramètre, la propriété est en lecture seule.

## Valeur renvoyée

`Boolean` - Valeur booléenne : `true` si la propriété a été créée correctement ; `false` dans tous les autres cas.

## Exemple

Dans l'exemple suivant, un objet comporte deux méthodes internes, `setQuantity()` et `getQuantity()`. Une propriété, `bookcount`, permet d'appeler ces méthodes lorsqu'elle est définie ou récupérée. Une troisième méthode, `getTitle()`, renvoie une valeur en lecture seule qui est associée à la propriété `bookname`. Lorsqu'un script récupère la valeur de `myBook.bookcount`, l'interpréteur ActionScript appelle automatiquement `myBook.getQuantity()`. Lorsqu'un script modifie la valeur de `myBook.bookcount`, l'interpréteur appelle `myObject.setQuantity()`. La propriété `bookname` ne spécifie pas une fonction de définition `set`. Les tentatives de modification `bookname` sont donc ignorées.

```
function Book() {
  this.setQuantity = function(numBooks:Number):Void {
  this.books = numBooks;
  };
  this.getQuantity = function():Number {
  return this.books;
  };
  this.getTitle = function():String {
  return "Catcher in the Rye";
  };
  this.addProperty("bookcount", this.getQuantity, this.setQuantity);
  this.addProperty("bookname", this.getTitle, null);
}
var myBook = new Book();
myBook.bookcount = 5;
trace("You ordered "+myBook.bookcount+" copies of "+myBook.bookname);
// output: You ordered 5 copies of Catcher in the Rye
```

L'exemple précédent est fonctionnel, mais les propriétés `bookcount` et `bookname` sont ajoutées à chaque occurrence de l'objet `Book`, ce qui implique deux propriétés pour chaque occurrence de l'objet. Si la classe comporte de nombreuses propriétés, telles que `bookcount` et `bookname`,, ces propriétés risquent de consommer beaucoup de mémoire. Une alternative consiste à ajouter les propriété à `Book.prototype` pour que les propriétés `bookcount` et `bookname` n'existent qu'en un seul emplacement. L'effet obtenu est toutefois identique à celui du code dans l'exemple où `bookcount` et `bookname` étaient directement ajoutés à chaque occurrence. Si vous tentez d'accéder à l'une de ces propriétés dans une occurrence de Book, l'absence de la propriété entraîne le chaînage de prototype jusqu'à ce que les versions définies dans `Book.prototype` soient détectées. L'exemple suivant indique comment ajouter les propriétés à `Book.prototype`:

```
function Book() {}
Book.prototype.setQuantity = function(numBooks:Number):Void {
  this.books = numBooks;
};
Book.prototype.getQuantity = function():Number {
  return this.books;
};
Book.prototype.getTitle = function():String {
  return "Catcher in the Rye";
};
Book.prototype.addProperty("bookcount", Book.prototype.getQuantity,
  Book.prototype.setQuantity);
Book.prototype.addProperty("bookname", Book.prototype.getTitle, null);
var myBook = new Book();
myBook.bookcount = 5;
trace("You ordered "+myBook.bookcount+" copies of "+myBook.bookname);
```

L'exemple suivant indique comment utiliser les fonctions de lecture/définition implicites qui sont disponibles dans ActionScript 2.0. Au lieu de définir la fonction `Book` et modifier `Book.prototype`, définissez la classe `Book` dans un fichier externe appelé Book.as. Le code suivant doit figurer dans un fichier externe distinct appelé Book.as qui ne contient que cette définition de classe et figure dans le chemin de classe de l'application Flash :

```
class Book {
  var books:Number;
  function set bookcount(numBooks:Number):Void {
  this.books = numBooks;
  }
  function get bookcount():Number {
  return this.books;
  }
  function get bookname():String {
  return "Catcher in the Rye";
  }
}
```

Le code suivant peut ensuite être placé dans un fichier FLA et fonctionner de la même façon que dans les exemples précédents :

```
var myBook:Book = new Book();
myBook.bookcount = 5;
trace("You ordered "+myBook.bookcount+" copies of "+myBook.bookname);
```

### Voir également

`Instruction get`, `Instruction set`

## constructeur (propriété Object.constructor)

```
public constructor : Object
```

Référence à la fonction constructeur pour une occurrence d'objet donnée. La propriété `constructor` est automatiquement affectée à tous les objets au moment de leur création à l'aide du constructeur de la classe Object.

**Disponibilité :** ActionScript 1.0 ; Flash Player 5

### Exemple

L'exemple suivant constitue une référence à la fonction constructor pour l'objet `myObject`.

```
var my_str:String = new String("sven");
trace(my_str.constructor == String); //output: true
```

Si vous utilisez l'opérateur `instanceof`, vous pouvez également déterminer si un objet appartient à une classe spécifiée :

```
var my_str:String = new String("sven");
trace(my_str instanceof String); //output: true
```

Cependant, dans l'exemple suivant, la propriété `Object.constructor` convertit les types de données primitifs (tels que le littéral de chaîne affiché ici) en objets enveloppe. L'opérateur `instanceof` n'effectue aucune conversion, comme il est indiqué dans l'exemple suivant :

```
var my_str:String = "sven";
trace(my_str.constructor == String); //output: true
trace(my_str instanceof String); //output: false
```

### Voir également

`Opérateur instanceof`

## hasOwnProperty (méthode Object.hasOwnProperty)

```
public hasOwnProperty(name:String) : Boolean
```

Indique si la propriété spécifiée d'un objet est définie. Cette méthode renvoie `true` si l'objet cible comporte une propriété qui correspond à la chaîne spécifiée par le paramètre `name` et `false` dans les autres cas. Cette méthode ne vérifie pas le chaînage de prototype de l'objet et ne renvoie `true` que si la propriété existe sur l'objet lui-même.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

### Paramètres

*`name`*`:String` -

### Valeur renvoyée

`Boolean` - Valeur booléenne : `true` si l'objet cible comporte la propriété spécifiée par le paramètre `name`, `false` dans tous les autres cas.

## isPropertyEnumerable (méthode Object.isPropertyEnumerable)

```
public isPropertyEnumerable(name:String) : Boolean
```

Indique si la propriété spécifiée existe et est énumérable. Si la valeur est `true`, la propriété existe et peut être énumérée dans une boucle for... in. La propriété doit exister au niveau de l'objet cible dans la mesure où cette méthode ne vérifie pas le chaînage de prototype de l'objet cible.

Les propriétés que vous créez sont énumérables, contrairement aux propriétés intégrées qui ne le sont généralement pas.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

### Paramètres

*`name`*`:String` -

### Valeur renvoyée

`Boolean` - Valeur booléenne : `true` si la propriété spécifiée par le paramètre `name` est énumérable.

### Exemple

L'exemple suivant crée un objet générique, ajoute une propriété à cet objet, puis vérifie si elle est énumérable. Par contraste, l'exemple indique également qu'une propriété intégrée, la propriété `Array.length`, n'est pas énumérable.

```
var myObj:Object = new Object();
myObj.prop1 = "hello";
trace(myObj.isPropertyEnumerable("prop1")); // Output: true

var myArray = new Array();
trace(myArray.isPropertyEnumerable("length")); // Output: false
```

### Voir également

`Instruction for..in`

## isPrototypeOf (méthode Object.isPrototypeOf)

```
public isPrototypeOf(theClass:Object) : Boolean
```

Indique si une occurrence de la classe Object figure dans la chaîne de prototypes de l'objet spécifié en tant qu'argument. Cette méthode renvoie `true` si l'objet figure dans le chaînage de prototype de l'objet spécifié par le paramètre `theClass`. La méthode renvoie `false` non seulement si l'objet cible est absent du chaînage de prototype de l'objet `theClass`, mais aussi si l'argument `theClass` n'est pas un objet.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

### Paramètres

*`theClass`*`:Object` -

### Valeur renvoyée

`Boolean` - Valeur booléenne : `true` si l'objet figure dans le chaînage de prototype de l'objet spécifié par le paramètre `theClass`; `false` dans tous les autres cas.

## Constructeur Object

```
public Object()
```

Crée un objet Object et stocke une référence à la méthode constructeur de l'objet dans la propriété `constructor` de ce dernier.

**Disponibilité :** ActionScript 1.0 ; Flash Player 5

### Exemple

L'exemple suivant crée un objet générique appelé myObject :

```
var myObject:Object = new Object();
```

## __proto__ (Object.__proto__ property)

```
public __proto__ : Object
```

Fait référence à la propriété `prototype` de la classe (ActionScript 2.0) ou de la fonction constructeur (ActionScript 1.0) utilisée pour créer l'objet. La propriété `__proto__` est automatiquement affectée à tous les objets au moment de leur création. L'interpréteur d'ActionScript utilise la propriété `__proto__` pour accéder à la propriété `prototype` de la classe de l'objet ou de la fonction constructeur afin de rechercher les propriétés et les méthodes héritées par l'objet de sa superclasse.

**Disponibilité :** ActionScript 1.0 ; Flash Player 5

### Exemple

L'exemple suivant crée une classe appelée Shape, ainsi qu'une sous-classe de Shape appelée Circle.

```
// Shape class defined in external file named Shape.as
class Shape {
  function Shape() {}
}

// Circle class defined in external file named Circle.as
class Circle extends Shape{
  function Circle() {}
}
```

La classe Circle permet de créer deux instances de Circle :

```
var oneCircle:Circle = new Circle();
var twoCircle:Circle = new Circle();
```

Les instructions de suivi ci-dessous indiquent que la propriété `__proto_` des deux occurrences se rapporte à la propriété `prototype` de la classe Circle.

```
trace(Circle.prototype == oneCircle.__proto__); // Output: true
trace(Circle.prototype == twoCircle.__proto__); // Output: true
```

### Voir également

`prototype (Object.prototype, propriété)`

# prototype (Object.prototype, propriété)

```
public static prototype : Object
```

Une référence à la superclasse d'un objet classe ou fonction. La propriété `prototype` est créée automatiquement et est liée à tout objet classe ou fonction que vous créez. Cette propriété est statique dans la mesure où elle est propre à la classe ou la fonction que vous créez. Par exemple, si vous créez une classe personnalisée, la valeur de la propriété `prototype` est partagée par toutes les occurrences de la classe et est accessible uniquement en tant que propriété de classe. Les occurrences de votre classe personnalisée ne permettent pas d'accéder directement à la propriété `prototype`, mais peuvent y accéder par l'intermédiaire de la propriété `__proto__`.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

## Exemple

L'exemple suivant crée une classe appelée Shape, ainsi qu'une sous-classe de Shape appelée Circle.

```
// Shape class defined in external file named Shape.as
class Shape {
  function Shape() {}
}

// Circle class defined in external file named Circle.as
class Circle extends Shape{
  function Circle() {}
}
```

La classe Circle permet de créer deux instances de Circle :

```
var oneCircle:Circle = new Circle();
var twoCircle:Circle = new Circle();
```

L'instruction trace suivante indique que la propriété `prototype` de la classe Circle pointe vers sa superclasse Shape. L'identifiant `Shape` fait référence à la fonction constructeur de la classe Shape.

```
trace(Circle.prototype.constructor == Shape); // Output: true
```

L'instruction trace suivante indique comment combiner les propriétés `prototype` et `__proto__` pour progresser de deux niveaux dans la hiérarchie d'héritage (ou le chaînage de prototype). La propriété `Circle.prototype.__proto__` contient une référence à la superclasse de la classe Shape.

```
trace(Circle.prototype.__proto__ == Shape.prototype); // Output: true
```

## Voir également

`__proto__ (Object.__proto__ property)`

# registerClass (méthode Object.registerClass)

```
public static registerClass(name:String, theClass:Function) : Boolean
```

Associe un symbole de clip à une classe d'objet ActionScript. Si aucun symbole n'existe, Flash crée une association entre un identifiant de chaîne et une classe d'objet.

Lorsqu'une occurrence du symbole de clip spécifié est placée sur le scénario, elle est enregistrée dans la classe spécifiée par le paramètre `theClass` et non pas dans la classe MovieClip.

Lorsqu'une occurrence du symbole de clip spécifié est créé via `MovieClip.attachMovie()` ou `MovieClip.duplicateMovieClip()`, elle est enregistrée dans la classe spécifiée par `theClass` et non pas dans la classe MovieClip. Si la valeur de `theClass` est `null`, cette méthode supprime toutes les définitions de classe ActionScript associées au symbole de clip ou à l'identifiant de classe spécifié. Pour les symboles de clip, toutes les occurrences existantes du clip restent inchangées ; en revanche, les nouvelles occurrences du symbole sont associées à la classe MovieClip par défaut.

Si un symbole est déjà enregistré dans une classe, cette méthode la remplace par le nouvel enregistrement.

Lorsqu'une occurrence de clip est placée par le scénario ou créée via `attachMovie()` ou `duplicateMovieClip()`, ActionScript invoque le constructeur pour la classe appropriée en utilisant le mot-clé `this` pointant vers l'objet. La fonction constructeur est appelée sans paramètre.

Si vous utilisez cette méthode pour enregistrer un clip avec une classe ActionScript autre que MovieClip, le symbole du clip n'hérite pas des méthodes, propriétés et événements de la classe MovieClip intégrée sauf si vous incluez la classe MovieClip dans le chaînage de prototype de la nouvelle classe. Le code suivant crée une nouvelle classe ActionScript appelée `theClass` héritant des propriétés de la classe MovieClip :

```
theClass.prototype = new MovieClip();
```

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

## Paramètres

*`name`*`:String` - Chaîne ; identifiant de liaison du symbole de clip ou identifiant de chaîne de la classe ActionScript.

*`theClass`*`:Function` - Référence à la fonction constructeur de la classe ActionScript ou `null` pour annuler l'enregistrement du symbole.

## Valeur renvoyée

`Boolean` - Valeur booléenne : si l'enregistrement de la classe réussit, la valeur `true` est renvoyée ; la valeur `false` est renvoyée dans tous les autres cas.

### Voir également

attachMovie (méthode MovieClip.attachMovie), duplicateMovieClip (méthode MovieClip.duplicateMovieClip)

## __resolve (Object.__resolve, propriété)

```
public __resolve : Object
```

Référence à une fonction définie par l'utilisateur qui est appelée si le code ActionScript fait référence à une propriété ou une méthode non définie. Si le code ActionScript fait référence à une propriété ou méthode non définie d'un objet, Flash Player détermine si la propriété `__resolve` de l'objet est définie. Si la propriété `__resolve` est définie, la fonction à laquelle elle fait référence est exécutée et reçoit le nom de la propriété ou de la méthode non définie. Cela vous permet de fournir par programmation des valeurs pour les propriétés et les instructions non définies des méthodes non définies en leur donnant l'apparence des propriétés ou des méthodes définies. Cette propriété est particulièrement utile pour établir une communication client/serveur hautement transparente et elle est recommandée pour appeler les méthodes côté serveur.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

### Exemple

Les exemples suivants développent progressivement le premier exemple et illustrent cinq utilisations différentes de la propriété `__resolve`. Pour faciliter la compréhension, les instructions clés qui diffèrent de l'usage précédent sont indiquées en gras.

Usage 1 : l'exemple suivant utilise `__resolve` pour créer un objet où toute propriété non définie renvoie la valeur `"Hello, world!"`.

```
// instantiate a new object
var myObject:Object = new Object();

// define the __resolve function
myObject.__resolve = function (name) {
  return "Hello, world!";
};
trace (myObject.property1); // output: Hello, world!
trace (myObject.property2); // output: Hello, world!
```

Usage 2 : l'exemple suivant utilise `__resolve` en tant que *foncteur*, qui est une fonction générant d'autres fonctions. L'utilisation de `__resolve` redirige les appels de méthode non définis vers une fonction générique nommée `myFunction`.

```
// instantiate a new object
var myObject:Object = new Object();
```

```
// define a function for __resolve to call
myObject.myFunction = function (name) {
  trace("Method " + name + " was called");
};

// define the __resolve function
myObject.__resolve = function (name) {
   return function () { this.myFunction(name); };
};

// test __resolve using undefined method names
myObject.someMethod(); // output: Method someMethod was called
myObject.someOtherMethod(); //output: Method someOtherMethod was called
```

Usage 3 : L'exemple suivant développe l'exemple précédent en permettant de placer les méthodes résolues en mémoire cache. En plaçant les méthodes en mémoire cache, `__resolve` n'est appelé qu'une seule fois pour chaque méthode concernée. Ceci autorise la *construction paresseuse* des méthodes d'objet. La construction paresseuse est une technique d'optimisation qui reporte la création, ou *construction*, des méthodes jusqu'à la première utilisation d'une méthode.

```
// instantiate a new object
var myObject:Object = new Object();
// define a function for __resolve to call
myObject.myFunction = function(name) {
  trace("Method "+name+" was called");
};
// define the __resolve function
myObject.__resolve = function(name) {
  trace("Resolve called for "+name); // to check when __resolve is called
  // Not only call the function, but also save a reference to it
  var f:Function = function () {
    this.myFunction(name);
  };
  // create a new object method and assign it the reference
  this[name] = f;
  // return the reference
  return f;
};
// test __resolve using undefined method names
// __resolve will only be called once for each method name
myObject.someMethod(); // calls __resolve
myObject.someMethod(); // does not call __resolve because it is now defined
myObject.someOtherMethod(); // calls __resolve
myObject.someOtherMethod(); // does not call __resolve, no longer undefined
```

Usage 4 : L'exemple suivant développe l'exemple précédent en réservant un nom de méthode, `onStatus()`, pour l'utilisation locale de façon à ce qu'il ne soit pas résolu de la même façon que les autres propriétés non définies. Le code ajouté figure en gras.

```
// instantiate a new object
var myObject:Object = new Object();
// define a function for __resolve to call
myObject.myFunction = function(name) {
  trace("Method "+name+" was called");
};
// define the __resolve function
myObject.__resolve = function(name) {
  // reserve the name "onStatus" for local use
  if (name == "onStatus") {
    return undefined;
  }
  trace("Resolve called for "+name); // to check when __resolve is called
  // Not only call the function, but also save a reference to it
  var f:Function = function () {
    this.myFunction(name);
  };
  // create a new object method and assign it the reference
  this[name] = f;
  // return the reference
  return f;
};
// test __resolve using the method name "onStatus"
trace(myObject.onStatus("hello"));
// output: undefined
```

Usage 5 : L'exemple suivant développe l'exemple précédent en créant un foncteur qui accepte des paramètres. Cet exemple utilise de façon intensive l'objet arguments et applique plusieurs méthodes de la classe Array.

```
// instantiate a new object
var myObject:Object = new Object();

// define a generic function for __resolve to call
myObject.myFunction = function (name) {
  arguments.shift();
  trace("Method " + name + " was called with arguments: " +
  arguments.join(','));
};

// define the __resolve function
myObject.__resolve = function (name) {
  // reserve the name "onStatus" for local use
  if (name == "onStatus") {
    return undefined;
  }
  var f:Function = function () {
    arguments.unshift(name);
    this.myFunction.apply(this, arguments);
  };
  // create a new object method and assign it the reference
```

```
  this[name] = f;
  // return the reference to the function
  return f;
};

// test __resolve using undefined method names with parameters
myObject.someMethod("hello");
// output: Method someMethod was called with arguments: hello

myObject.someOtherMethod("hello","world");
// output: Method someOtherMethod was called with arguments: hello,world
```

### Voir également

`arguments`, `Array`

## toString (méthode Object.toString)

```
public toString() : String
```

Convertit l'objet spécifié en chaîne et renvoie cette dernière.

**Disponibilité :** ActionScript 1.0 ; Flash Player 5

### Valeur renvoyée

`String` - Chaîne.

### Exemple

Cet exemple affiche la valeur renvoyée pour toString() sur un objet générique :

```
var myObject:Object = new Object();
trace(myObject.toString()); // output: [object Object]
```

Cette méthode peut être remplacée pour renvoyer une valeur plus significative. Les exemples suivants indiquent que cette méthode a été remplacée pour les classes intégrées Date, Array et Number :

```
// Date.toString() returns the current date and time
var myDate:Date = new Date();
trace(myDate.toString()); // output: [current date and time]

// Array.toString() returns the array contents as a comma-delimited string
var myArray:Array = new Array("one", "two");
trace(myArray.toString()); // output: one,two

// Number.toString() returns the number value as a string
// Because trace() won't tell us whether the value is a string or number
// we will also use typeof() to test whether toString() works.
var myNumber:Number = 5;
```

```
trace(typeof (myNumber)); // output: number
trace(myNumber.toString()); // output: 5
trace(typeof (myNumber.toString())); // output: string
```

L'exemple suivant indique comment remplacer`toString()` dans une classe personnalisée. Créez tout d'abord un fichier texte appelé *Vehicle.as* contenant la définition de la classe Vehicle et placez-le dans le dossier Classes figurant dans le dossier Configuration.

```
// contents of Vehicle.as
class Vehicle {
  var numDoors:Number;
  var color:String;
  function Vehicle(param_numDoors:Number, param_color:String) {
  this.numDoors = param_numDoors;
  this.color = param_color;
  }
  function toString():String {
  var doors:String = "door";
  if (this.numDoors > 1) {
    doors += "s";
  }
  return ("A vehicle that is " + this.color + " and has " + this.numDoors +
  " " + doors);
  }
}

// code to place into a FLA file
var myVehicle:Vehicle = new Vehicle(2, "red");
trace(myVehicle.toString());
// output: A vehicle that is red and has 2 doors

// for comparison purposes, this is a call to valueOf()
// there is no primitive value of myVehicle, so the object is returned
// giving the same output as toString().
trace(myVehicle.valueOf());
// output: A vehicle that is red and has 2 doors
```

# unwatch (méthode Object.unwatch)

```
public unwatch(name:String) : Boolean
```

Supprime un point d'observation créé par `Object.watch()`. Cette méthode renvoie la valeur `true` si le point d'observation a été supprimé avec succès, et `false` dans tous les autres cas.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

## Paramètres

*`name`*`:String` - Chaîne ; nom de la propriété d'objet qui ne doit plus être supervisée.

### Valeur renvoyée

`Boolean` - Valeur booléenne : `true` si le point d'observation a été supprimé avec succès, `false` dans tous les autres cas.

### Exemple

Consultez l'exemple relatif à `Object.watch()`.

### Voir également

`watch (méthode Object.watch)`, `addProperty (méthode Object.addProperty)`

## valueOf (méthode Object.valueOf)

```
public valueOf() : Object
```

Renvoie la valeur primitive de l'objet spécifié. Si l'objet n'a pas de valeur primitive, il est renvoyé.

**Disponibilité :** ActionScript 1.0 ; Flash Player 5

### Valeur renvoyée

`Object` - Valeur primitive de l'objet spécifié ou l'objet lui-même.

### Exemple

L'exemple suivant affiche la valeur renvoyée de valueOf() pour un objet générique (qui n'a pas de valeur primitive) et la compare à la valeur renvoyée par toString(). Créez tout d'abord un objet générique : Dans une deuxième étape, créez un nouvel objet date défini sur le 1er février 2004, 8:15. La méthode toString() renvoie l'heure actuelle en clair. La méthode valueOf() renvoie la valeur primitive en millisecondes. Dans une troisième étape, créez un nouvel objet tableau contenant deux éléments simples. toString() et valueOf() renvoient tous les deux la même valeur : one,two :

```
// Create a generic object
var myObject:Object = new Object();
trace(myObject.valueOf()); // output: [object Object]
trace(myObject.toString()); // output: [object Object]
```

Les exemples suivants affichent les valeurs renvoyées pour les classes intégrées Date et Array, et les comparent aux valeurs renvoyées de `Object.toString()`:

```
// Create a new Date object set to February 1, 2004, 8:15 AM
// The toString() method returns the current time in human-readable form
// The valueOf() method returns the primitive value in milliseconds
var myDate:Date = new Date(2004,01,01,8,15);
trace(myDate.toString()); // output: Sun Feb 1 08:15:00 GMT-0800 2004
```

```
trace(myDate.valueOf()); // output: 1075652100000

// Create a new Array object containing two simple elements
// In this case both toString() and valueOf() return the same value: one,two
var myArray:Array = new Array("one", "two");
trace(myArray.toString()); // output: one,two
trace(myArray.valueOf()); // output: one,two
```

Voir l'exemple relatif à `Object.toString()` pour obtenir un exemple de la valeur renvoyée par `Object.valueOf()` pour une classe personnalisée remplaçant `toString()`.

### Voir également

`toString (méthode Object.toString)`

## watch (méthode Object.watch)

```
public watch(name:String, callback:Function, [userData:Object]) : Boolean
```

Enregistre un gestionnaire d'événements à appeler lorsque la propriété spécifiée d'un objet ActionScript change. Lorsque la propriété est modifiée, le gestionnaire d'événements est appelé avec `myObject` comme objet contenant.

Vous pouvez utiliser l'instruction `return` dans la définition de votre méthode de rappel `callback` pour affecter la valeur de la propriété que vous observez. La valeur renvoyée par votre méthode de rappel `callback` est affectée à la propriété de l'objet observé. La valeur que vous choisissez de renvoyer diffère selon que vous souhaitez surveiller, modifier ou empêcher toute modification apportée à la propriété :

- Si vous vous contentez de surveiller la propriété, renvoyez le paramètre `newVal`.
- Si vous modifiez la valeur de la propriété, renvoyez votre propre valeur.
- Si vous souhaitez empêcher toute modification apportée à la propriété, renvoyez le paramètre `oldVal`.

Si la méthode de rappel `callback` que vous définissez ne dispose pas d'instruction `return` la propriété de l'objet observé est définie sur une valeur `undefined`.

Un point d'observation peut filtrer (ou annuler) l'affectation de la valeur, en renvoyant un paramètre `newval` (ou `oldval`) modifié. Si vous supprimez une propriété pour laquelle un point d'observation a été défini, ce dernier ne disparaît pas. Si vous recréez la propriété ultérieurement, le point d'observation est toujours en vigueur. Pour supprimer un point d'observation, utilisez la méthode `Object.unwatch`.

Un seul point d'observation peut être enregistré sur une propriété. Les prochains appels de `Object.watch()` sur la même propriété remplacent le point d'observation d'origine.

La méthode `Object.watch()` se comporte de la même façon que la fonction `Object.watch()` à partir de JavaScript 1.2. La principale différence est liée au paramètre `userData`, qui est un ajout de Flash à `Object.watch()` que Netscape Navigator ne prend pas en charge. Vous pouvez transmettre le paramètre `userData` au gestionnaire d'événements et l'utiliser dans celui-ci.

La méthode `Object.watch()` ne peut pas observer les propriétés de lecture/définition. Les propriétés de lecture/définition fonctionnent par l'intermédiaire d'une *évaluation " paresseuse "* : la valeur de la propriété est déterminée lors de l'interrogation de la propriété. L'évaluation « paresseuse » est souvent efficace car la propriété n'est pas constamment mise à jour ; elle est évaluée en cas de besoin. Cependant, `Object.watch()` doit évaluer une propriété afin de déterminer s'il convient d'appeler la fonction de rappel `callback`. Pour pouvoir travailler avec une propriété de lecture/définition, `Object.watch()` doit évaluer la propriété en permanence, ce qui n'est pas efficace.

Généralement, les propriétés ActionScript prédéfinies, telles que `_x`, `_y`, `_width`, et `_height`, sont des propriétés de lecture/définition ne pouvant pas être observées par `Object.watch()`.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

## Paramètres

*`name`*`:String` - Chaîne ; nom de la propriété d'objet à observer.

*`callback`*`:Function` - Fonction à appeler lorsque la propriété observée change. Ce paramètre est un objet de fonction et non pas un nom de fonction exprimé en tant que chaîne. La forme de rappel `callback` est `callback(prop, oldVal, newVal, userData)`.

*`userData`*`:Object` [facultatif] - Elément arbitraire des données ActionScript qui est transmis à la méthode `callback`. Si le paramètre `userData` est omis, `undefined` est transmis à la méthode callback.

## Valeur renvoyée

`Boolean` - Valeur booléenne : `true` si le point d'observation a été créé avec succès, `false` dans tous les autres cas.

## Exemple

L'exemple suivant utilise `watch()` pour vérifier si la propriété `speed` a une valeur supérieure à la limitation de vitesse :

```
// Create a new object
var myObject:Object = new Object();

// Add a property that tracks speed
myObject.speed = 0;
```

```
// Write the callback function to be executed if the speed property changes
var speedWatcher:Function = function(prop, oldVal, newVal, speedLimit) {
  // Check whether speed is above the limit
  if (newVal > speedLimit) {
  trace ("You are speeding.");
  }
  else {
  trace ("You are not speeding.");
  }

  // Return the value of newVal.
  return newVal;
}
// Use watch() to register the event handler, passing as parameters:
// - the name of the property to watch: "speed"
// - a reference to the callback function speedWatcher
// - the speedLimit of 55 as the userData parameter
myObject.watch("speed", speedWatcher, 55);

// set the speed property to 54, then to 57
myObject.speed = 54; // output: You are not speeding
myObject.speed = 57; // output: You are speeding

// unwatch the object
myObject.unwatch("speed");
myObject.speed = 54; // there should be no output
```

### Voir également

addProperty (méthode Object.addProperty), unwatch (méthode Object.unwatch)

# Point (flash.geom.Point)

```
Object
  |
  +-flash.geom.Point
```

public class ***Point***
extends Object

La classe Point représente un emplacement dans un système de coordonnées à deux dimensions, où *x* représente l'axe horizontal et *y* l'axe vertical.

Le code suivant crée un point à (0,0) :

```
var myPoint:Point = new Point();
```

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

## Résumé des propriétés

| Modificateurs | Propriété | Description |
|---|---|---|
| | `length:Number` | Longueur du segment de ligne de (0,0) à ce point. |
| | `x:Number` | Coordonnées horizontales du point. |
| | `y:Number` | Coordonnée verticale du point. |

*Propriétés héritées de la classe Object*

`constructeur (propriété Object.constructor), __proto__ (Object.__proto__ property), prototype (Object.prototype, propriété), __resolve (Object.__resolve, propriété)`

## Résumé des constructeurs

| Signature | Description |
|---|---|
| `Point(x:Number, y:Number)` | Crée un nouveau point. |

## Résumé de la méthode

| Modificateurs | Signature | Description |
|---|---|---|
| | `add(v:Point) : Point` | Ajoute les coordonnées d'un autre point à celles de ce point pour créer un nouveau point. |
| | `clone() : Point` | Crée une copie de cet objet Point. |
| `static` | `distance(pt1:Point, pt2:Point) : Number` | Renvoie la distance entre pt1 et pt2. |
| | `equals(toCompare:Object) : Boolean` | Détermine si deux points sont égaux. |
| `static` | `interpolate(pt1:Point, pt2:Point, f:Number) : Point` | Détermine un point entre deux points spécifiés. |
| | `normalize(length:Number) : Void` | Met à l'échelle le segment de ligne entre (0,0) et le point actuel en fonction d'une longueur définie. |
| | `offset(dx:Number, dy:Number) : Void` | Décale l'objet Point du montant spécifié. |
| `static` | `polar(len:Number, angle:Number) : Point` | Convertit une paire de coordonnées polaires en coordonnées cartésiennes. |

| Modificateurs | Signature | Description |
|---|---|---|
| | `subtract(v:Point) : Point` | Soustrait les coordonnées d'un autre point à celles de ce point pour créer un nouveau point. |
| | `toString() : String` | Renvoie une chaîne qui contient les valeurs des coordonnées *x* et *y*. |

*Méthodes héritées de la classe Object*

`addProperty (méthode Object.addProperty)`, `hasOwnProperty (méthode Object.hasOwnProperty)`, `isPropertyEnumerable (méthode Object.isPropertyEnumerable)`, `isPrototypeOf (méthode Object.isPrototypeOf)`, `registerClass (méthode Object.registerClass)`, `toString (méthode Object.toString)`, `unwatch (méthode Object.unwatch)`, `valueOf (méthode Object.valueOf)`, `watch (méthode Object.watch)`

# add (méthode Point.add)

`public add(v:Point) : Point`

Ajoute les coordonnées d'un autre point à celles de ce point pour créer un nouveau point.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

## Paramètres

*v*:`flash.geom.Point` - Point à ajouter.

## Valeur renvoyée

`flash.geom.Point` - Nouveau point.

## Exemple

L'exemple suivant crée un objet Point `resultPoint` en ajoutant `point_2` à `point_1`.

```
import flash.geom.Point;
var point_1:Point = new Point(4, 8);
var point_2:Point = new Point(1, 2);
var resultPoint:Point = point_1.add(point_2);
trace(resultPoint.toString()); // (x=5, y=10)
```

# clone (méthode Point.clone)

`public clone() : Point`

Crée une copie de cet objet Point.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

### Valeur renvoyée

`flash.geom.Point` - Nouvel objet Point.

### Exemple

L'exemple suivant crée une copie de l'objet Point appelé `clonedPoint` à partir des valeurs trouvées dans l'objet `myPoint`. L'objet `clonedPoint` contient toutes les valeurs de `myPoint`, mais il ne s'agit pas du même objet.

```
import flash.geom.Point;
var myPoint:Point = new Point(1, 2);
var clonedPoint:Point = myPoint.clone();
trace(clonedPoint.x); // 1
trace(clonedPoint.y); // 2
trace(myPoint.equals(clonedPoint)); // true
trace(myPoint === clonedPoint); // false
```

## distance (méthode Point.distance)

```
public static distance(pt1:Point, pt2:Point) : Number
```

Renvoie la distance entre pt1 et pt2.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

### Paramètres

*`pt1`*`:flash.geom.Point` - Premier point.

*`pt2`*`:flash.geom.Point` - Deuxième point.

### Valeur renvoyée

`Number` - Distance entre le premier et le deuxième point.

### Exemple

L'exemple suivant crée `point_1` et `point_2`, puis détermine la distance entre ces derniers (`distanceBetween`).

```
import flash.geom.Point;
var point_1:Point = new Point(-5, 0);
var point_2:Point = new Point(5, 0);
var distanceBetween:Number = Point.distance(point_1, point_2);
trace(distanceBetween); // 10
```

# equals (méthode Point.equals)

```
public equals(toCompare:Object) : Boolean
```

Détermine si deux points sont égaux. Deux points sont considérés comme égaux s'ils ont les mêmes valeurs *x* et *y*.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

## Paramètres

*toCompare*:Object - Point à comparer.

## Valeur renvoyée

Boolean - Si l'objet est égal à cet objet Point, `true`; s'il n'est pas égal, `false`.

## Exemple

L'exemple suivant détermine si les valeurs d'un point sont égales aux valeurs d'un autre point. Si les objets sont identiques, `equals()` ne renvoie pas le même résultat que celui envoyé par l'opérateur de stricte égalité (===).

```
import flash.geom.Point;
var point_1:Point = new Point(1, 2);
var point_2:Point = new Point(1, 2);
var point_3:Point = new Point(4, 8);
trace(point_1.equals(point_2)); // true
trace(point_1.equals(point_3)); // false
trace(point_1 === point_2); // false
trace(point_1 === point_3); // false
```

# interpolate (méthode Point.interpolate)

```
public static interpolate(pt1:Point, pt2:Point, f:Number) : Point
```

Détermine un point entre deux points spécifiés.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

## Paramètres

*pt1*:flash.geom.Point - Premier point.

*pt2*:flash.geom.Point - Deuxième point.

*f*:Number - Niveau d'interpolation entre les deux points. Indique l'emplacement du nouveau point sur la ligne reliant pt1 et pt2. Si f=0, pt1 est renvoyé ; si f=1, pt2 est renvoyé.

### Valeur renvoyée

`flash.geom.Point` - Nouveau point interpolé.

### Exemple

L'exemple suivant situe le point interpolé (`interpolatedPoint`) à mi-chemin (50 %) entre `point_1` et `point_2`.

```
import flash.geom.Point;
var point_1:Point = new Point(-100, -100);
var point_2:Point = new Point(50, 50);
var interpolatedPoint:Point = Point.interpolate(point_1, point_2, .5);
trace(interpolatedPoint.toString()); // (x=-25, y=-25)
```

## length (propriété Point.length)

```
longueur publique : Number
```

La longueur du segment de ligne de (0,0) à ce point.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

### Exemple

L'exemple suivant crée un objet Point, `myPoint`, et détermine la longueur d'une ligne de (0, 0) à ce Point.

```
import flash.geom.Point;
var myPoint:Point = new Point(3,4);
trace(myPoint.length); // 5
```

### Voir également

`polar (méthode Point.polar)`

## normalize (méthode Point.normalize)

```
public normalize(length:Number) : Void
```

Met à l'échelle le segment de ligne entre (0,0) et le point actuel en fonction d'une longueur définie.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

### Paramètres

*`length`*`:Number` - Valeur de mise à l'échelle. Si, par exemple, le point actuel est (0,5) et que vous le normalisez à 1, les coordonnées suivantes sont renvoyées pour le point (0,1).

### Exemple

L'exemple suivant fait passer la longueur de l'objet `normalizedPoint` de 5 à 10.

```
import flash.geom.Point;
var normalizedPoint:Point = new Point(3, 4);
trace(normalizedPoint.length); // 5
trace(normalizedPoint.toString()); // (x=3, y=4)
normalizedPoint.normalize(10);
trace(normalizedPoint.length); // 10
trace(normalizedPoint.toString()); // (x=6, y=8)
```

### Voir également

`length (propriété Point.length)`

## offset (méthode Point.offset)

```
public offset(dx:Number, dy:Number) : Void
```

Décale l'objet Point du montant spécifié. La valeur de `dx` est ajoutée à la valeur d'origine de *x* pour créer la nouvelle valeur de *x*. La valeur de `dy` est ajoutée à la valeur d'origine de *y* pour créer la nouvelle valeur de *y*.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

### Paramètres

*dx*`:Number` - Montant de décalage de la coordonnée horizontale, *x*.

*dy*`:Number` - Montant de décalage de la coordonnée verticale, *y*.

### Exemple

L'exemple suivant décale la position d'un point en fonction des valeurs *x* et *y* spécifiées.

```
import flash.geom.Point;
var myPoint:Point = new Point(1, 2);
trace(myPoint.toString()); // (x=1, y=2)
myPoint.offset(4, 8);
trace(myPoint.toString()); // (x=5, y=10)
```

### Voir également

`add (méthode Point.add)`

# Point, constructeur

```
public Point(x:Number, y:Number)
```

Crée un nouveau point. Si vous ne transmettez pas de paramètres à cette méthode, un point est créé aux coordonnées (0,0).

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

## Paramètres

*x*:Number - Coordonnée horizontale. La valeur par défaut est 0.

*y*:Number - Coordonnée verticale. La valeur par défaut est 0.

## Exemple

Le premier exemple crée un objet point_1 avec le constructeur par défaut.

```
import flash.geom.Point;
var point_1:Point = new Point();
trace(point_1.x); // 0
trace(point_1.y); // 0
```

Le deuxième exemple crée un objet Point point_2 avec les coordonnées $x = 1$ et $y = 2$.

```
import flash.geom.Point;
var point_2:Point = new Point(1, 2);
trace(point_2.x); // 1
trace(point_2.y); // 2
```

# polar (méthode Point.polar)

```
public static polar(len:Number, angle:Number) : Point
```

Convertit une paire de coordonnées polaires en coordonnées cartésiennes.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

## Paramètres

*len*:Number - Coordonnée de longueur de la paire polaire.

*angle*:Number - Angle, en radians, de la paire polaire.

## Valeur renvoyée

flash.geom.Point - Point cartésien.

### Exemple

L'exemple suivant crée un objet Point `cartesianPoint` à partir des valeurs de `angleInRadians` et d'une longueur de ligne de 5. La valeur `angleInRadians` égale à Math.atan(3/4) est utilisée en raison des caractéristiques des triangles rectangles avec des côtés dont les rapports sont de 3:4:5.

```
import flash.geom.Point;
var len:Number = 5;
var angleInRadians:Number = Math.atan(3/4);
var cartesianPoint:Point = Point.polar(len, angleInRadians);
trace(cartesianPoint.toString()); // (x=4, y=3)
```

Lorsque les ordinateurs travaillent avec des nombres transcendants tels que pi, des erreurs d'arrondissement se produisent car l'arithmétique en virgule flottante est d'une précision qui n'est que finie. Si vous utilisez `Math.PI`, pensez à utiliser la fonction `Math.round()`, comme il est indiqué dans l'exemple suivant.

```
import flash.geom.Point;
var len:Number = 10;
var angleInRadians:Number = Math.PI;
var cartesianPoint:Point = Point.polar(len, angleInRadians);
trace(cartesianPoint.toString()); // should be (x=-10, y=0), but is (x=-10,
  y=1.22460635382238e-15)
trace(Math.round(cartesianPoint.y)); // 0
```

### Voir également

`length (propriété Point.length)`, `round (méthode Math.round)`

## subtract (méthode Point.subtract)

```
public subtract(v:Point) : Point
```

Soustrait les coordonnées d'un autre point à celles de ce point pour créer un nouveau point.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

### Paramètres

*v*`:flash.geom.Point` - Point à soustraire.

### Valeur renvoyée

`flash.geom.Point` - Nouveau point.

### Exemple

L'exemple suivant crée `point_3` en soustrayant `point_2` de `point_1`.

```
import flash.geom.Point;
```

```
var point_1:Point = new Point(4, 8);
var point_2:Point = new Point(1, 2);
var resultPoint:Point = point_1.subtract(point_2);
trace(resultPoint.toString()); // (x=3, y=6)
```

# toString (méthode Point.toString)

```
public toString() : String
```

Renvoie une chaîne qui contient les valeurs des coordonnées *x* et *y*. Elle prend la forme (*x*, *y*), un point avec les coordonnées 23,17 renvoie ainsi "(*x*=23, *y*=17)".

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

## Valeur renvoyée

`String` - Une chaîne.

## Exemple

L'exemple suivant crée un point et convertit ses valeurs en chaîne au format (x=x, y=y).

```
import flash.geom.Point;
var myPoint:Point = new Point(1, 2);
trace("myPoint: " + myPoint.toString()); // (x=1, y=2)
```

# x (Point.x, propriété)

```
public x : Number
```

Les coordonnées horizontales du point. La valeur par défaut est 0.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

## Exemple

L'exemple suivant crée un objet Point `myPoint` et définit la valeur coordonnée *x*.

```
import flash.geom.Point;
var myPoint:Point = new Point();
trace(myPoint.x); // 0
myPoint.x = 5;
trace(myPoint.x); // 5
```

# y (Point.y, propriété)

```
public y : Number
```

Coordonnée verticale du point. La valeur par défaut est 0.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

### Exemple

L'exemple suivant crée un objet Point `myPoint` et définit la valeur de la coordonnée *y*.

```
import flash.geom.Point;
var myPoint:Point = new Point();
trace(myPoint.y); // 0
myPoint.y = 5;
trace(myPoint.y); // 5
```

# PrintJob

```
Object
  |
  +-PrintJob
```

```
public class PrintJob
extends Object
```

La classe PrintJob permet de créer un contenu et de l'imprimer sur une ou plusieurs pages. Outre la mise à disposition de fonctions d'impression améliorées avec la méthode `print()`, cette classe permet de rendre le contenu dynamique hors écran, d'inviter les utilisateurs à l'aide d'une seule boîte de dialogue d'impression et d'imprimer un document non mis à l'échelle dans des proportions correspondant au contenu. Cette fonctionnalité est particulièrement utile pour rendre et imprimer le contenu dynamique, par exemple le contenu et le texte dynamique d'une base de données.

En outre, avec les propriétés remplies par `PrintJob.start()`, votre document peut lire les paramètres d'impression de l'utilisateur, tels que hauteur, largeur et orientation de la page : vous pouvez configurer votre document afin de mettre dynamiquement en forme le contenu Flash en fonction des paramètres de l'imprimante. Ces propriétés de disposition utilisateur sont en lecture seule et ne peuvent pas être changées par Flash Player.

**Disponibilité :** ActionScript 1.0 ; Flash Player 7

## Résumé des propriétés

| Modificateurs | Propriété | Description |
|---|---|---|
| | `orientation:String` [lecture seule] | Orientation de l'image pour l'impression. |
| | `pageHeight:Number` [lecture seule] | La hauteur de la zone imprimable réelle de la page, en points. |
| | `pageWidth:Number` [lecture seule] | La largeur de la zone imprimable réelle de la page, en points. |
| | `paperHeight:Number` [lecture seule] | La hauteur globale de la page, en points. |
| | `paperWidth:Number` [lecture seule] | La largeur globale de la page, en points. |

*Propriétés héritées de la classe Object*

`constructeur (propriété Object.constructor), __proto__ (Object.__proto__ property), prototype (Object.prototype, propriété), __resolve (Object.__resolve, propriété)`

## Résumé des constructeurs

| Signature | Description |
|---|---|
| `PrintJob()` | Crée un objet PrintJob permettant d'imprimer une ou plusieurs pages. |

Résumé de la méthode

| Modificateurs | Signature | Description |
|---|---|---|
| | `addPage(target:Object, [printArea:Object], [options:Object], [frameNum:Number]) : Boolean` | Envoie le niveau spécifié ou le clip en tant que page unique au spouleur d'impression. |
| | `send() : Void` | Utilisée conformément aux méthodes `PrintJob.start()` et `PrintJob.addPage()` pour envoyer les pages mises en file d'attente vers l'imprimante. |
| | `start() : Boolean` | Affiche les boîtes de dialogue d'impression du système d'exploitation et commence la mise en file d'attente. |

*Méthodes héritées de la classe Object*

`addProperty (méthode Object.addProperty)`, `hasOwnProperty (méthode Object.hasOwnProperty)`, `isPropertyEnumerable (méthode Object.isPropertyEnumerable)`, `isPrototypeOf (méthode Object.isPrototypeOf)`, `registerClass (méthode Object.registerClass)`, `toString (méthode Object.toString)`, `unwatch (méthode Object.unwatch)`, `valueOf (méthode Object.valueOf)`, `watch (méthode Object.watch)`

# addPage (méthode PrintJob.addPage)

```
public addPage(target:Object, [printArea:Object], [options:Object],
  [frameNum:Number]) : Boolean
```

Envoie le niveau spécifié ou le clip en tant que page unique au spouleur d'impression. Avant d'utiliser cette méthode, vous devez utiliser `PrintJob.start()`; après avoir appelé une ou plusieurs fois `PrintJob.addPage()` pour une tâche d'impression, vous devez utiliser `PrintJob.send()` pour envoyer les pages mises en file d'attente vers l'imprimante.

Si cette méthode renvoie la valeur `false` (par exemple, si vous n'avez pas appelé `PrintJob.start()` ou si l'utilisateur a annulé la tâche d'impression), les appels suivants de `PrintJob.addPage()` échoueront. Cependant, si les appels précédents de `PrintJob.addPage()` ont réussi, la commande finale de `PrintJob.send()` envoie les pages mises en file d'attente avec succès vers l'imprimante.

Si vous avez transmis une valeur à `printArea`, les coordonnées `xMin` et `yMin` établissent une correspondance avec le coin supérieur gauche (coordonnées 0,0) de la zone imprimable sur la page. La zone imprimable de l'utilisateur est décrite par les propriétés `pageHeight` et `pageWidth` en lecture seule définies par `PrintJob.start()`. Dans la mesure où elle s'aligne sur le coin supérieur gauche de la zone imprimable sur la page, l'impression est tronquée sur la droite et/ou dans la partie inférieure si la taille de la zone définie dans `printArea` est supérieure à celle de la zone imprimable sur la page. Si vous n'avez pas transmis de valeur pour `printArea` et si la taille de la scène est supérieure à celle de la zone imprimable, le même type de découpage est effectué.

Si vous souhaitez mettre un clip à l'échelle avant de l'imprimer, définissez ses propriétés `MovieClip._xscale` et `MovieClip._yscale` avant d'appeler cette méthode, puis redéfinissez-les sur leurs valeurs d'origine. L'échelle d'un clip ne dépend pas de `printArea`. Autrement dit, si vous spécifiez une zone d'impression dont la taille est 50 x 50 pixels, 2 500 pixels sont imprimés. Si vous avez mis le clip à l'échelle, 2 500 pixels sont également imprimés, mais le clip est imprimé à l'échelle retenue.

La fonction d'impression de Flash Player prend en charge les imprimantes PostScript et non PostScript. Les imprimantes non PostScript convertissent les vecteurs en bitmaps.

**Disponibilité :** ActionScript 1.0 ; Flash Player 7

## Paramètres

*`target`*`:Object` - Nombre ou chaîne ; niveau ou nom d'occurrence du clip à imprimer. Transmettez un nombre pour spécifier un niveau (par exemple, 0 correspond à l'animation `_root`), ou une chaîne (entre guillemets [""]) pour spécifier le nom d'occurrence du clip.

*`printArea`*`:Object` [facultatif] - Objet qui spécifie la zone à imprimer, au format suivant :

`{xMin:`*topLeft*, xMax:*topRight*, yMin:*bottomLeft*, yMax:*bottomRight*`}`

Les coordonnées que vous spécifiez pour `printArea` représentent les pixels de l'écran par rapport au point d'alignement du clip `_root` (si `target` = 0) ou du niveau du clip spécifié par `target`. Vous devez fournir les quatre coordonnées. La largeur (`xMax-xMin`) et la hauteur (`yMax-yMin`) doivent chacune être supérieures à 0.

Les points sont des unités de mesure d'impression et les pixels sont des unités de mesure d'écran. Les points ont une taille physique fixe (1/72 de pouce). Par contre, la taille des pixels dépend de la résolution de l'écran donné. Vous pouvez utiliser les équivalences suivantes pour convertir les pouces ou les centimètres en twips ou points (un twip correspond à 1/20ème de point) :

- 1 point = 1/72 pouce = 20 twips
- 1 pouce = 72 points = 1 440 twips
- 1 cm = 567 twips

Il n'est pas possible de convertir de façon fiable les pixels en points et inversement. Le taux de conversion dépend de l'écran et de sa résolution. Si, par exemple, l'écran est conçu pour afficher 72 pixels par pouce, un point est égal à un pixel.

*Remarque :* Si vous avez précédemment utilisé `print()`, `printAsBitmap()`, `printAsBitmapNum()`, ou `printNum()` pour imprimer à partir de Flash, il se peut que vous ayez utilisé une `#b` étiquette d'image pour spécifier la zone d'impression. Avec la méthode `addPage()`, utilisez le paramètre `printArea` pour spécifier la zone d'impression; `#b` Les étiquettes d'image sont ignorées.

Si vous omettez le paramètre `printArea` ou s'il est transmis de façon incorrecte, l'ensemble de la scène de `target` est imprimé. Si vous ne souhaitez pas spécifier de valeur pour `printArea` mais souhaitez spécifier une valeur pour `options` ou `frameNumber`, transmettez `null` pour `printArea`.

*`options`*`:Object` [facultatif] - Paramètre permettant d'indiquer si l'impression doit être de type vectoriel ou bitmap, au format suivant :

`{printAsBitmap:`*`Boolean`*`}`

La valeur par défaut est `false`, ce qui représente une demande d'impression vectorielle. Pour imprimer `target` en tant que bitmap, transmettez `true` pour printAsBitmap. Tenez compte des suggestions suivantes lors de la détermination de la valeur à utiliser:

- Lorsque le contenu à imprimer contient une image bitmap, utilisez `{printAsBitmap:true}` pour inclure les effets de transparence et de couleur.
- Si le contenu ne contient pas d'images bitmap, omettez ce paramètre ou utilisez `{printAsBitmap:false}` pour imprimer le contenu au format vectoriel et à une qualité supérieure.

Si `options` est omis ou transmis de façon incorrecte, l'impression vectorielle s'applique. Si vous ne souhaitez pas spécifier de valeur pour `options` mais souhaitez spécifier une valeur pour `frameNumber`, transmettez `null` pour `options`.

*`frameNum`*`:Number` [facultatif] - Nombre qui permet de spécifier l'image à imprimer. Le fait de transmettre `frameNumber` n'implique pas l'appel d'ActionScript à partir de cette image. Si vous omettez ce paramètre, l'image actuelle de `target` est imprimée.

*Remarque :* Si vous avez précédemment utilisé `print()`, `printAsBitmap()`, `printAsBitmapNum()`, ou `printNum()` pour imprimer à partir de Flash, il se peut que vous ayez utilisé une `#p` étiquette d'image sur plusieurs images pour spécifier les pages à imprimer. Pour utiliser `PrintJob.addPage()` afin d'imprimer plusieurs images, vous devez émettre une commande `PrintJob.addPage()` pour chaque image ; les étiquettes d'image `#p` sont ignorées. Pour savoir comment procéder par programmation, consultez la section Exemple.

## Valeur renvoyée

`Boolean` - Valeur booléenne : `true` si la page est envoyée avec succès vers le spouleur d'impression ; `false` dans tous les autres cas.

## Exemple

L'exemple suivant présente plusieurs façons d'émettre la commande `addPage()` :

```
my_btn.onRelease = function()
{
  var pageCount:Number = 0;

  var my_pj:PrintJob = new PrintJob();

  if (my_pj.start())
  {
  // Print entire current frame of the _root movie in vector format
  if (my_pj.addPage(0)){
    pageCount++;

    // Starting at 0,0, print an area 400 pixels wide and 500 pixels high
    // of the current frame of the _root movie in vector format
    if (my_pj.addPage(0, {xMin:0,xMax:400,yMin:0,yMax:500})){
    pageCount++;

    // Starting at 0,0, print an area 400 pixels wide and 500 pixels high
    // of frame 1 of the _root movie in bitmap format
    if (my_pj.addPage(0, {xMin:0,xMax:400,yMin:0,yMax:500},
      {printAsBitmap:true}, 1)){
      pageCount++;

      // Starting 50 pixels to the right of 0,0 and 70 pixels down,
      // print an area 500 pixels wide and 600 pixels high
      // of frame 4 of level 5 in vector format
      if (my_pj.addPage(5, {xMin:50,xMax:550,yMin:70,yMax:670},null, 4)){
      pageCount++;

      // Starting at 0,0, print an area 400 pixels wide
      // and 400 pixels high of frame 3 of the "dance_mc" movie clip
      // in bitmap format
      if (my_pj.addPage("dance_mc",
        {xMin:0,xMax:400,yMin:0,yMax:400},{printAsBitmap:true}, 3)){
        pageCount++;

        // Starting at 0,0, print an area 400 pixels wide
        // and 600 pixels high of frame 3 of the "dance_mc" movie clip
        // in vector format at 50% of its actual size
        var x:Number = dance_mc._xscale;
        var y:Number = dance_mc._yscale;
        dance_mc._xscale = 50;
```

```
          dance_mc._yscale = 50;

          if (my_pj.addPage("dance_mc",
            {xMin:0,xMax:400,yMin:0,yMax:600},null, 3)){
          pageCount++;
          }
          dance_mc._xscale = x;
          dance_mc._yscale = y;
        }
        }
      }
      }
    }
    }

    // If addPage() was successful at least once, print the spooled pages.
    if (pageCount > 0){
    my_pj.send();
    }
    delete my_pj;
}
```

### Voir également

`send (méthode PrintJob.send)`, `start (méthode PrintJob.start)`

## orientation (PrintJob.orientation, propriété)

```
orientation publique : String [lecture seule]
```

Orientation de l'image pour l'impression. Cette propriété peut être soit `"landscape"`, soit `"portrait"`. Cette propriété n'est disponible qu'après un appel de la méthode `PrintJob.start()`.

**Disponibilité :** ActionScript 1.0 ; Flash Player 7

## pageHeight (propriété PrintJob.pageHeight)

```
public pageHeight : Number [lecture seule]
```

La hauteur de la zone imprimable réelle de la page, en points. Les marges éventuellement définies par l'utilisateur sont ignorées. Cette propriété n'est disponible qu'après un appel de la méthode `PrintJob.start()`.

**Disponibilité :** ActionScript 1.0 ; Flash Player 7

## pageWidth (propriété PrintJob.pageWidth)

```
public pageWidth : Number [lecture seule]
```

La largeur de la zone imprimable réelle de la page, en points. Les marges éventuellement définies par l'utilisateur sont ignorées. Cette propriété n'est disponible qu'après un appel de la méthode `PrintJob.start()`.

**Disponibilité :** ActionScript 1.0 ; Flash Player 7

## paperHeight (propriété PrintJob.paperHeight)

```
public paperHeight : Number [lecture seule]
```

La hauteur globale de la page, en points. Cette propriété n'est disponible qu'après un appel de la méthode `PrintJob.start()`.

**Disponibilité :** ActionScript 1.0 ; Flash Player 7

## paperWidth (propriété PrintJob.paperWidth)

```
public paperWidth : Number [lecture seule]
```

La largeur globale de la page, en points. Cette propriété n'est disponible qu'après un appel de la méthode `PrintJob.start()`.

**Disponibilité :** ActionScript 1.0 ; Flash Player 7

## PrintJob, constructeur

```
public PrintJob()
```

Crée un objet PrintJob permettant d'imprimer une ou plusieurs pages.

Pour implémenter une tâche d'impression, utilisez les méthodes suivantes en séquence. Vous devez placer toutes les commandes propres à une tâche d'impression spécifique dans la même image, du constructeur jusqu'à Delete en passant `PrintJob.send()`. Remplacez les paramètres `[params]` des appels de la méthode `my_pj.addPage()` par vos paramètres personnalisés.

```
// create PrintJob object
var my_pj:PrintJob = new PrintJob();

// display print dialog box, but only initiate the print job
// if start returns successfully.
if (my_pj.start()) {

  // use a variable to track successful calls to addPage
  var pagesToPrint:Number = 0;
```

```
  // add specified area to print job
  // repeat once for each page to be printed
  if (my_pj.addPage([params])) {
  pagesToPrint++;
  }
  if (my_pj.addPage([params])) {
  pagesToPrint++;
  }
  if (my_pj.addPage([params])) {
  pagesToPrint++;
  }

  // send pages from the spooler to the printer, but only if one or more
  // calls to addPage() was successful. You should always check for
  successful
  // calls to start() and addPage() before calling send().
  if (pagesToPrint > 0) {
  my_pj.send();  // print page(s)
  }
}

// clean up
delete my_pj;  // delete object
```

Vous ne pouvez pas créer un deuxième objet PrintJob si le premier est toujours actif. Vous ne pouvez pas créer un deuxième objet PrintJob (en appelant `new PrintJob()`) si le premier objet PrintJob est toujours actif ; le deuxième objet PrintJob ne sera pas créé.

**Disponibilité :** ActionScript 1.0 ; Flash Player 7

## Exemple

Pour plus d'informations, consultez `PrintJob.addPage()`.

## Voir également

`addPage (méthode PrintJob.addPage)`, `send (méthode PrintJob.send)`, `start (méthode PrintJob.start)`

# send (méthode PrintJob.send)

```
public send() : Void
```

Utilisée conformément aux méthodes `PrintJob.start()` et `PrintJob.addPage()` pour envoyer les pages mises en file d'attente vers l'imprimante. Puisque les appels de `PrintJob.send()` ne peuvent pas aboutir si les appels connexes de `PrintJob.start()` et `PrintJob.addpage()` ont échoué, vous devez vous assurer que les appels de `PrintJob.addpage()` et de `PrintJob.start()` ont réussi avant d'appeler `PrintJob.send()`:

```
var my_pj:PrintJob = new PrintJob();
if (my_pj.start()) {
  if (my_pj.addPage(this)) {
  my_pj.send();
  }
}
delete my_pj;
```

**Disponibilité :** ActionScript 1.0 ; Flash Player 7

## Exemple

Reportez-vous à `PrintJob.addPage()` et à `PrintJob.start()`.

## Voir également

`addPage (méthode PrintJob.addPage)`, `start (méthode PrintJob.start)`

# start (méthode PrintJob.start)

```
public start() : Boolean
```

Affiche les boîtes de dialogue d'impression du système d'exploitation et commence la mise en file d'attente. Les boîtes de dialogue d'impression permettent à l'utilisateur de modifier les paramètres d'impression. Lorsque le retour de la méthode `PrintJob.start()` est correct, les propriétés en lecture seule suivantes sont renseignées, représentant les paramètres d'impression de l'utilisateur :

| Propriété | Type | Unités | Remarques |
|---|---|---|---|
| PrintJob.paperHeight | Number | Points | Hauteur générale du papier. |
| PrintJob.paperWidth | Number | Points | Largeur générale du papier. |

| Propriété | Type | Unités | Remarques |
|---|---|---|---|
| PrintJob.pageHeight | Number | Points | Hauteur de la zone imprimable réelle de la page. Les marges éventuellement définies par l'utilisateur sont ignorées. |
| PrintJob.pageWidth | Number | Points | Largeur de la zone imprimable réelle de la page. Les marges éventuellement définies par l'utilisateur sont ignorées. |
| PrintJob.orientation | Number | Points | "Portrait" ou "landscape" (paysage). |

Une fois que l'utilisateur a cliqué sur OK dans la boîte de dialogue Imprimer, le lecteur commence à mettre en attente une tâche d'impression vers le système d'exploitation. Il est recommandé d'exécuter toutes les commandes ActionScript qui affectent l'impression et utiliser les commandes `PrintJob.addPage()` pour envoyer des pages vers le spouleur. Vous pouvez utiliser les propriétés height, width et orientation en lecture seule renseignées par cette méthode pour mettre en forme l'impression.

Dans la mesure où l'utilisateur accède à des informations telles que « Impression de la page 1 » juste après avoir cliqué sur OK, il est recommandé d'appeler les commandes `PrintJob.addPage()` et `PrintJob.send()` dès que possible.

Si cette méthode renvoie la valeur `false` (par exemple, lorsque l'utilisateur clique sur Annuler au lieu de OK dans la boîte de dialogue Imprimer du système d'exploitation), les prochains appels de `PrintJob.addPage()` et de `PrintJob.send()` échoueront. Toutefois, si vous testez cette valeur renvoyée et que vous n'envoyez pas de commandes `PrintJob.addPage()` ensuite, vous devrez quand même supprimer l'objet PrintJob afin de vous assurer que le spouleur d'impression est supprimé, comme indiqué dans l'exemple suivant :

```
var my_pj:PrintJob = new PrintJob();
var myResult:Boolean = my_pj.start();
  if(myResult) {
  // addPage() and send() statements here
  }
delete my_pj;
```

**Disponibilité :** ActionScript 1.0 ; Flash Player 7

## Valeur renvoyée

`Boolean` - Valeur booléenne : `true` si l'utilisateur clique sur OK lorsque les boîtes de dialogue d'impression apparaissent ; `false` si l'utilisateur clique sur Annuler ou si une erreur se produit.

## Exemple

L'exemple suivant indique comment utiliser la valeur de la propriété orientation pour régler le format d'impression :

```
// create PrintJob object
var my_pj:PrintJob = new PrintJob();

// display print dialog box
if (my_pj.start()) {
  // boolean to track whether addPage succeeded, change this to a counter
  // if more than one call to addPage is possible
  var pageAdded:Boolean = false;

  // check the user's printer orientation setting
  // and add appropriate print area to print job
  if (my_pj.orientation == "portrait") {
  // Here, the printArea measurements are appropriate for an 8.5" x 11"
  // portrait page.
  pageAdded = my_pj.addPage(this,{xMin:0,xMax:600,yMin:0,yMax:800});
  }
  else {
  // my_pj.orientation is "landscape".
  // Now, the printArea measurements are appropriate for an 11" x 8.5"
  // landscape page.
  pageAdded = my_pj.addPage(this,{xMin:0,xMax:750,yMin:0,yMax:600});
  }

  // send pages from the spooler to the printer
  if (pageAdded) {
  my_pj.send();
  }
}

// clean up
delete my_pj;
```

## Voir également

`addPage (méthode PrintJob.addPage)`, `send (méthode PrintJob.send)`

# Rectangle (flash.geom.Rectangle)

```
Object
  |
  +-flash.geom.Rectangle
```

```
public class Rectangle
extends Object
```

La classe Rectangle permet de créer et modifier des objets Rectangle. Un objet Rectangle est une zone définie par sa position, indiquée par son angle supérieur gauche (*x*, *y*), ainsi que par sa largeur et sa hauteur. Procédez avec prudence lors de la conception de ces zones : si un rectangle est décrit comme ayant son point supérieur gauche aux coordonnées 0,0 pour une hauteur de 10 et une largeur de 20, son angle inférieur droit est placé aux coordonnées 9,19, dans la mesure où l'unité de mesure de la largeur et de la hauteur commencent à 0,0.

Les propriétés `x`, `y`, `width` et `height` de la classe Rectangle sont indépendantes les unes des autres. Le fait de modifier l'une de ces dernières n'a aucun effet sur les autres. Cependant, les propriétés de `right` (droite) et du `bottom` (bas) sont liées de façon intégrale à ces quatre propriétés. Ainsi, si vous changez la valeur de `right` (droite), vous modifiez également la valeur de `width` (largeur) ; et si vous modifiez la valeur de `bottom` (bas), vous modifiez celle du `height` (haut), etc. Et la propriété de `left` (gauche) ou *x* doit être établie avant de définir la propriété `width` (largeur) ou `right` (droite).

Les objets rectangles permettent de prendre en charge les filtres de classe BitmapData. ls sont également utilisés dans la propriété `MovieClip.scrollRect` pour permettre de recadrer et faire défiler une occurrence `MovieClip` avec des valeurs spécifiques de décalage de la largeur, de la hauteur et du défilement.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

## Voir également

`scrollRect (propriété MovieClip.scrollRect)`

## Résumé des propriétés

| Modificateurs | Propriété | Description |
|---|---|---|
| | `bottom:Number` | Somme des propriétés `y` et de `height`. |
| | `bottomRight:Point` | Emplacement du coin inférieur droit de l'objet Rectangle déterminé par les valeurs des propriétés `x` et `y`. |
| | `height:Number` | Hauteur du rectangle en pixels. |
| | `left:Number` | Coordonnée *x* du coin supérieur gauche du rectangle. |
| | `right:Number` | Somme des propriétés `x` et de `width`. |
| | `size:Point` | Taille de l'objet Rectangle, exprimée en tant qu'objet Point avec les valeurs des propriétés `width` (largeur) et `height` (hauteur). |
| | `top:Number` | La coordonnée *y* du coin supérieur gauche du rectangle. |
| | `topLeft:Point` | L'emplacement du coin supérieur gauche de l'objet Rectangle déterminé par les valeurs *x* et *y* du point. |
| | `width:Number` | La largeur de l'objet rectangle en pixels. |
| | `x:Number` | La coordonnée *x* du coin supérieur gauche du rectangle. |
| | `y:Number` | La coordonnée *y* du coin supérieur gauche du rectangle. |

*Propriétés héritées de la classe Object*

`constructeur (propriété Object.constructor), __proto__ (Object.__proto__ property), prototype (Object.prototype, propriété), __resolve (Object.__resolve, propriété)`

## Résumé des constructeurs

| Signature | Description |
|---|---|
| `Rectangle(x:Number, y:Number, width:Number, height:Number)` | Crée un nouvel objet Rectangle dont le coin supérieur gauche est spécifié par les paramètres `x` et `y`. |

Résumé de la méthode

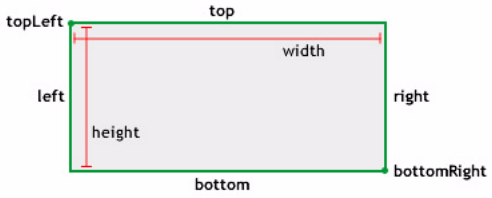
| Modificateurs | Signature | Description |
|---|---|---|
| | `clone() : Rectangle` | Renvoie un nouvel objet Rectangle avec les mêmes valeurs que l'objet Rectangle d'origine pour les propriétés `x`, `y`, `width` (largeur) et `height` (hauteur). |
| | `contains(x:Number, y:Number) : Boolean` | Détermine si le point spécifié figure dans la zone rectangulaire définie par cet objet Rectangle. |
| | `containsPoint(pt:Point) : Boolean` | Détermine si le point spécifié figure dans la zone rectangulaire définie par cet objet Rectangle. |
| | `containsRectangle(rect:Rectangle) : Boolean` | Détermine si l'objet Rectangle spécifié par le paramètre `rect` figure dans cet objet Rectangle. |
| | `equals(toCompare:Object) : Boolean` | Détermine si l'objet spécifié dans le paramètre `toCompare` est égal à cet objet Rectangle. |
| | `inflate(dx:Number, dy:Number) : Void` | Agrandit la taille de l'objet Rectangle en fonction des montants spécifiés. |
| | `inflatePoint(pt:Point) : Void` | Agrandit la taille de l'objet Rectangle. |
| | `intersection(toIntersect:Rectangle) : Rectangle` | Si l'objet Rectangle spécifié dans les paramètres `toIntersect` forme une intersection avec cet objet Rectangle, la méthode `intersection()` renvoie la zone d'intersection en tant qu'objet Rectangle. |
| | `intersects(toIntersect:Rectangle) : Boolean` | Détermine si l'objet spécifié par le paramètre `toIntersect` forme une intersection avec cet objet Rectangle. |
| | `isEmpty() : Boolean` | Détermine si cet objet Rectangle est vide. |
| | `offset(dx:Number, dy:Number) : Void` | Règle la position de l'objet Rectangle, identifié par son coin supérieur gauche, en fonction des montants spécifiés. |
| | `offsetPoint(pt:Point) : Void` | Règle l'emplacement de l'objet Rectangle en utilisant un objet Point en tant que paramètre. |
| | `setEmpty() : Void` | Définit toutes les propriétés de l'objet Rectangle sur 0. |
| | `toString() : String` | Crée et renvoie une chaîne qui répertorie les positions horizontale et verticale ainsi que la largeur et la hauteur de l'objet Rectangle. |
| | `union(toUnion:Rectangle) : Rectangle` | Additionne deux rectangles pour créer un nouvel objet Rectangle en remplissant l'essentiel de l'espace horizontal et vertical qui sépare les deux rectangles. |

addProperty (méthode Object.addProperty), hasOwnProperty (méthode Object.hasOwnProperty), isPropertyEnumerable (méthode Object.isPropertyEnumerable), isPrototypeOf (méthode Object.isPrototypeOf), registerClass (méthode Object.registerClass), toString (méthode Object.toString), unwatch (méthode Object.unwatch), valueOf (méthode Object.valueOf), watch (méthode Object.watch)

# bottom (propriété Rectangle.bottom)

```
public bottom : Number
```

La somme des propriétés `y` et de `height`.




**Disponibilité :** ActionScript 1.0 ; Flash Player 8

## Exemple

L'exemple suivant crée un objet Rectangle et modifie la valeur de sa propriété `bottom` (bas), qui passe de 15 à 30. Notez que la valeur `rect.height` (hauteur) a elle aussi été modifiée, passant de 10 à 25.

```
import flash.geom.Rectangle;

var rect:Rectangle = new Rectangle(5, 5, 10, 10);
trace(rect.height); // 10
trace(rect.bottom); // 15

rect.bottom = 30;
trace(rect.height); // 25
trace(rect.bottom); // 30
```

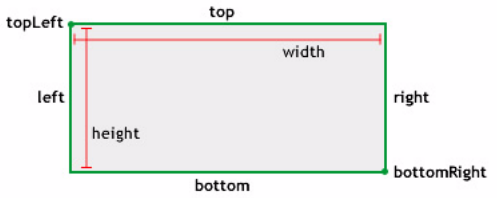
## Voir également

y (propriété Rectangle.y), height (propriété Rectangle.height)

# bottomRight (propriété Rectangle.bottomRight)

```
public bottomRight : Point
```

L'emplacement du coin inférieur droit de l'objet Rectangle déterminé par les valeurs des propriétés `x` et `y`.




**Disponibilité :** ActionScript 1.0 ; Flash Player 8

## Exemple

L'exemple suivant définit la propriété `bottomRight` (en bas à droite) de l'objet Rectangle en reprenant les valeurs de l'objet Point. Notez que `rect.width` et `rect.height` ont été modifiées.

```
import flash.geom.Rectangle;
import flash.geom.Point;

var rect:Rectangle = new Rectangle(1, 2, 4, 8);
trace(rect.bottom); // 10
trace(rect.right); // 5
trace(rect.height); // 8
trace(rect.width); // 4

var myBottomRight:Point = new Point(16, 32);
rect.bottomRight = myBottomRight;
trace(rect.bottom); // 32
trace(rect.right); // 16
trace(rect.height); // 30
trace(rect.width); // 15
```

## Voir également

`Point (flash.geom.Point)`

# clone (méthode Rectangle.clone)

```
public clone() : Rectangle
```

Renvoie un nouvel objet Rectangle avec les mêmes valeurs que l'objet Rectangle d'origine pour les propriétés `x`, `y`, `width` (largeur) et `height` (hauteur).

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

## Valeur renvoyée

`flash.geom.Rectangle` - Nouvel objet Rectangle avec les mêmes valeurs que l'objet Rectangle d'origine pour les propriétés `x`, `y`, `width` (largeur) et `height` (hauteur).

## Exemple

L'exemple suivant crée trois objets Rectangle et les compare. `rect_1` est créé à l'aide du constructeur de rectangle. `rect_2` est créé en lui définissant des valeurs égales à celles du `rect_1`. Enfin, `clonedRect` est créé par clonage du `rect_1`. Remarque : Alors que le `rect_2` est évalué comme étant égal au `rect_1`, ce n'est pas le cas pour le `clonedRect`, même si ce dernier contient les même valeurs que le `rect_1`.

```
import flash.geom.Rectangle;

var rect_1:Rectangle = new Rectangle(1, 2, 4, 8);
var rect_2:Rectangle = rect_1;
var clonedRect:Rectangle = rect_1.clone();

trace(rect_1 == rect_2); // true
trace(rect_1 == clonedFilter); // false

for(var i in rect_1) {
  trace(">> " + i + ": " + rect_1[i]);
  >> toString: [type Function]
  >> equals: [type Function]
  >> union: [type Function]
  >> intersects: [type Function]
  >> intersection: [type Function]
  >> containsRectangle: [type Function]
  >> containsPoint: [type Function]
  >> contains: [type Function]
  >> offsetPoint: [type Function]
  >> offset: [type Function]
  >> inflatePoint: [type Function]
  >> inflate: [type Function]
  >> size: (x=4, y=8)
  >> bottomRight: (x=5, y=10)
  >> topLeft: (x=1, y=2)
  >> bottom: 10
  >> top: 2
  >> right: 5
  >> left: 1
  >> isEmpty: [type Function]
  >> setEmpty: [type Function]
  >> clone: [type Function]
  >> height: 8
```

```
  >> width: 4
  >> y: 2
  >> x: 1
}

for(var i in clonedRect) {
  trace(">> " + i + ": " + clonedRect[i]);
  >> toString: [type Function]
  >> equals: [type Function]
  >> union: [type Function]
  >> intersects: [type Function]
  >> intersection: [type Function]
  >> containsRectangle: [type Function]
  >> containsPoint: [type Function]
  >> contains: [type Function]
  >> offsetPoint: [type Function]
  >> offset: [type Function]
  >> inflatePoint: [type Function]
  >> inflate: [type Function]
  >> size: (x=4, y=8)
  >> bottomRight: (x=5, y=10)
  >> topLeft: (x=1, y=2)
  >> bottom: 10
  >> top: 2
  >> right: 5
  >> left: 1
  >> isEmpty: [type Function]
  >> setEmpty: [type Function]
  >> clone: [type Function]
  >> height: 8
  >> width: 4
  >> y: 2
  >> x: 1
}
```

Pour mieux illustrer les relations existant entre `rect_1`, `rect_2` et `clonedRect`, l'exemple ci-dessous modifie la propriété `x` du `rect_1`. La modification de `x` montre que la méthode `clone()` crée une nouvelle occurrence, basée sur les valeurs du `rect_1`, au lieu de leur faire référence.

```
import flash.geom.Rectangle;

var rect_1:Rectangle = new Rectangle(1, 2, 4, 8);
var rect_2:Rectangle = rect_1;
var clonedRect:Rectangle = rect_1.clone();

trace(rect_1.x); // 1
trace(rect_2.x); // 1
trace(clonedRect.x); // 1
```

```
rect_1.x = 10;

trace(rect_1.x); // 10
trace(rect_2.x); // 10
trace(clonedRect.x); // 1
```

### Voir également

`x (propriété Rectangle.x)`, `y (propriété Rectangle.y)`, `width (propriété Rectangle.width)`, `height (propriété Rectangle.height)`

## contains (méthode Rectangle.contains)

```
public contains(x:Number, y:Number) : Boolean
```

Détermine si le point spécifié figure dans la zone rectangulaire définie par cet objet Rectangle.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

### Paramètres

*x*`:Number` - Valeur *x* (position horizontale) du point.

*y*`:Number` - Valeur *y* (position verticale) du point.

### Valeur renvoyée

`Boolean` - Si le point spécifié figure dans l'objet Rectangle, renvoie `true` ; renvoie `false` dans tous les autres cas.

### Exemple

L'exemple suivant crée un objet Rectangle et teste si chacune des trois paires de coordonnées sont comprises dans ses limites.

```
import flash.geom.Rectangle;

var rect:Rectangle = new Rectangle(10, 10, 50, 50);
trace(rect.contains(59, 59)); // true
trace(rect.contains(10, 10)); // true
trace(rect.contains(60, 60)); // false
```

### Voir également

`Point (flash.geom.Point)`

# containsPoint (méthode Rectangle.containsPoint)

```
public containsPoint(pt:Point) : Boolean
```

Détermine si le point spécifié figure dans la zone rectangulaire définie par cet objet Rectangle. Cette méthode est similaire à la méthode `Rectangle.contains()`, à ceci près qu'elle prend un objet Point comme paramètre.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

## Paramètres

*pt*`:flash.geom.Point` - Point, représenté par ses valeurs *x*,*y*.

## Valeur renvoyée

`Boolean` - Si le point spécifié figure dans l'objet Rectangle, `true` est renvoyé ; `false` dans tous les autres cas.

## Exemple

L'exemple suivant crée un objet Rectangle et trois objets Point et teste si chacun des points est compris dans les limites du rectangle.

```
import flash.geom.Rectangle;
import flash.geom.Point;

var rect:Rectangle = new Rectangle(10, 10, 50, 50);
trace(rect.containsPoint(new Point(10, 10))); // true
trace(rect.containsPoint(new Point(59, 59))); // true
trace(rect.containsPoint(new Point(60, 60))); // false
```

## Voir également

`contains (méthode Rectangle.contains)`, `Point (flash.geom.Point)`

# containsRectangle (méthode Rectangle.containsRectangle)

```
public containsRectangle(rect:Rectangle) : Boolean
```

Détermine si l'objet Rectangle spécifié par le paramètre `rect` figure dans cet objet Rectangle. On dit qu'un objet Rectangle en contient un autre si ce dernier est entièrement circonscrit dans les limites du premier.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

### Paramètres

*rect*:flash.geom.Rectangle - Objet Rectangle en cours de vérification.

### Valeur renvoyée

Boolean - Si l'objet Rectangle que vous spécifiez est compris dans cet objet Rectangle, true est renvoyé ; false dans tous les autres cas.

### Exemple

L'exemple suivant crée quatre nouveaux objets Rectangles et détermine si le rectangle A contient le rectangle B, C, ou D.

```
import flash.geom.Rectangle;

var rectA:Rectangle = new Rectangle(10, 10, 50, 50);
var rectB:Rectangle = new Rectangle(10, 10, 50, 50);
var rectC:Rectangle = new Rectangle(10, 10, 51, 51);
var rectD:Rectangle = new Rectangle(15, 15, 45, 45);

trace(rectA.containsRectangle(rectB)); // true
trace(rectA.containsRectangle(rectC)); // false
trace(rectA.containsRectangle(rectD)); // true
```

## equals (méthode Rectangle.equals)

```
public equals(toCompare:Object) : Boolean
```

Détermine si l'objet spécifié dans le paramètre toCompare est égal à cet objet Rectangle. Cette méthode compare les propriétés x, y, width et height d'un objet aux mêmes propriétés de cet objet Rectangle.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

### Paramètres

*toCompare*:Object - Rectangle que vous souhaitez comparer à cet objet Rectangle.

### Valeur renvoyée

Boolean - Si l'objet a exactement les mêmes valeurs que cet objet Rectangle en ce qui concerne les propriétés x, y, width et height, true est renvoyé ; false est renvoyé dans tous les autres cas.

### Exemple

Dans l'exemple suivant, `rect_1` et `rect_2` sont égaux, mais `rect_3` n'est pas égal aux deux autres objets parce que ses propriétés `x`, `y`, `width`, et `height` ne sont pas égales à celles de `rect_1` et de `rect_2`.

```
import flash.geom.Rectangle;

var rect_1:Rectangle = new Rectangle(0, 0, 50, 100);
var rect_2:Rectangle = new Rectangle(0, 0, 50, 100);
var rect_3:Rectangle = new Rectangle(10, 10, 60, 110);

trace(rect_1.equals(rect_2)); // true;
trace(rect_1.equals(rect_3)); // false;
```

Même si la signature de la méthode prévoit uniquement un objet abstrait, seules d'autres occurrences de Rectangle sont traitées en tant qu'égales.

```
import flash.geom.Rectangle;

var rect_1:Rectangle = new Rectangle(0, 0, 50, 100);
var nonRect:Object = new Object();
nonRect.x = 0;
nonRect.y = 0;
nonRect.width = 50;
nonRect.height = 100;
trace(rect_1.equals(nonRect));
```

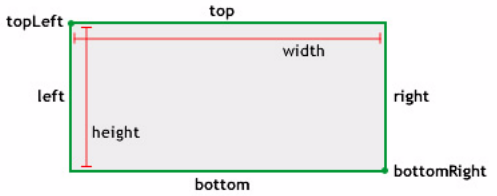
### Voir également

`x (propriété Rectangle.x)`, `y (propriété Rectangle.y)`, `width (propriété Rectangle.width)`, `height (propriété Rectangle.height)`

## height (propriété Rectangle.height)

```
public height : Number
```

La hauteur du rectangle en pixels. La modification de la valeur `height` d'un objet Rectangle n'a pas d'effet sur les propriétés `x`, `y` et `width` (largeur).




**Disponibilité :** ActionScript 1.0 ; Flash Player 8

### Exemple

L'exemple suivant crée un objet Rectangle et modifie sa propriété `height` (hauteur) en la faisant passer de 10 à 20. La valeur de `rect.bottom` est également modifiée.

```
import flash.geom.Rectangle;

var rect:Rectangle = new Rectangle(5, 5, 10, 10);
trace(rect.height); // 10
trace(rect.bottom); // 15

rect.height = 20;
trace(rect.height); // 20
trace(rect.bottom); // 25
```

### Voir également

`x (propriété Rectangle.x)`, `y (propriété Rectangle.y)`, `width (propriété Rectangle.width)`

## inflate (méthode Rectangle.inflate)

```
public inflate(dx:Number, dy:Number) : Void
```

Agrandit la taille de l'objet Rectangle en fonction des montants spécifiés. Le point central de l'objet Rectangle reste inchangé tandis que sa taille augmente de `dx` sur la gauche et la droite et de `dy` vers le haut et bas.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

### Paramètres

*`dx`*`:Number` - Valeur à ajouter sur la gauche et la droite de l'objet Rectangle. On a recours à l'équation suivante pour calculer la nouvelle largeur et la nouvelle position du rectangle :

```
x -= dx;
width += 2 * dx;
```

*`dy`*`:Number` - Valeur à ajouter en haut et en bas de l'objet Rectangle. On a recours à l'équation suivante pour calculer la nouvelle hauteur et la nouvelle position du rectangle.

```
y -= dy;
height += 2 * dy;
```

### Exemple

L'exemple suivant crée un objet Rectangle et augmente la valeur de sa propriété `width` de 16 * 2 (32) et de sa propriété `height` de 32 * 2 (64)

```
import flash.geom.Rectangle;
```

```
var rect:Rectangle = new Rectangle(1, 2, 4, 8);
trace(rect.toString()); // (x=1, y=2, w=4, h=8)

rect.inflate(16, 32);
trace(rect.toString()); // (x=-15, y=-30, w=36, h=72)
```

### Voir également

x (propriété Rectangle.x), y (propriété Rectangle.y)

## inflatePoint (méthode Rectangle.inflatePoint)

```
public inflatePoint(pt:Point) : Void
```

Agrandit la taille de l'objet Rectangle. Cette méthode est similaire à la méthode `Rectangle.inflate()`, à ceci près qu'elle prend un objet Point comme paramètre.

On obtient le même résultat avec les deux exemples de code suivants :

```
rect1 = new flash.geom.Rectangle(0,0,2,5);
rect1.inflate(2,2)
rect1 = new flash.geom.Rectangle(0,0,2,5);
pt1 = new flash.geom.Point(2,2);
rect1.inflatePoint(pt1)
```

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

### Paramètres

*pt*:flash.geom.Point - Agrandit le rectangle en fonction des coordonnées *x* et *y* du point.

### Exemple

L'exemple suivant crée un objet Rectangle et l'agrandit en fonction des montants *x* (horizontal) et *y* (vertical) figurant dans un point.

```
import flash.geom.Rectangle;
import flash.geom.Point;

var rect:Rectangle = new Rectangle(0, 0, 2, 5);
trace(rect.toString()); // (x=0, y=0, w=2, h=5

var myPoint:Point = new Point(2, 2);
rect.inflatePoint(myPoint);
trace(rect.toString()); // (x=-2, y=-2, w=6, h=9)
```

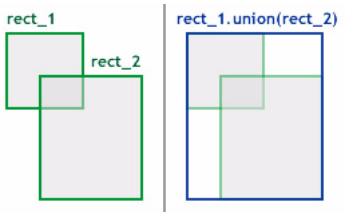
### Voir également

Point (flash.geom.Point)

# intersection (méthode Rectangle.intersection)

```
public intersection(toIntersect:Rectangle) : Rectangle
```

Si l'objet Rectangle spécifié dans les paramètres `toIntersect` forme une intersection avec cet objet Rectangle, la méthode `intersection()` renvoie la zone d'intersection en tant qu'objet Rectangle. Si les rectangles ne se recoupent pas, cette méthode renvoie un objet Rectangle vide dont les propriétés sont définies sur 0.




**Disponibilité :** ActionScript 1.0 ; Flash Player 8

## Paramètres

*`toIntersect`*`:flash.geom.Rectangle` - Objet Rectangle à prendre comme comparaison pour voir s'il recoupe cet objet Rectangle.

## Valeur renvoyée

`flash.geom.Rectangle` - Objet Rectangle qui correspond à la zone d'intersection. Si les rectangles ne se recoupent pas, cette méthode renvoie un objet Rectangle vide dont les propriétés `x`, `y`, `width` et `height` sont définies sur 0.

## Exemple

L'exemple suivant détermine la zone d'intersection de `rect_1` et `rect_2`.

```
import flash.geom.Rectangle;

var rect_1:Rectangle = new Rectangle(0, 0, 50, 50);
var rect_2:Rectangle = new Rectangle(25, 25, 100, 100);
var intersectingArea:Rectangle = rect_1.intersection(rect_2);
trace(intersectingArea.toString()); // (x=25, y=25, w=25, h=25)
```

# intersects (méthode Rectangle.intersects)

```
public intersects(toIntersect:Rectangle) : Boolean
```

Détermine si l'objet spécifié par le paramètre `toIntersect` forme une intersection avec cet objet Rectangle. Cette méthode vérifie les propriétés `x`, `y`, `width` et `height` de l'objet Rectangle spécifié pour déterminer s'il recoupe cet objet Rectangle.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

### Paramètres

*toIntersect*:flash.geom.Rectangle - Objet Rectangle à comparer à cet autre objet Rectangle.

### Valeur renvoyée

Boolean - Si l'objet spécifié recoupe cet objet Rectangle, true est renvoyé ; false dans tous les autres cas.

### Exemple

L'exemple suivant détermine si rectA recoupe rectB ou rectC.

```
import flash.geom.Rectangle;
var rectA:Rectangle = new Rectangle(10, 10, 50, 50);
var rectB:Rectangle = new Rectangle(59, 59, 50, 50);
var rectC:Rectangle = new Rectangle(60, 60, 50, 50);
var rectAIntersectsB:Boolean = rectA.intersects(rectB);
var rectAIntersectsC:Boolean = rectA.intersects(rectC);
trace(rectAIntersectsB); // true
trace(rectAIntersectsC); // false

var firstPixel:Rectangle = new Rectangle(0, 0, 1, 1);
var adjacentPixel:Rectangle = new Rectangle(1, 1, 1, 1);
var pixelsIntersect:Boolean = firstPixel.intersects(adjacentPixel);
trace(pixelsIntersect); // false
```

### Voir également

x (propriété Rectangle.x), y (propriété Rectangle.y), width (propriété Rectangle.width), height (propriété Rectangle.height)

## isEmpty (méthode Rectangle.isEmpty)

```
public isEmpty() : Boolean
```

Détermine si cet objet Rectangle est vide.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

### Valeur renvoyée

Boolean - Si la largeur de l'objet Rectangle ou sa hauteur est inférieure ou égale à 0, true est renvoyé ; false dans tous les autres cas.

### Exemple

L'exemple suivant crée un objet Rectangle vide et vérifie qu'il est vide.

```
import flash.geom.*;
var rect:Rectangle = new Rectangle(1, 2, 0, 0);
trace(rect.toString()); // (x=1, y=2, w=0, h=0)
trace(rect.isEmpty()); // true
```

L'exemple suivant crée un Rectangle non vide puis le vide.

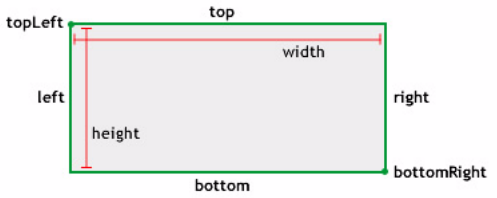
```
import flash.geom.Rectangle;

var rect:Rectangle = new Rectangle(1, 2, 4, 8);
trace(rect.isEmpty()); // false
rect.width = 0;
trace(rect.isEmpty()); // true
rect.width = 4;
trace(rect.isEmpty()); // false
rect.height = 0;
trace(rect.isEmpty()); // true
```

## left (propriété Rectangle.left)

```
public left : Number
```

La coordonnée *x* du coin supérieur gauche du rectangle. La modification de la valeur *x* d'un objet Rectangle n'a pas d'effet sur les propriétés `y`, `width` et `height`.

La propriété `left` est égale à la propriété `x`.




**Disponibilité :** ActionScript 1.0 ; Flash Player 8

### Exemple

L'exemple suivant modifie la propriété `left`, la faisant passer de 0 à 10. Notez que `rect.x` est également modifiée.

```
import flash.geom.Rectangle;

var rect:Rectangle = new Rectangle();
trace(rect.left); // 0
trace(rect.x); // 0
```

```
rect.left = 10;
trace(rect.left); // 10
trace(rect.x); // 10
```

### Voir également

`x (propriété Rectangle.x)`, `y (propriété Rectangle.y)`, `width (propriété Rectangle.width)`, `height (propriété Rectangle.height)`

## offset (méthode Rectangle.offset)

```
public offset(dx:Number, dy:Number) : Void
```

Règle la position de l'objet Rectangle, identifié par son coin supérieur gauche, en fonction des montants spécifiés.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

### Paramètres

*dx*`:Number` - Déplace en fonction de ce montant la valeur *x* de l'objet Rectangle.

*dy*`:Number` - Déplace en fonction de ce montant la valeur *y* de l'objet Rectangle.

### Exemple

L'exemple suivant crée un objet Rectangle et décale ses valeurs *x* et *y* de 5 et 10 respectivement.

```
import flash.geom.Rectangle;

var rect:Rectangle = new Rectangle(1, 2, 4, 8);
trace(rect.toString()); // (x=1, y=2, w=4, h=8)

rect.offset(16, 32);
trace(rect.toString()); // (x=17, y=34, w=4, h=8)
```

## offsetPoint (méthode Rectangle.offsetPoint)

```
public offsetPoint(pt:Point) : Void
```

Règle l'emplacement de l'objet Rectangle en utilisant un objet Point en tant que paramètre. Cette méthode est similaire à la méthode `Rectangle.offset()`, à ceci près qu'elle prend un objet Point comme paramètre.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

### Paramètres

*pt*`:flash.geom.Point` - Objet Point à utiliser pour décaler cet objet Rectangle.

### Exemple

L'exemple suivant décale un Rectangle en reprenant les valeurs figurant dans un point.

```
import flash.geom.Rectangle;
import flash.geom.Point;

var rect:Rectangle = new Rectangle(1, 2, 4, 8);
trace(rect.toString()); // (x=1, y=2, w=4, h=8)

var myPoint:Point = new Point(16, 32);
rect.offsetPoint(myPoint);
trace(rect.toString()); // (x=17, y=34, w=4, h=8)
```

### Voir également

`Point (flash.geom.Point)`

## Rectangle, constructeur

```
public Rectangle(x:Number, y:Number, width:Number, height:Number)
```

Crée un nouvel objet Rectangle dont le coin supérieur gauche est spécifié par les paramètres `x` et `y`. Si vous appelez cette fonction constructeur sans paramètres, un rectangle est créé, dont les propriétés `x`, `y`, `width` et `height` sont définies sur 0.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

### Paramètres

*`x`*`:Number` - Coordonnée *x* du coin supérieur gauche du rectangle.

*`y`*`:Number` - Coordonnée *y* du coin supérieur gauche du rectangle.

*`width`*`:Number` - Largeur du rectangle en pixels.

*`height`*`:Number` - Hauteur du rectangle en pixels.

### Exemple

L'exemple suivant crée un objet Rectangle avec les paramètres spécifiés.

```
import flash.geom.Rectangle;

var rect:Rectangle = new Rectangle(5, 10, 50, 100);
trace(rect.toString()); // (x=5, y=10, w=50, h=100)
```

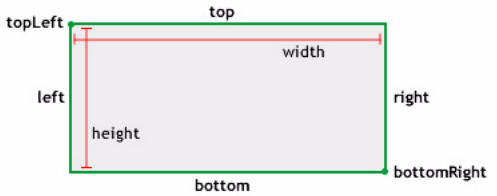
### Voir également

`x (propriété Rectangle.x)`, `y (propriété Rectangle.y)`, `width (propriété Rectangle.width)`, `height (propriété Rectangle.height)`

# right (propriété Rectangle.right)

```
public right : Number
```

La somme des propriétés `x` et de `width`.




**Disponibilité :** ActionScript 1.0 ; Flash Player 8

## Exemple

L'exemple suivant crée un objet Rectangle et modifie sa propriété `right` (hauteur) en la faisant passer de 15 à 30. La valeur de `rect.width` est également modifiée.

```
import flash.geom.Rectangle;

var rect:Rectangle = new Rectangle(5, 5, 10, 10);
trace(rect.width); // 10
trace(rect.right); // 15

rect.right = 30;
trace(rect.width); // 25
trace(rect.right); // 30
```

## Voir également

`x (propriété Rectangle.x)`, `width (propriété Rectangle.width)`

# setEmpty (méthode Rectangle.setEmpty)

```
public setEmpty() : Void
```

Définit toutes les propriétés de l'objet Rectangle sur 0. Un objet Rectangle est vide si sa largeur ou sa hauteur est inférieure ou égale à 0.

Cette méthode règle les valeurs des propriétés `x`, `y`, `width` et `height` sur 0.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

## Exemple

L'exemple suivant crée un objet Rectangle non vide puis le vide.

```
import flash.geom.Rectangle;
```

```
var rect:Rectangle = new Rectangle(5, 10, 50, 100);
trace(rect.isEmpty()); // false
rect.setEmpty();
trace(rect.isEmpty()); // true
```

### Voir également

x (propriété Rectangle.x), y (propriété Rectangle.y), width (propriété Rectangle.width), height (propriété Rectangle.height)

## size (propriété Rectangle.size)

```
public size : Point
```

La taille de l'objet Rectangle, exprimée en tant qu'objet Point avec les valeurs des propriétés `width` (largeur) et `height` (hauteur).

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

### Exemple

L'exemple suivant crée un objet Rectangle, lit sa taille (`size`), change sa taille (`size`) et définit les nouvelles valeurs sur l'objet Rectangle. Il est important de ne pas oublier que l'objet `Point` utilisé par la propriété `size` reprend les valeurs *x* et *y* pour représenter les propriétés `width` et `height` de l'objet Rectangle.

```
import flash.geom.Rectangle;
import flash.geom.Point;

var rect:Rectangle = new Rectangle(1, 2, 4, 8);
var size:Point = rect.size;
trace(size.x); // 4;
trace(size.y); // 8;

size.x = 16;
size.y = 32;
rect.size = size;
trace(rect.x); // 1
trace(rect.y); // 2
trace(rect.width); // 16
trace(rect.height); // 32
```

### Voir également

Point (flash.geom.Point)

# top (propriété Rectangle.top)

```
public top : Number
```

La coordonnée *y* du coin supérieur gauche du rectangle. La modification de la valeur de la propriété `top` d'un objet Rectangle n'a pas d'effet sur les propriétés `x`, `width` et `height`.

La valeur de la propriété `top` est égale à la valeur de la propriété de `y`.


**Disponibilité :** ActionScript 1.0 ; Flash Player 8

## Exemple

L'exemple suivant modifie la valeur de la propriété `top`, la faisant passer de 0 à 10. Notez que `rect.y` est également modifiée.

```
import flash.geom.Rectangle;

var rect:Rectangle = new Rectangle();
trace(rect.top); // 0
trace(rect.y); // 0

rect.top = 10;
trace(rect.top); // 10
trace(rect.y); // 10
```

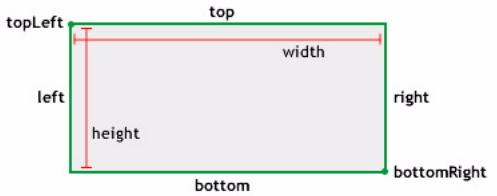
## Voir également

`x (propriété Rectangle.x)`, `y (propriété Rectangle.y)`, `width (propriété Rectangle.width)`, `height (propriété Rectangle.height)`

# topLeft (propriété Rectangle.topLeft)

```
public topLeft : Point
```

L'emplacement du coin supérieur gauche de l'objet Rectangle déterminé par les valeurs *x* et *y* du point.




**Disponibilité :** ActionScript 1.0 ; Flash Player 8

### Exemple

L'exemple suivant définit la propriété `topLeft` (en bas à droite) de l'objet Rectangle en reprenant les valeurs figurant dans un objet Point. Notez que `rect.x` et `rect.y` ont été modifiées.

```
import flash.geom.Rectangle;
import flash.geom.Point;

var rect:Rectangle = new Rectangle();
trace(rect.left); // 0
trace(rect.top); // 0
trace(rect.x); // 0
trace(rect.y); // 0

var myTopLeft:Point = new Point(5, 15);
rect.topLeft = myTopLeft;
trace(rect.left); // 5
trace(rect.top); // 15
trace(rect.x); // 5
trace(rect.y); // 15
```

### Voir également

`Point (flash.geom.Point)`, `x (propriété Rectangle.x)`, `y (propriété Rectangle.y)`

## toString (méthode Rectangle.toString)

```
public toString() : String
```

Crée et renvoie une chaîne qui répertorie les positions horizontale et verticale ainsi que la largeur et la hauteur de l'objet Rectangle.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

### Valeur renvoyée

`String` - Chaîne qui répertorie la valeur des différentes propriétés de l'objet Rectangle : `x`, `y`, `width` et `height`.

### Exemple

L'exemple suivant concatène une représentation par string de `rect_1` avec un texte de débogage utile.

```
import flash.geom.Rectangle;

var rect_1:Rectangle = new Rectangle(0, 0, 50, 100);
```

```
trace("Rectangle 1 : " + rect_1.toString()); // Rectangle 1 : (x=0, y=0,
  w=50, h=100)
```

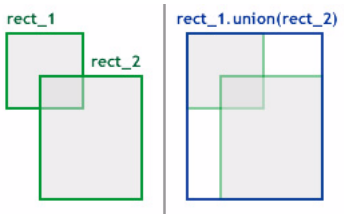
### Voir également

x (propriété Rectangle.x), y (propriété Rectangle.y), width (propriété Rectangle.width), height (propriété Rectangle.height)

## union (méthode Rectangle.union)

```
public union(toUnion:Rectangle) : Rectangle
```

Additionne deux rectangles pour créer un nouvel objet Rectangle en remplissant l'essentiel de l'espace horizontal et vertical qui sépare les deux rectangles.




**Disponibilité :** ActionScript 1.0 ; Flash Player 8

### Paramètres

*toUnion*:flash.geom.Rectangle - Objet Rectangle à ajouter à cet objet Rectangle.

### Valeur renvoyée

flash.geom.Rectangle - Nouvel objet Rectangle qui correspond à l'union des deux rectangles.

### Exemple

L'exemple suivant crée un objet Rectangle à partir de l'union de deux autres.

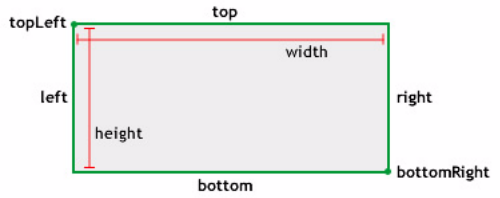
Par exemple, soit un rectangle ayant les propriétés suivantes, x=20, y=50, width=60 et height=30 (20, 50, 60, 30), et un deuxième rectangle avec les propriétés (150, 130, 50, 30). L'union de ces deux rectangles devient un rectangle qui assimile les deux rectangles avec les propriétés (20, 50, 180, 110).

```
import flash.geom.Rectangle;

var rect_1:Rectangle = new Rectangle(20, 50, 60, 30);
var rect_2:Rectangle = new Rectangle(150, 130, 50, 30);
var combined:Rectangle = rect_1.union(rect_2);
trace(combined.toString()); // (x=20, y=50, w=180, h=110)
```

# width (propriété Rectangle.width)

```
public width : Number
```

La largeur de l'objet rectangle en pixels. La modification de la valeur de la propriété `width` d'un objet Rectangle n'a pas d'effet sur les propriétés `x`, `y` et `height`.




**Disponibilité :** ActionScript 1.0 ; Flash Player 8

## Exemple

L'exemple suivant crée un objet Rectangle et modifie sa propriété `width` (largeur) en la faisant passer de 10 à 20. La valeur de `rect.right` est également modifiée.

```
import flash.geom.Rectangle;

var rect:Rectangle = new Rectangle(5, 5, 10, 10);
trace(rect.width); // 10
trace(rect.right); // 15

rect.width = 20;
trace(rect.width); // 20
trace(rect.right); // 25
```

## Voir également

`x (propriété Rectangle.x)`, `y (propriété Rectangle.y)`, `height (propriété Rectangle.height)`

# x (propriété Rectangle.x)

```
public x : Number
```

La coordonnée *x* du coin supérieur gauche du rectangle. La modification de la valeur de la propriété *x* d'un objet Rectangle n'a pas d'effet sur les propriétés `y`, `width` et `height`.

La propriété `x` est égale à la propriété `left`.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

### Exemple

L'exemple suivant crée un Rectangle vide et définit sa propriété `x` sur 10. Notez que la valeur de `rect.left` est également modifiée.

```
import flash.geom.Rectangle;

var rect:Rectangle = new Rectangle();
trace(rect.x); // 0
trace(rect.left); // 0

rect.x = 10;
trace(rect.x); // 10
trace(rect.left); // 10
```

### Voir également

`left (propriété Rectangle.left)`

## y (propriété Rectangle.y)

```
public y : Number
```

La coordonnée *y* du coin supérieur gauche du rectangle. La modification de la valeur de la propriété `y` d'un objet Rectangle n'a pas d'effet sur les propriétés `x`, `width` et `height`.

La propriété `y` est égale à la propriété `top`.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

### Exemple

L'exemple suivant crée un Rectangle vide et définit sa propriété `y` sur 10. Notez que la valeur de `rect.top` est également modifiée.

```
import flash.geom.Rectangle;

var rect:Rectangle = new Rectangle();
trace(rect.y); // 0
trace(rect.top); // 0

rect.y = 10;
trace(rect.y); // 10
trace(rect.top); // 10
```

### Voir également

`x (propriété Rectangle.x)`, `width (propriété Rectangle.width)`, `height (propriété Rectangle.height)`, `top (propriété Rectangle.top)`

# security (System.security)

```
Object
  |
  +-System.security
```

```
public class security
extends Object
```

Le System.security contient des méthodes spécifiant la façon dont les fichiers SWF peuvent communiquer entre eux dans différents domaines.

Pour plus d'informations, consultez les sections suivantes :

- Chapitre 17, « Understanding Security » (Comprendre la sécurité), du manuel *Formation à ActionScript 2.0 dans Flash*
- La présentation technique de sécurité de Flash Player 8 disponible à l'adresse : http://www.macromedia.com/go/fp8_security
- La présentation technique des API relatives à la sécurité de Flash Player 8 disponible à l'adresse : http://www.macromedia.com/go/fp8_security_apis

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

## Résumé des propriétés

| Modificateurs | Propriété | Description |
|---|---|---|
| `static` | `sandboxType:String` [lecture seule] | Indique le type de sanbdox de sécurité dans lequel fonctionne le fichier SWF appelant. |

*Propriétés héritées de la classe Object*

`constructeur (propriété Object.constructor), __proto__ (Object.__proto__ property), prototype (Object.prototype, propriété), __resolve (Object.__resolve, propriété)`

Résumé de la méthode

| Modificateurs | Signature | Description |
|---|---|---|
| static | allowDomain(domain1:String) : Void | Permet aux fichiers SWF et HTML figurant dans les domaines identifiés d'accéder aux objets et aux variables du fichier SWF qui contient l'appel allowDomain(). |
| static | allowInsecureDomain(domain:String) : Void | Permet aux fichiers SWF et HTML appartenant aux domaines identifiés d'accéder aux objets et variables du fichier SWF effectuant l'appel, hébergé à l'aide du protocole HTTPS. |
| static | loadPolicyFile(url:String) : Void | Charge un fichier de régulation interdomaine à partir d'un emplacement spécifié par le paramètre url. |

*Méthodes héritées de la classe Object*

addProperty (méthode Object.addProperty), hasOwnProperty (méthode Object.hasOwnProperty), isPropertyEnumerable (méthode Object.isPropertyEnumerable), isPrototypeOf (méthode Object.isPrototypeOf), registerClass (méthode Object.registerClass), toString (méthode Object.toString), unwatch (méthode Object.unwatch), valueOf (méthode Object.valueOf), watch (méthode Object.watch)

# allowDomain (méthode security.allowDomain)

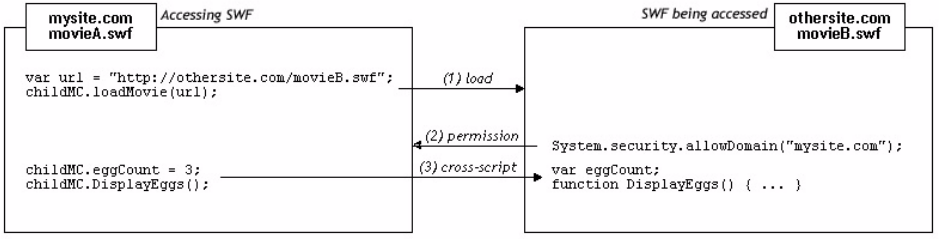
```
public static allowDomain(domain1:String) : Void
```

Permet aux fichiers SWF et HTML figurant dans les domaines identifiés d'accéder aux objets et aux variables du fichier SWF qui contient l'appel `allowDomain()`.

Si deux fichiers SWF sont servis à partir du même domaine - par exemple, http://mysite.com/movieA.swf et http://mysite.com/movieB.swf, movieA.swf peut alors analyser et modifier les variables, les objets, les propriétés, les méthodes, etc. dans movieB.swf, et movieB peut faire la même chose pour movieA.swf. Ceci se dénomme *programmation entre plusieurs animations* ou simplement *programmation croisée*.

Si deux fichiers SWF sont servis à partir de différents domaines - par exemple, http://mysite.com/movieA.swf et http://othersite.com/movieB.swf, par défaut, Flash Player ne permettra pas à movieA.swf de programmer movieB.swf, ou à movieB de programmer movieA.swf. Un fichier SWF autorise des fichiers SWF d'autres domaines à le programmer en appelant `System.security.allowDomain()`. Ceci s'appelle programmation de *scripts interdomaine*. En appelant `System.security.allowDomain("mysite.com")`, movieB.swf autorise movieA.swf à programmer movieB.swf.

En cas de situation de type interdomaine, deux parties sont impliquées et il est important de les identifier. Dans le cadre de cette discussion, le côté procédant à la programmation croisée sera appelé *partie procédant à l'accès* (habituellement le fichier SWF procédant à l'accès), et l'autre côté sera appelé *partie cible* (généralement le fichier SWF cible). Pour poursuivre l'exemple, lorsque movieA.swf programme movieB.swf, movieA.swf est la source, tandis que movieB.swf constitue la cible.




Les autorisations interdomaine établies avec `System.security.allowDomain()` sont asymétriques. Dans l'exemple ci-dessus, movieA.swf peut programmer movieB.swf, mais movieB.swf ne peut pas programmer movieA.swf, car movieA.swf n'a pas appelé `System.security.allowDomain()` pour autoriser othersite.com à programmer movieA.swf. Vous pouvez définir les autorisations symétriques en amenant les deux fichiers SWF à appeler System.security.allowDomain().

En dehors de la protection des fichiers SWF contre les scripts interdomaine provenant d'autres fichiers SWF, Flash Player protège également les fichiers SWF contre ce type de script provenant des fichiers HTML. La programmation HTML vers SWF peut s'effectuer avec des fonctions anciennes du navigateur Flash telles que `SetVariable` ou en appelant des fonctions de rappel établies avec `ExternalInterface.addCallback()`. Lorsque la programmation HTML vers SWF franchit les limites du domaine, le SWF cible doit également appeler `System.security.allowDomain()`, comme s'il avait été appelé par un fichier SWF, faute de quoi l'opération échouera.

La spécification de l'adresse IP en tant que paramètre pour `System.security.allowDomain()` n'autorise pas l'accès de toutes les parties provenant de l'adresse IP spécifiée. Par contre, ceci restreint l'accès aux parties qui ont été chargées en spécifiant explicitement cette adresse IP dans leur URL, plutôt qu'en utilisant un nom de domaine qui corresponde à cette adresse IP.

*Différences spécifiques à la version* Les règles de sécurité interdomaine de Flash Player ont évolué de version en version. Le tableau suivant récapitule les différences.

| **Versions SWF les plus récentes impliquées dans les opérations de programmation croisée.** | **allowDomain() nécessaire ?** | **allowInsecureDomain() nécessaire ?** | **Quel fichier SWF doit appeler allowDomain() ou allowInsecureDomain() ?** | **Qu'est-ce qui peut être spécifié dans allowDomain() ou allowInsecureDomain() ?** |
|---|---|---|---|---|
| 5 ou plus récente | Non | Non | S/O | |
| 6 | Oui, si les superdomaines ne concordent pas | | Le fichier SWF en cours d'accès ou tout fichier SWF appartenant au même superdomaine que le fichier SWF en cours d'accès. | ■ Domaine de type texte (monsite.com)<br>■ Adresse IP (192.168.1.1) |
| 7 | Oui, si les domaines ne concordent pas exactement | Oui, en cas d'accès HTTP vers HTTPS (même si les domaines correspondent exactement) | Le fichier SWF en cours d'accès ou tout fichier SWF appartenant exactement au même domaine que le fichier SWF en cours d'accès. | |
| 8 ou plus récente | | | SWF cible | ■ Domaine de type texte (monsite.com)<br>■ Adresse IP (192.168.1.1)<br>■ Caractère générique (*) |

Les versions qui contrôlent le comportement de Flash Player désignent les *versions des fichiers SWF* (la version publiée d'un fichier SWF), non pas la version de Flash Player. Par exemple, lorsque Flash Player 8 lit un fichier SWF publié pour la version 7, il applique les comportements correspondant à cette version 7. Cette pratique permet de s'assurer que les mises à niveau du lecteur ne changent pas le comportement de `System.security.allowDomain()` dans les fichiers SWF déployés.

La colonne Version du tableau précédent indique la version la plus récente des fichiers SWF lors des opérations de programmation croisée. Le comportement de Flash Player dépend de la version du fichier SWF procédant à l'accès ou du fichier SWF cible, en retenant la version supérieure.

Les paragraphes suivants fournissent de plus amples informations sur les modifications de sécurité de Flash Player impliquant `System.security.allowDomain()`.

*Version 5*. Aucune restriction de programmation interdomaine.

*Version 6*. Des fonctions de sécurité contre la programmation interdomaine ont été introduites. Par défaut, Flash Player empêche la programmation interdomaine, tandis que `System.security.allowDomain()` l'autorise. Pour déterminer si deux fichiers appartiennent au même domaine, Flash Player utilise le superdomaine de chaque fichier, qui correspond au nom d'hôte exact de l'URL du fichier, moins le premier segment, jusqu'à un minimum de deux segments. Par exemple, le superdomaine de www.mysite.com est simplement mysite.com. Ceci autoriserait, par exemple, les fichiers SWF de www.mysite.com et store.mysite.com de se programmer l'un l'autre sans appeler `System.security.allowDomain()`.

*Version 7*. Le filtrage de superdomaine est modifié pour obtenir la correspondance exacte des domaines. Deux fichiers ne peuvent se programmer que si les noms d'hôte figurant dans leurs URL sont identiques ; sinon vous devez effectuer un appel à `System.security.allowDomain()`. Par défaut, les fichiers chargés à partir des URL qui ne sont pas de type HTTPS ne sont plus autorisés à programmer les fichiers chargés à partir des URL HTTPS, même si les fichiers sont chargés à partir d'un domaine rigoureusement identique. Cette restriction permet de protéger les fichiers HTTPS, dans la mesure où les fichiers qui ne sont pas régis par le protocole HTTPS sont vulnérables aux modifications pendant les téléchargements, et si ce type de fichier est manipulé, il risque de corrompre un fichier HTTPS, qui est normalement à l'abri de ce genre de manipulation. `System.security.allowInsecureDomain()` a été ajouté pour permettre aux fichiers SWF HTTPS cible de désactiver cette restriction si nécessaire. Néanmoins, Macromedia déconseille d'utiliser `System.security.allowInsecureDomain()`.

*Version 8*. Deux grandes zones de modification :

- L'appel de `System.security.allowDomain()` autorise désormais uniquement les opérations de programmation croisée où le fichier SWF cible correspond au fichier SWF qui a appelé `System.security.allowDomain()`. En d'autres termes, tout fichier SWF qui appelle désormais `System.security.allowDomain()` n'autorise que l'accès à lui-même. Dans des versions précédentes, l'appel de `System.security.allowDomain()` autorisait les opérations de programmation croisée lorsque le fichier SWF cible appartenait au même domaine que le fichier SWF qui a appelé `System.security.allowDomain()`. L'appel de `System.security.allowDomain()` ouvrait auparavant l'ensemble du domaine du fichier SWF ayant procédé à l'appel.
- Une prise en charge a été ajoutée pour les valeurs des caractères génériques avec `System.security.allowDomain("*")` et `System.security.allowInsecureDomain("*")`. La valeur caractère générique (*) autorise les opérations de programmation croisée quel que soit le fichier procédant à l'accès et quelle que soit l'origine de ce dernier. Le caractère générique sert alors d'autorisation globale. Les autorisations par caractère générique peuvent être utiles de façon générale et elles sont en particulier requises pour activer certains types d'opération conformément aux nouvelles règles de sécurité des fichiers locaux de Flash Player 8. Plus précisément, pour qu'un fichier SWF local avec des autorisations d'accès au réseau puisse programmer un fichier SWF sur Internet, le fichier cible doit appeler `System.security.allowDomain("*")`, ce qui tient compte du fait que l'origine du fichier SWF local est inconnue. (Si le fichier SWF Internet cible est chargé à partir d'une URL HTTPS, le fichier SWF Internet doit alors appeler `System.security.allowInsecureDomain("*")`.)

La situation suivante risque parfois de se présenter : Vous chargez un fichier SWF enfant à partir d'un domaine différent et souhaitez lui permettre de créer un script sur le fichier SWF parent, mais vous ne connaissez pas le domaine final à partir duquel sera issu le fichier SWF enfant. Cela peut se produire, par exemple, lorsque vous utilisez des redirections d'équilibrage de charge ou des serveurs tiers.

Dans ce cas, vous pouvez utiliser la propriété `MovieClip._url` comme paramètre de cette méthode. Par exemple, si vous chargez un fichier SWF dans le clip `my_mc`, vous pouvez appeler `System.security.allowDomain(my_mc._url)`. Si vous procédez ainsi, veuillez patienter jusqu'au début du chargement du fichier SWF dans `my_mc` car la propriété `_url` ne dispose pas de sa valeur correcte et finale qu'à ce moment là. La meilleure façon de déterminer si le chargement d'un fichier SWF enfant a commencé est d'utiliser `MovieClipLoader.onLoadStart`.

La situation opposée peut également se produire ; en effet, vous pouvez créer un fichier SWF enfant sur lequel son fichier parent pourra créer un script, mais qui ignore le domaine de celui-ci. Dans ce cas, appelez `System.security.allowDomain(_parent._url)` à partir du fichier SWF enfant. Il n'est pas nécessaire d'attendre la fin du chargement du fichier SWF parent ; le parent sera déjà chargé lorsque celui de l'enfant commencera.

Si vous procédez à la publication de Flash Player 8, vous pouvez également traiter ces situations en appelant `System.security.allowDomain("*")`. Cependant, il peut parfois s'agir d'un raccourci dangereux, dans la mesure où il autorise tout autre fichier SWF, quel que soit le domaine de ce dernier, à accéder au fichier SWF procédant à l'appel. Il est généralement plus sûr d'utiliser la propriété `_url`.

Pour plus d'informations, consultez les sections suivantes :

- Chapitre 17, « Understanding Security » (Comprendre la sécurité), dans *Learning ActionScript 2.0 dans Flash*
- La présentation technique de sécurité de Flash Player 8 disponible à l'adresse : http://www.macromedia.com/go/fp8_security
- La présentation technique des API relatives à la sécurité de Flash Player 8 disponibl à l'adresse : http://www.macromedia.com/go/fp8_security_apis

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

## Paramètres

*`domain1`*`:String` - Chaîne(s) spécifiant les domaines qui peuvent accéder aux objets et aux variables dans le fichier SWF contenant l'appel `System.Security.allowDomain()`. Les domaines peuvent être formatés de différentes façons :

- `"domain.com"`
- `"http://domain.com"`
- `"http://IPaddress"`
- (Flash Player 8 uniquement) `"*"`
  Vous pouvez transmettre un caractère générique ("*") à `System.security.allowDomain()` pour permettre à l'ensemble des domaines, ce qui inclut les hôtes locaux, d'accéder au fichier SWF procédant à l'appel. Avant d'utiliser le caractère générique, assurez-vous qu'un tel accès général au fichier SWF source est réellement nécessaire. Consultez la description principale de cette méthode pour obtenir plus de détails.

### Exemple

Le fichier SWF, www.macromedia.com/MovieA.swf, contient les lignes suivantes :

```
System.security.allowDomain("www.shockwave.com");
loadMovie("http://www.shockwave.com/MovieB.swf", my_mc);
```

Dans la mesure où MovieA contient l'appel `allowDomain()`, MovieB peut accéder aux objets et aux variables de MovieA. Si MovieA ne contenait pas cet appel, la fonction de sécurité de Flash Player empêcherait MovieB d'accéder aux objets et aux variables de MovieA.

### Voir également

`addCallback (méthode ExternalInterface.addCallback)`, `onLoadComplete (écouteur d'événement MovieClipLoader.onLoadComplete)`, `_parent (propriété MovieClip._parent)`, `_url (propriété MovieClip._url)`, `allowInsecureDomain (méthode security.allowInsecureDomain)`

## allowInsecureDomain (méthode security.allowInsecureDomain)

```
public static allowInsecureDomain(domain:String) : Void
```

Permet aux fichiers SWF et HTML appartenant aux domaines identifiés d'accéder aux objets et variables du fichier SWF effectuant l'appel, hébergé à l'aide du protocole HTTPS. Macromedia déconseille cette méthode ; consultez la section relative à la sécurité, plus bas dans cette section.

Cette méthode fonctionne de la même façon que `System.security.allowDomain()`, mais elle autorise en outre des opérations où la partie qui procède à l'accès est chargée avec un protocole non HTTPS et la partie cible est chargée avec le protocole HTTPS. A partir de la version 7 de Flash Player, les fichiers non HTTPS ne sont pas autorisés à programmer les fichiers HTTPS. La méthode `allowInsecureDomain()` lève cette restriction lorsque le fichier SWF HTTPS cible l'utilise.

Utilisez `allowInsecureDomain()` uniquement pour activer la programmation des fichiers non HTTPS vers les fichiers HTTPS. Utilisez cette méthode pour activer la programmation lorsque le fichier non HTTPS source et le fichier HTTPS cible sont servis à partir du même domaine, par exemple, si un fichier SWF figurant sur http://mysite.com doit programmer https://mysite.com. Ne l'utilisez pas pour activer la programmation entre les fichiers non HTTPS, entre les fichiers HTTPS ou des fichiers HTTPS vers les fichiers non HTTPS. Dans ces situations, recourez plutôt à `allowDomain()`.

*Section relative à la sécurité*: Flash Player fournit `allowInsecureDomain()` pour une plus grande souplesse, bien que Macromedia en déconseille l'utilisation. La transmission d'un fichier avec le protocole HTTPS offre plusieurs protections pour vous et vos utilisateurs. Le fait d'appeler `allowInsecureDomain` affaiblit l'une de ces protections. Le scénario suivant illustre la façon dont la méthode `allowInsecureDomain()`, si elle n'est pas utilisée avec prudence, risque de compromettre la sécurité.

*Remarque* : les informations suivantes constituent uniquement l'un des scénarios possibles et sont conçues pour vous aider à comprendre `allowInsecureDomain()` par l'intermédiaire d'un exemple réaliste de programmation croisée. Cet exemple ne couvre pas tous les problèmes relatifs à l'architecture de sécurité et doit être utilisé uniquement comme référence générale. Le Centre de développement de Macromedia contient des informations détaillées sur Flash Player et la sécurité. Pour plus d'informations, consultez le site http://www.macromedia.com/devnet/security/.

Imaginons que vous deviez créer un site d'e-commerce comprenant deux composants : un catalogue, qui ne doit pas nécessairement être sécurisé, dans la mesure où il contient uniquement des informations publiques, et l'autre composant, un caddie/une caisse, qui doit être sécurisé pour protéger les informations financières et personnelles des utilisateurs. Supposons que vous placiez le catalogue dans http://mysite.com/catalog.swf et le caddie dans https://mysite.com/cart.swf. L'un des éléments du cahier des charges stipule qu'aucun tiers ne doit pouvoir voler les numéros de carte bancaire des utilisateurs en exploitant une faiblesse de l'architecture de sécurité.

Imaginons qu'un intermédiaire malveillant tente d'intervenir entre le serveur et vos utilisateurs pour s'emparer des numéros de carte de crédit que vos utilisateurs pénètrent dans votre application de caddie. L'intermédiaire, peut être un FAI peu scrupuleux, par exemple, ou un administrateur malveillant travaillant dans la même entreprise que certains utilisateurs, ou de façon plus générale, toute personne ayant la possibilité d'afficher ou modifier les paquets réseau transmis sans protection sur Internet, entre vos utilisateurs et vos serveurs. Cette situation n'est pas rare.

Si cart.swf utilise HTTPS pour transmettre les informations bancaires aux serveurs, l'intermédiaire ne peut pas voler directement ces informations en détournant les paquets réseau, dans la mesure où la transmission HTTPS est chiffrée. Cependant, l'attaquant peut utiliser une autre technique : modifier le contenu de l'un de vos fichiers SWF pendant sa remise à l'utilisateur, en remplaçant le fichier SWF par une version modifiée qui détourne les informations relatives à l'utilisateur vers un autre serveur.

Le protocole HTTPS, entre autres, empêche l'application de cette « modification », dans la mesure où non seulement les transmissions HTTPS sont chiffrées mais encore protégées contre les modifications. Si un intermédiaire tente de modifier un paquet, le récepteur détecte la modification et refuse le paquet. Ainsi, l'attaquant ne peut pas modifier cart.swf, dans la mesure où il est transmis par l'intermédiaire du protocole HTTPS.

Supposons maintenant que vous souhaitiez autoriser les boutons dans catalog.swf, servi par le protocole HTTP, pour ajouter des éléments au caddie dans cart.swf, servi par le protocole HTTPS. Pour appliquer cette fonctionnalité, cart.swf appelle `allowInsecureDomain()`, qui permet à catalog.swf de programmer cart.swf. Cette action a une conséquence imprévue : un attaquant potentiel peut modifier catalog.swf lorsqu'il est téléchargé par l'utilisateur, car catalog.swf est transmis avec le protocole HTTP et n'offre aucune protection contre les modifications. Le fichier catalog.swf modifié par l'attaquant peut désormais programmer cart.swf, dans la mesure où cart.swf contient un appel à `allowInsecureDomain()`. Le fichier catalog.swf modifié peut utiliser ActionScript pour accéder aux variables de cart.swf et lire ainsi les informations sur les cartes bancaires et autres données sensibles. Le fichier catalog.swf peut ensuite envoyer ces données au serveur d'un attaquant.

Naturellement, cette implémentation n'est pas souhaitable, mais vous devez autoriser la programmation croisée entre les deux fichiers SWF de votre site. Voici deux façons de changer la conception de ce site virtuel d'e-commerce afin d'éviter `allowInsecureDomain()` :

- Servez tous les fichiers SWF de l'application avec le protocole HTTPS. Il s'agit de la solution la plus simple et la plus fiable. Dans le scénario décrit, vous pouvez servir les fichiers catalog.swf et cart.swf par l'intermédiaire du protocole HTTPS. Vous risquez de consommer un peu plus de bande passante et d'augmenter la charge du processeur du serveur en faisant basculer un fichier tel que catalog.swf du protocole HTTP au protocole HTTPS, ce qui se traduira par une légère augmentation du temps de chargement des applications au niveau de l'utilisateur. Vous devez faire des essais avec des serveurs réels pour déterminer la gravité de ces effets. De manière générale, elle reste cantonnée entre 10 et 20 %, et est parfois totalement absente. Vous pouvez généralement améliorer les résultats avec du matériel et des logiciels d'accélération HTTPS sur vos serveurs. L'un des principaux avantage de l'application du protocole HTTPS aux fichiers SWF qui doivent coopérer est que vous pouvez utiliser une URL HTTPS en tant qu'URL principale dans le navigateur de l'utilisateur sans générer d'avertissements de contenu mixtes à partir du navigateur. En outre, l'icône en forme de cadenas devient visible dans le navigateur, ce qui permet d'offrir aux utilisateurs un indicateur de sécurité reconnu.

- Utilisez la programmation HTTPS vers HTTP, et non pas HTTP vers HTTPS. Dans le scénario proposé, vous pouvez stocker le contenu du caddie de l'utilisateur dans catalog.swf, puis utiliser cart.swf pour gérer le processus de règlement. Lors du règlement, cart.swf peut extraire le contenu du caddie à l'aide des variables ActionScript de catalog.swf. La restriction relative à la programmation HTTP vers HTTPS est asymétrique. Ainsi, si catalog.swf est transmis avec le protocole HTTP, ce fichier ne permet pas de programmer en toute sécurité le fichier HTTPS cart.swf. Par contre, un fichier HTTPS, par ex. cart.swf, peut programmer un fichier HTTP, par ex. catalog.swf. Cette approche est plus délicate que l'approche intégralement HTTPS ; vous ne devez pas faire confiance aux fichiers SWF transmis avec le protocole HTTP, qui n'est pas protégé contre les modifications. Par exemple, lorsque cart.swf extrait la variable ActionScript qui décrit le contenu du caddie, le code ActionScript de cart.swf ne peut pas être certain que la valeur de cette variable est au format attendu. Vous devez vous assurer que le caddie ne contient pas de données non valides qui risquent d'entraîner une action imprévue de cart.swf. Vous devez également accepter le risque qu'un intermédiaire, en modifiant catalog.swf, fournisse des données valides mais inexactes à cart.swf, par exemple en plaçant des éléments dans le caddie de l'utilisateur. La procédure normale de règlement permet d'atténuer ce risque, sans toutefois l'écarter totalement, en affichant le contenu du caddie et le montant total pour approbation par l'utilisateur.

Les navigateurs Web appliquent la séparation des fichiers HTTPS et non HTTPS depuis de nombreuses années et le scénario ci-dessus illustre l'utilité de cette restriction. Flash Player permet de contourner cette restriction de sécurité lorsque c'est strictement nécessaire, mais analysez les conséquences avant d'y procéder.

Pour plus d'informations, consultez les sections suivantes :

- Chapitre 17, « Understanding Security » (Comprendre la sécurité), dans *Learning ActionScript 2.0 dans Flash*
- La présentation technique de sécurité de Flash Player 8 disponible à l'adresse : The Flash Player 8 Secuirity whitepape
- La présentation technique des API relatives à la sécurité de Flash Player 8 disponible à l'adresse : http://www.macromedia.com/go/fp8_security_apis

**Disponibilité :** ActionScript 1.0 ; Flash Player 7

### Paramètres

*domain*:String - Nom de domaine exact, tel que www.myDomainName.com ou store.myDomainName.com. Dans Flash Player 8, vous pouvez transmettre un caractère générique ("*") à System.security.allowInsecureDomain() pour permettre à l'ensemble des domaines, ce qui inclut les hôtes locaux, d'accéder au fichier SWF procédant à l'appel. N'utilisez ce caractère générique que si vous êtes certain de devoir autoriser *tous* les domaines, ce qui inclut les hôtes locaux, à accéder au fichier SWF recourant au protocole HTTPS.

### Exemple

Dans l'exemple suivant, vous hébergez un test de mathématique sur un domaine sécurisé, de façon à ce que seuls les étudiants enregistrés puissent y accéder. Vous avez également développé un ensemble de fichiers SWF, pour illustrer certains concepts, que vous avez placés dans un domaine non sécurisé. Vous souhaitez que les étudiants accèdent au test à partir du fichier SWF qui contient les informations relatives à un concept.

```
// This SWF file is at https://myEducationSite.somewhere.com/mathTest.swf
// Concept files are at http://myEducationSite.somewhere.com
System.security.allowInsecureDomain("myEducationSite.somewhere.com");
```

### Voir également

allowDomain (méthode security.allowDomain), exactSettings (propriété System.exactSettings)

## loadPolicyFile (méthode security.loadPolicyFile)

```
public static loadPolicyFile(url:String) : Void
```

Charge un fichier de régulation interdomaine à partir d'un emplacement spécifié par le paramètre url. Les fichiers de régulation font office de mécanisme d'autorisation. Ils permettent de charger des données vers des animations Flash depuis un serveur autre que celui sur lequel elles se trouvent.

Auparavant, Flash Player 7.0.14.0 recherchait les fichiers de régulation à un emplacement défini : /crossdomain.xml sur le serveur auquel la demande de chargement de données était envoyée. Dans le cas d'une connexion XMLSocket, Flash Player 7.0.14.0 recherchait l'emplacement /crossdomain.xml sur le port 80 d'un serveur HTTP, dans le sous-domaine auquel la connexion XMLSocket était envoyée. Flash Player 7.0.14.0 (ainsi que les lecteurs antérieurs) limitait les connexions XMLSocket aux ports 1024 et supérieurs.

Grâce à l'ajout de System.security.loadPolicyFile(), Flash Player 7.0.19.0 peut charger les fichiers de régulation à partir d'emplacements aléatoires, comme il est indiqué dans l'exemple suivant :

```
System.security.loadPolicyFile("http://foo.com/sub/dir/pf.xml");
```

De cette manière, Flash Player peut récupérer un fichier de régulation à l'URL spécifiée. Les permissions accordées par l'intermédiaire de ce fichier s'appliquent à l'ensemble du contenu, au même niveau ou à un niveau inférieur dans la hiérarchie virtuelle des répertoires du serveur. Le code suivant reprend l'exemple précédent :

```
loadVariables("http://foo.com/sub/dir/vars.txt") // allowed
loadVariables("http://foo.com/sub/dir/deep/vars2.txt") // allowed
loadVariables("http://foo.com/elsewhere/vars3.txt") // not allowed
```

Vous pouvez utiliser `loadPolicyFile()` pour charger un nombre illimité de fichiers de régulation. Dans le cas d'une requête impliquant un fichier de régulation, Flash Player attend que le téléchargement des fichiers de régulation soit terminé avant de rejeter une requête. En dernier recours, si aucun des fichiers de régulation spécifiés par `loadPolicyFile()` n'autorise la requête, Flash Player effectue une recherche à l'emplacement par défaut, /crossdomain.xml.

L'utilisation du protocole `xmlsocket` avec un numéro de port spécifique permet de récupérer directement les fichiers de régulation depuis un serveur XMLSocket, comme indiqué dans l'exemple suivant :

```
System.security.loadPolicyFile("xmlsocket://foo.com:414");
```

De cette manière, Flash Player peut récupérer un fichier de régulation au niveau du port et de l'hôte spécifiés. Il est possible d'utiliser n'importe quel port (et non plus uniquement le port 1024 ou un port de numéro supérieur). Lors de la connexion au port spécifié, Flash Player transmet `<policy-file-request />`, suivi d'un octet de terminaison nul `null`. Il est possible de configurer un serveur XMLSocket afin qu'il distribue des fichiers de régulation et des connexions XMLSocket normales en utilisant un port unique. Dans ce cas, le serveur attend de recevoir `<policy-file-request />` avant de transmettre un fichier de régulation. Il est également possible de configurer un serveur afin qu'il distribue les fichiers de régulation et les connexions standard via des ports différents. Dans ce cas, le serveur peut transmettre un fichier dès qu'une connexion est établie au niveau du port dédié aux fichiers de régulation. Le serveur doit renvoyer un octet nul à la fin du fichier de régulation avant de fermer la connexion. Si le serveur ne ferme pas la connexion, Flash Player y met fin après avoir reçu l'octet nul de terminaison `null`.

La syntaxe des fichiers de régulation transmis par un serveur XMLSocket est identique à celle des autres fichiers de régulation, à l'exception près que ces fichiers doivent également spécifier les ports auxquels ils permettent d'accéder. Un fichier de régulation transmis via un port dont le numéro est inférieur à 1024 peut autoriser l'accès à tous les ports. Un fichier de régulation transmis via le port 1024 ou supérieur ne peut définir l'accès qu'au port 1024 et aux ports supérieurs. Les ports accessibles sont spécifiés par l'attribut `"to-ports"` dans la balise `<allow-access-from>`. Il est possible d'utiliser des numéros de ports, des plages de ports et des caractères génériques. L'exemple suivant illustre un fichier de régulation XMLSocket :

```
<cross-domain-policy>
<allow-access-from domain="*" to-ports="507" />
<allow-access-from domain="*.foo.com" to-ports="507,516" />
<allow-access-from domain="*.bar.com" to-ports="516-523" />
<allow-access-from domain="www.foo.com" to-ports="507,516-523" />
<allow-access-from domain="www.bar.com" to-ports="*" />
</cross-domain-policy>
```

Les fichiers de régulation récupérés à partir de l'ancien emplacement par défaut (emplacement --/crossdomain.xml d'un serveur HTTP sur le port 80) permettent implicitement d'accéder aux ports 1024 et supérieurs. Il est impossible de récupérer un fichier de régulation autorisant des opérations XMLSocket depuis un autre emplacement sur un serveur HTTP. Les emplacements personnalisés des fichiers de régulation XMLSocket doivent être situés sur le serveur XMLSocket.

Etant donné que la possibilité de se connecter aux ports dont le numéro est inférieur à 1024 est une nouveauté, les fichiers de régulation chargés par l'intermédiaire de `loadPolicyFile()` doivent toujours autoriser cette connexion, même lors de la connexion d'un clip à son propre sous-domaine.

Pour plus d'informations, consultez les sections suivantes :

- Chapitre 17, « Understanding Security » (Comprendre la sécurité), dans *Learning ActionScript 2.0 dans Flash*
- La présentation technique de sécurité de Flash Player 8 disponible à l'adresse : http://www.macromedia.com/go/fp8_security
- La présentation technique des API relatives à la sécurité de Flash Player 8 disponible à l'adresse : http://www.macromedia.com/go/fp8_security_apis

**Disponibilité :** ActionScript 1.0; Flash Player 7.0.19.0

## Paramètres

*url*:`String` - Chaîne ; URL où réside le fichier de régulation interdomaine à charger.

# sandboxType (propriété security.sandboxType)

`public static sandboxType : String [lecture seule]`

Indique le type de sanbdox de sécurité dans lequel fonctionne le fichier SWF appelant.

`System.security.sandboxType` a l'une des valeurs suivantes :

- `remote` : Ce fichier SWF provient d'une URL et fonctionnera selon les règles basées sur le domaine sandbox.
- `localWithFile`: Ce fichier SWF est un fichier local qui n'a pas reçu la confiance de l'utilisateur et n'a pas été publié avec une désignation de mise en réseau. Ce fichier SWF peut lire à partir de sources locales de données mais ne peut pas communiquer avec Internet.
- `localWithNetwork`: Ce fichier SWF est un fichier local qui n'a pas reçu la confiance de l'utilisateur et a été publié avec une désignation de mise en réseau. Ce fichier SWF peut communiquer avec Internet mais ne peut pas lire à partir de sources locales de données.
- `localTrusted`: Ce fichier SWF est un fichier local qui a reçu la confiance de l'utilisateur en utilisant soit le gestionnaire de paramètres, soit un fichier de configuration FlashPlayerTrust. Ce fichier SWF peut aussi bien lire à partir de sources locales de données qu'il peut communiquer avec Internet.

Notez que cette propriété peut être examinée depuis un fichier SWF de quelque version que ce soit, mais qu'elle est prise en charge uniquement dans Flash Player 8 ou supérieur. Cette convention inhabituelle signifie que vous pouvez par exemple examiner cette propriété à partir d'un fichier SWF de la version 7 lu sous Flash Player 8. Cette prise en charge de toutes les versions signifie que si vous publiez pour une version antérieure à 8, au moment de la publication, vous ne saurez pas si cette propriété sera prise en charge ou non au moment de la lecture. Ainsi, dans un fichier SWF de version 7 ou inférieure, vous trouverez peut-être que cette propriété a une valeur indéfinie. Ceci ne devrait se produire que lorsque la version du player (indiquée par `System.capabilities.version`) est inférieure à 8. Dans ce cas, vous pouvez déterminer le type de sandbox selon que l'URL de votre fichier SWF est un fichier local ou non. Si c'est le cas, vous pouvez supposer que Flash Player classera votre fichier SWF en tant que `"localTrusted"` (avant Flash Player 8, tous les contenus locaux étaient ainsi traités). Dans le cas contraire, vous pouvez partir du principe que Flash Player classera votre fichier SWF en tant que `"remote"`.

Pour plus d'informations, consultez les sections suivantes :

- Chapitre 17, « Understanding Security » (Comprendre la sécurité), dans *Learning ActionScript 2.0 dans Flash*
- La présentation technique de sécurité de Flash Player 8 disponible à l'adresse : http://www.macromedia.com/go/fp8_security

- La présentation technique des API relatives à la sécurité de Flash Player 8 disponible à l'adresse : http://www.macromedia.com/go/fp8_security_apis

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

# Sélection

```
Object
  |
  +-Selection
```

```
public class Selection
extends Object
```

La classe Selection vous permet de définir et de contrôler le champ de texte dans lequel se trouve le point d'insertion (à savoir, le champ ayant le focus). Les index de plages de sélection sont basés sur zéro (par exemple, la première position est 0, la deuxième position est 1, etc.).

Il n'existe aucune fonction constructeur pour la classe Selection car un seul champ ayant reçu le focus peut être défini à la fois.

**Disponibilité :** ActionScript 1.0 ; Flash Player 5

## Résumé des propriétés

*Propriétés héritées de la classe Object*

| |
|---|
| constructeur (propriété Object.constructor), __proto__ (Object.__proto__ property), prototype (Object.prototype, propriété), __resolve (Object.__resolve, propriété) |

## Résumé des événements

| Evénement | Description |
|---|---|
| onSetFocus = function([oldfocus:Object], [newfocus:Object]) {} | Notifié lorsque le focus d'entrée change. |

## Résumé de la méthode

| Modificateurs | Signature | Description |
|---|---|---|
| static | addListener(listener:Object) : Void | Enregistre un objet pour recevoir les notifications de changement du focus clavier. |
| static | getBeginIndex() : Number | Renvoie l'index au début de la plage de sélection. |
| static | getCaretIndex() : Number | Renvoie l'index de la position du point d'insertion clignotant (caret). |
| static | getEndIndex() : Number | Renvoie l'index de fin de la plage de sélection ayant actuellement le focus. |
| static | getFocus() : String | Renvoie une chaîne spécifiant le chemin cible de l'objet ayant le focus. |
| static | removeListener(listener:Object) : Boolean | Supprime un objet précédemment enregistré avec Selection.addListener(). |
| static | setFocus(newFocus:Object) : Boolean | Donne le focus au champ de texte, bouton ou clip sélectionnable (modifiable) spécifié par le paramètre newFocus. |
| static | setSelection(beginIndex:Number, endIndex:Number) : Void | Définit la plage de sélection du champ de texte ayant actuellement le focus. |

*Méthodes héritées de la classe Object*

addProperty (méthode Object.addProperty), hasOwnProperty (méthode Object.hasOwnProperty), isPropertyEnumerable (méthode Object.isPropertyEnumerable), isPrototypeOf (méthode Object.isPrototypeOf), registerClass (méthode Object.registerClass), toString (méthode Object.toString), unwatch (méthode Object.unwatch), valueOf (méthode Object.valueOf), watch (méthode Object.watch)

# addListener (méthode Selection.addListener)

```
public static addListener(listener:Object) : Void
```

Enregistre un objet pour recevoir les notifications de changement du focus clavier. Lorsque le focus change (par exemple, à chaque fois que `Selection.setFocus()` est appelé), tous les objets d'écoute enregistrés avec `addListener()`, ont leur méthode `onSetFocus` appelée. Plusieurs objets peuvent écouter les notifications de changement de focus. Si l'écouteur spécifié est déjà enregistré, aucun changement ne se produit.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

## Paramètres

*`listener`*`:Object` - Nouvel objet avec une méthode `onSetFocus`.

## Exemple

Dans l'exemple suivant, vous créez deux champs texte de saisie lors de l'exécution et définissez les bordures de ces champs sur `true`. Ce code crée un nouvel objet ActionScript (générique) appelé `focusListener`. Cet objet définit pour lui-même une propriété `onSetFocus` à laquelle il affecte une fonction. La fonction prend en compte deux paramètres : une référence au champ de texte qui a perdu le focus et une référence au champ de texte qui a reçu le focus. La fonction définit la propriété `border` du champ de texte qui a perdu le focus sur `false` et définit la propriété border du champ de texte qui a reçu le focus sur `true`:

```
this.createTextField("one_txt", 99, 10, 10, 200, 20);
this.createTextField("two_txt", 100, 10, 50, 200, 20);
one_txt.border = true;
one_txt.type = "input";
two_txt.border = true;
two_txt.type = "input";

var focusListener:Object = new Object();
focusListener.onSetFocus = function(oldFocus_txt, newFocus_txt) {
  oldFocus_txt.border = false;
  newFocus_txt.border = true;
};
Selection.addListener(focusListener);
```

Lorsque vous testez le fichier SWF, essayez d'utiliser la touche de tabulation pour passer d'un champ texte à l'autre. Sélectionnez Contrôle > Désactiver les raccourcis clavier de façon à pouvoir changer le focus avec la touche de tabulation.

## Voir également

`setFocus (méthode Selection.setFocus)`

# getBeginIndex (méthode Selection.getBeginIndex)

```
public static getBeginIndex() : Number
```

Renvoie l'index au début de la plage de sélection. Si aucun index n'existe ou si aucun champ de texte n'a actuellement le focus, la méthode renvoie -1. Les index de plages de sélection sont basés sur zéro (par exemple, la première position est 0, la deuxième position est 1, etc.).

**Disponibilité :** ActionScript 1.0 ; Flash Player 5

## Valeur renvoyée

`Number` - Entier.

## Exemple

L'exemple suivant crée un champ texte lors de l'exécution et définit ses propriétés. Une option de menu contextuel est ajoutée pour permettre de mettre le texte sélectionné en majuscules.

```
this.createTextField("output_txt", this.getNextHighestDepth(), 0, 0, 300,
  200);
output_txt.multiline = true;
output_txt.wordWrap = true;
output_txt.border = true;
output_txt.type = "input";
output_txt.text = "Enter your text here";
var my_cm:ContextMenu = new ContextMenu();
my_cm.customItems.push(new ContextMenuItem("Uppercase...", doUppercase));
function doUppercase():Void {
  var startIndex:Number = Selection.getBeginIndex();
  var endIndex:Number = Selection.getEndIndex();
  var stringToUppercase:String = output_txt.text.substring(startIndex,
  endIndex);
  output_txt.replaceText(startIndex, endIndex,
  stringToUppercase.toUpperCase());
}
output_txt.menu = my_cm;
```

La méthode `MovieClip.getNextHighestDepth()` utilisée dans cet exemple requiert Flash Player 7 ou version ultérieure. Si votre fichier SWF comporte un composant de la version 2, utilisez la classe DepthManager des composants de la version 2 au lieu de la méthode `MovieClip.getNextHighestDepth()`.

Vous trouverez également un exemple dans le fichier Strings.fla du dossier d'exemples ActionScript. Les chemins type de ce dossier sont :

- Windows : *lecteur d'amorçage*\Program Files\Macromedia\Flash 8\Samples and Tutorials\Samples\ActionScript
- Macintosh : *disque dur Macintosh*/Applications/Macromedia Flash 8/Samples and Tutorials/Samples/ActionScript

### Voir également

getEndIndex (méthode Selection.getEndIndex)

## getCaretIndex (méthode Selection.getCaretIndex)

```
public static getCaretIndex() : Number
```

Renvoie l'index de la position du point d'insertion clignotant (caret). Si aucun point d'insertion clignotant ne s'affiche, la méthode renvoie -1. Les index de plages de sélection sont basés sur zéro (par exemple, la première position est 0, la deuxième position est 1, etc.).

**Disponibilité :** ActionScript 1.0 ; Flash Player 5

### Valeur renvoyée

`Number` - Entier.

### Exemple

L'exemple suivant crée et définit les propriétés d'un champ texte lors de l'exécution. La méthode `getCaretIndex()` permet de renvoyer l'index du signe circonflexe et affiche sa valeur dans un autre champ texte.

```
this.createTextField("pos_txt", this.getNextHighestDepth(), 50, 20, 100,
  22);
this.createTextField("content_txt", this.getNextHighestDepth(), 50, 50,
  400, 300);
content_txt.border = true;
content_txt.type = "input";
content_txt.wordWrap = true;
content_txt.multiline = true;
content_txt.onChanged = getCaretPos;

var keyListener:Object = new Object();
keyListener.onKeyUp = getCaretPos;
Key.addListener(keyListener);

var mouseListener:Object = new Object();
mouseListener.onMouseUp = getCaretPos;
Mouse.addListener(mouseListener);

function getCaretPos() {
  pos_txt.text = Selection.getCaretIndex();
}
```

La méthode `MovieClip.getNextHighestDepth()` utilisée dans cet exemple requiert Flash Player 7 ou version ultérieure. Si votre fichier SWF comporte un composant de la version 2, utilisez la classe DepthManager des composants de la version 2 au lieu de la méthode `MovieClip.getNextHighestDepth()`.

Vous trouverez également un exemple dans le fichier Strings.fla du dossier d'exemples ActionScript. Les chemins type de ce dossier sont :

- Windows : *lecteur d'amorçage*\Program Files\Macromedia\Flash 8\Samples and Tutorials\Samples\ActionScript
- Macintosh : *disque dur Macintosh*/Applications/Macromedia Flash 8/Samples and Tutorials/Samples/ActionScript

# getEndIndex (méthode Selection.getEndIndex)

```
public static getEndIndex() : Number
```

Renvoie l'index de fin de la plage de sélection ayant actuellement le focus. Si aucun index n'existe ou si aucune plage de sélection n'a actuellement le focus, la méthode renvoie -1. Les index de plages de sélection sont basés sur zéro (par exemple, la première position est 0, la deuxième position est 1, etc.).

**Disponibilité :** ActionScript 1.0 ; Flash Player 5

## Valeur renvoyée

`Number` - Entier.

## Exemple

Cet exemple est tiré du fichier Strings.fla figurant dans le dossier d'exemples ActionScript.

```
// define the function which converts the selected text in an instance,
// and convert the string to upper or lower case.
function convertCase(target, menuItem) {
  var beginIndex:Number = Selection.getBeginIndex();
  var endIndex:Number = Selection.getEndIndex();
  var tempString:String;
  // make sure that text is actually selected.
  if (beginIndex>-1 && endIndex>-1) {
  // set the temporary string to the text before the selected text.
  tempString = target.text.slice(0, beginIndex);
  switch (menuItem.caption) {
  case 'Uppercase...' :
    // if the user selects the "Uppercase..." context menu item,
    // convert the selected text to upper case.
    tempString += target.text.substring(beginIndex,
  endIndex).toUpperCase();
```

```
    break;
  case 'Lowercase...' :
    tempString += target.text.substring(beginIndex,
  endIndex).toLowerCase();
    break;
  }
  // append the text after the selected text to the temporary string.
  tempString += target.text.slice(endIndex);
  // set the text in the target text field to the contents of the temporary
  string.
  target.text = tempString;
  }
}
```

Reportez-vous au fichier Strings.fla pour l'ensemble du script. Les chemins type du dossier d'exemples ActionScript sont :

- Windows : *lecteur d'amorçage*\Program Files\Macromedia\Flash 8\Samples and Tutorials\Samples\ActionScript
- Macintosh : *disque dur Macintosh*/Applications/Macromedia Flash 8/Samples and Tutorials/Samples/ActionScript

### Voir également

`getBeginIndex (méthode Selection.getBeginIndex)`

## getFocus (méthode Selection.getFocus)

```
public static getFocus() : String
```

Renvoie une chaîne spécifiant le chemin cible de l'objet ayant le focus.

- Si un objet TextField a le focus et dispose d'un nom d'occurrence, cette méthode renvoie le chemin cible de l'objet TextField. Sinon, elle renvoie le nom de la variable de l'objet TextField.
- Si un objet bouton ou le clip d'un bouton a le focus, cette méthode renvoie le chemin cible de l'objet bouton ou du clip du bouton.
- Si ni un objet TextField, ni un bouton, une occurrence de composant ou un clip de bouton n'a le focus, cette méthode renvoie la valeur `null`.

**Disponibilité :** ActionScript 1.0 ; Flash Player 5

### Valeur renvoyée

`String` - Chaîne ou `null`.

### Exemple

L'exemple suivant affiche le chemin cible de la sélection ayant le focus dans une occurrence de composant TextArea. Ajoutez plusieurs occurrences de composant ou de bouton, de champ texte et de clip sur la Scène. Ajoutez plusieurs occurrences de composant ou de bouton, de champ texte et de clip dans le fichier SWF. Ajoutez le code ActionScript suivant à votre fichier AS ou FLA.

```
var focus_ta:mx.controls.TextArea;
my_mc.onRelease = function() {};
my_btn.onRelease = function() {};

var keyListener:Object = new Object();
keyListener.onKeyDown = function() {
  if (Key.isDown(Key.SPACE)) {
  focus_ta.text = Selection.getFocus()+newline+focus_ta.text;
  }
};
Key.addListener(keyListener);
```

Testez le fichier SWF et utilisez la touche de tabulation pour parcourir les occurrences sur la Scène. Assurez-vous que Contrôle > Désactiver les raccourcis clavier est sélectionné dans l'environnement de test.

### Voir également

`onSetFocus (Selection.onSetFocus, écouteur d'événement)`, `setFocus (méthode Selection.setFocus)`

## onSetFocus (Selection.onSetFocus, écouteur d'événement)

```
onSetFocus = function([oldfocus:Object], [newfocus:Object]) {}
```

Notifié lorsque le focus d'entrée change. Pour utiliser cet écouteur, vous devez créer un objet écouteur. Vous pouvez alors définir une fonction pour cet écouteur et utiliser Selection.addListener() pour enregistrer l'écouteur avec l'objet Selection, comme dans le code suivant :

```
var someListener:Object = new Object();
someListener.onSetFocus = function () {
  // statements
}
Selection.addListener(someListener);
```

Les écouteurs permettent à divers blocs de code de coopérer car plusieurs écouteurs peuvent recevoir une notification sur un événement unique.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

## Paramètres

*oldfocus*:Object [facultatif] Objet qui perd le focus.

*newfocus*:Object [facultatif] - Objet qui reçoit le focus.

## Exemple

L'exemple suivant démontre comment déterminer les changements de focus de plusieurs champs texte créés de façon dynamique dans un fichier SWF. Entrez le code ActionScript suivant dans un fichier FLA ou AS, puis testez le document :

```
this.createTextField("one_txt", 1, 0, 0, 100, 22);
this.createTextField("two_txt", 2, 0, 25, 100, 22);
this.createTextField("three_txt", 3, 0, 50, 100, 22);
this.createTextField("four_txt", 4, 0, 75, 100, 22);

for (var i in this) {
  if (this[i] instanceof TextField) {
  this[i].border = true;
  this[i].type = "input";
  }
}

this.createTextField("status_txt", this.getNextHighestDepth(), 200, 10,
  300, 100);
status_txt.html = true;
status_txt.multiline = true;

var someListener:Object = new Object();
someListener.onSetFocus = function(oldFocus, newFocus) {
  status_txt.htmlText = "<b>setFocus triggered</b>";
  status_txt.htmlText += "<textformat tabStops='[20,80]'>";
  status_txt.htmlText += " \toldFocus:\t"+oldFocus;
  status_txt.htmlText += " \tnewFocus:\t"+newFocus;
  status_txt.htmlText += " \tgetFocus:\t"+Selection.getFocus();
  status_txt.htmlText += "</textformat>";
};
Selection.addListener(someListener);
```

La méthode `MovieClip.getNextHighestDepth()` utilisée dans cet exemple requiert Flash Player 7 ou version ultérieure. Si votre fichier SWF comporte un composant de la version 2, utilisez la classe DepthManager des composants de la version 2 au lieu de la méthode `MovieClip.getNextHighestDepth()`.

## Voir également

addListener (méthode Selection.addListener), setFocus (méthode Selection.setFocus)

# removeListener (méthode Selection.removeListener)

`public static removeListener(listener:Object) : Boolean`

Supprime un objet précédemment enregistré avec `Selection.addListener()`.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

## Paramètres

*`listener`*`:Object` - Objet qui ne recevra plus de notifications de focus.

## Valeur renvoyée

`Boolean` - Si `listener` a été supprimé avec succès, la méthode renvoie une valeur `true`. Si l'écouteur `listener` n'a pas été supprimé avec succès (par exemple, si `listener` n'était pas sur la liste d'écouteurs de l'objet de la Sélection, la méthode renvoie une valeur de `false`.

## Exemple

Le code ActionScript suivant crée de façon dynamique plusieurs occurrences de champ texte. Lorsque vous sélectionnez un champ texte, les informations correspondantes s'affichent dans le panneau de sortie. Lorsque vous cliquez sur l'occurrence `remove_btn`, l'écouteur est supprimé et aucune information ne s'affiche dans le panneau de sortie.

```
this.createTextField("one_txt", 1, 0, 0, 100, 22);
this.createTextField("two_txt", 2, 0, 25, 100, 22);
this.createTextField("three_txt", 3, 0, 50, 100, 22);
this.createTextField("four_txt", 4, 0, 75, 100, 22);

for (var i in this) {
  if (this[i] instanceof TextField) {
  this[i].border = true;
  this[i].type = "input";
  }
}

var selectionListener:Object = new Object();
selectionListener.onSetFocus = function(oldFocus, newFocus) {
  trace("Focus shifted from "+oldFocus+" to "+newFocus);
};
Selection.addListener(selectionListener);

remove_btn.onRelease = function() {
  trace("removeListener invoked");
  Selection.removeListener(selectionListener);
};
```

### Voir également

addListener (méthode Selection.addListener)

## setFocus (méthode Selection.setFocus)

```
public static setFocus(newFocus:Object) : Boolean
```

Donne le focus au champ de texte, bouton ou clip sélectionnable (modifiable) spécifié par le paramètre `newFocus`. Si la valeur `null` ou `undefined` est transmise, le focus actuel est supprimé.

**Disponibilité :** ActionScript 1.0 ; Flash Player 5

### Paramètres

*`newFocus`*`:Object` - Objet tel qu'une occurrence de bouton, de clip ou de champ texte, ou chaîne spécifiant le chemin de l'une de ces occurrences. Si vous transmettez un littéral de chaîne spécifiant un chemin, placez le chemin entre guillemets (" "). Vous pouvez utiliser la notation avec point ou avec barre oblique pour spécifier le chemin. Si vous utilisez ActionScript 2.0, vous devez utiliser la notation avec point. Vous pouvez utiliser un chemin relatif ou absolu.

### Valeur renvoyée

`Boolean` - Valeur booléenne ; `true` si la tentative de focus réussit, `false` si elle échoue.

### Exemple

Dans l'exemple suivant, le champ texte donne le focus au champ texte `username_txt` lorsque ce dernier s'affiche dans une fenêtre de navigateur. Si l'utilisateur ne remplit pas l'un des champs texte requis (`username_txt` et `password_txt`), le curseur se place automatiquement dans le champ texte présentant des données manquantes. Par exemple, si l'utilisateur ne tape rien dans le champ texte `username_txt` et clique sur le bouton submit, un message d'erreur s'affiche et le curseur se place dans le champ texte `username_txt`.

```
this.createTextField("status_txt", this.getNextHighestDepth(), 100, 70,
  100, 22);
this.createTextField("username_txt", this.getNextHighestDepth(), 100, 100,
  100, 22);
this.createTextField("password_txt", this.getNextHighestDepth(), 100, 130,
  100, 22);
this.createEmptyMovieClip("submit_mc", this.getNextHighestDepth());
submit_mc.createTextField("submit_txt", this.getNextHighestDepth(), 100,
  160, 100, 22);
submit_mc.submit_txt.autoSize = "center";
submit_mc.submit_txt.text = "Submit";
```

```
submit_mc.submit_txt.border = true;
submit_mc.onRelease = checkForm;
username_txt.border = true;
password_txt.border = true;
username_txt.type = "input";
password_txt.type = "input";
password_txt.password = true;
Selection.setFocus("username_txt");
//
function checkForm():Boolean {
  if (username_txt.text.length == 0) {
  status_txt.text = "fill in username";
  Selection.setFocus("username_txt");
  return false;
  }
  if (password_txt.text.length == 0) {
  status_txt.text = "fill in password";
  Selection.setFocus("password_txt");
  return false;
  }
  status_txt.text = "success!";
  Selection.setFocus(null);
  return true;
}
```

La méthode `MovieClip.getNextHighestDepth()` utilisée dans cet exemple requiert Flash Player 7 ou version ultérieure. Si votre fichier SWF comporte un composant de la version 2, utilisez la classe DepthManager des composants de la version 2 au lieu de la méthode `MovieClip.getNextHighestDepth()`.

### Voir également

`getFocus (méthode Selection.getFocus)`

## setSelection (méthode Selection.setSelection)

```
public static setSelection(beginIndex:Number, endIndex:Number) : Void
```

Définit la plage de sélection du champ de texte ayant actuellement le focus. La nouvelle plage de sélection commence à l'index spécifié dans le paramètre `beginIndex` et se termine à l'index spécifié dans le paramètre `endIndex`. Les index de plages de sélection sont basés sur zéro (par exemple, la première position est 0, la deuxième position est 1, etc.). Cette méthode n'a aucun effet si aucun champ de texte n'a actuellement le focus.

**Disponibilité :** ActionScript 1.0 ; Flash Player 5

### Paramètres

*`beginIndex`*`:Number` - Index de début de la plage de sélection.

*endIndex*:Number - Index de fin de la plage de sélection.

### Exemple

Le code ActionScript suivant crée un champ texte lors de l'exécution et place une chaîne dans ce dernier. Il donne ensuite le focus au champ texte et sélectionne une plage de caractère dans ce dernier.

```
this.createTextField("myText_txt", 99, 10, 10, 200, 30);
myText_txt.text = "this is my text";
this.onEnterFrame = function () {
  Selection.setFocus("myText_txt");
  Selection.setSelection(0, 3);
  delete this.onEnterFrame;
}
```

L'exemple suivant illustre dans quelle mesure le paramètre `endIndex` n'est pas global. Pour sélectionner le premier caractère, vous devez utiliser un paramètre `endIndex` d'une valeur de 1, non pas 0. Si vous définissez le paramètre `endIndex` sur 0, rien ne sera sélectionné.

```
this.createTextField("myText_txt", 99, 10, 10, 200, 30);
myText_txt.text = "this is my text";
this.onEnterFrame = function () {
  Selection.setFocus("myText_txt");
  Selection.setSelection(0, 1);
  delete this.onEnterFrame;
}
```

# SharedObject

```
Object
  |
  +-SharedObject
```

```
public dynamic class SharedObject
extends Object
```

La classe SharedObject permet de lire et stocker des quantités limitées de données sur l'ordinateur d'un utilisateur. Les objets partagés permettent de partager les données en temps réel, entre des objets persistants, sur l'ordinateur de l'utilisateur. Les objets partagés locaux sont similaires aux cookies d'un navigateur.

Les objets partagés peuvent être utilisés de trois façons :

- Un jeu qui stocke les scores les plus élevés de l'utilisateur. Le jeu peut fournir des données personnalisées aux utilisateurs, telles que le nom d'utilisateur et les scores les plus élevés, sans avoir à dédier une partie de l'espace de stockage du serveur.

- Une application d'annuaire téléphonique qui peut fonctionner en ligne ou hors connexion. L'annuaire, livré en tant qu'application de projection, pourrait contenir un cache de données local regroupant les noms et les numéros de téléphone entrés par l'utilisateur. Lorsqu'une connexion Internet est disponible, l'application pourrait récupérer les informations les plus à jour sur le serveur. En l'absence de connexion, l'application utilise les données les plus récentes qui figurent dans les objets partagés.
- Les préférences utilisateur ou les données de suivi pour un site Web complexe, telles qu'un enregistrement des articles lus par un utilisateur sur un site d'information. Le suivi de ces informations permet de différencier l'affichage des articles qui ont été lus de ceux qui sont nouveaux ou non lus. Le stockage de ces informations sur l'ordinateur de l'utilisateur réduit la charge du serveur.

Les objets partagés locaux maintiennent la persistance locale. Par exemple, vous pouvez appeler `SharedObject.getLocal()` pour créer un objet partagé contenant le score le plus élevé du jeu. Dans la mesure où l'objet partagé est persistant localement, Flash enregistre ses attributs de données sur l'ordinateur de l'utilisateur à la fermeture du jeu. Lorsque le jeu est à nouveau ouvert, il affiche le score le plus élevé de la session précédente. Une autre méthode consiste à définir les propriétés de l'objet partagé sur `null` avant de fermer le jeu. Lorsque le fichier SWF est exécuté de nouveau, le jeu s'ouvre sans afficher le score le plus élevé.

Pour créer un objet partagé localement, utilisez la syntaxe suivante :

```
var so:SharedObject = SharedObject.getLocal("userHighScore");
so.data.highScore = new Number();
so.flush();
```

Dans cet exemple, l'objet partagé est *purgé* de façon explicite ou écrit sur un disque. Lorsqu'une application se ferme, les objets partagés sont purgés de façon automatique. Cependant, ceci est indiqué ici pour démontrer la procédure d'écriture des données sur un disque.

*Eléments importants relatifs à l'espace disque local :* Les objets partagés locaux peuvent être particulièrement utiles, mais ils sont soumis à certaines limites qui doivent être prises en compte lors de la conception de votre application. Certains fichiers SWF ne sont pas autorisés à écrire des objets locaux partagés et certaines données stockées dans des objets partagés locaux peuvent être supprimées à votre insu. Les utilisateurs de Flash Player peuvent gérer l'espace disque disponible pour des domaines spécifiques ou l'ensemble des domaines. Lorsque des utilisateurs réduisent le montant d'espace disque disponible, certains objets locaux partagés peuvent être supprimés. Les utilisateurs de Flash Player disposent également de contrôles de confidentialité qui peuvent empêcher les domaines tiers (domaines autres que le domaine figurant dans la barre d'adresses du navigateur) de lire ou d'écrire des objets locaux partagés.

*Remarque* : un contenu local peut toujours écrire s des objets partagés par un tiers sur un disque, quoiqu'il soit interdit à un domaine tiers d'écrire des objets partagés sur un disque.

Macromedia recommande de vérifier tout dysfonctionnement relatif au montant d'espace disque disponible et aux contrôles de confidentialité de l'utilisateur. Effectuez ces vérifications lorsque vous appelez `getLocal()` et `flush()`:

- `SharedObject.getLocal()` - Cette méthode renvoie `null` lorsque l'utilisateur a désactivé les objets partagés des tiers et lorsque votre domaine SWF ne correspond pas au domaine de la barre d'adresses du navigateur.
- `SharedObject.flush()` - Cette méthode renvoie `false` lorsque l'utilisateur a désactivé des objets partagés pour votre domaine ou pour l'ensemble des domaines. Elle renvoie `"pending"` (en attente) lorsqu'un espace supplémentaire est requis et l'utilisateur doit décider de façon interactive s'il doit autoriser une augmentation.

Si votre fichier SWF tente de créer ou modifier des objets locaux partagés, assurez-vous que le fichier SWF fait au moins 215 pixels de large et 138 pixels de hauteur (ce qui constitue les dimensions minimales d'affichage de la boîte de dialogue qui suggère à l'utilisateur d'augmenter sa limite locale de stockage des objets partagés locaux). Si votre fichier SWF est inférieur à ces dimensions et si une augmentation de la limite de stockage est nécessaire, `SharedObject.flush()` échoue, renvoie `"pending"`, mais appelle par la suite votre gestionnaire `SharedObject.onStatus` avec le résultat `"SharedObject.Flush.Failed"`.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

## Voir également

`getLocal (méthode SharedObject.getLocal)`, `flush (méthode SharedObject.flush)`, `onStatus (gestionnaire SharedObject.onStatus)`

## Résumé des propriétés

| Modificateurs | Propriété | Description |
|---|---|---|
| | `data:Object` | La collecte des attributs associés à la propriété `data` de l'objet. Ces attributs peuvent être partagés et/ou stockés. |

*Propriétés héritées de la classe Object*

`constructeur (propriété Object.constructor)`, `__proto__ (Object.__proto__ property)`, `prototype (Object.prototype, propriété)`, `__resolve (Object.__resolve, propriété)`

### Résumé des événements

| Evénement | Description |
|---|---|
| `onStatus = function(infoObject:Object) {}` | Appelé dès qu'une erreur, un avertissement ou une note d'informations est publié pour un objet partagé. |

### Résumé de la méthode

| Modificateurs | Signature | Description |
|---|---|---|
| | `clear() : Void` | Purge toutes les données de l'objet partagé, puis supprime cet objet du disque. |
| | `flush([minDiskSpace:Number]) : Object` | Ecrit immédiatement un objet partagé persistant localement dans un fichier local. |
| `static` | `getLocal(name:String, [localPath:String], [secure:Boolean]) : SharedObject` | Renvoie une référence à un objet partagé persistant localement disponible uniquement pour le client actuel. |
| | `getSize() : Number` | Obtient la taille actuelle de l'objet partagé, en octets. |

*Méthodes héritées de la classe Object*

`addProperty (méthode Object.addProperty), hasOwnProperty (méthode Object.hasOwnProperty), isPropertyEnumerable (méthode Object.isPropertyEnumerable), isPrototypeOf (méthode Object.isPrototypeOf), registerClass (méthode Object.registerClass), toString (méthode Object.toString), unwatch (méthode Object.unwatch), valueOf (méthode Object.valueOf), watch (méthode Object.watch)`

# clear (méthode SharedObject.clear)

```
public clear() : Void
```

Purge toutes les données de l'objet partagé, puis supprime cet objet du disque. La référence à `my_so` est toujours active et `my_so` est désormais vide.

**Disponibilité :** ActionScript 1.0 ; Flash Player 7

## Exemple

L'exemple suivant définit les données dans l'objet partagé, puis vide l'ensemble des données en provenance de l'objet partagé.

```
var my_so:SharedObject = SharedObject.getLocal("superfoo");
```

```
my_so.data.name = "Hector";
trace("before my_so.clear():");
for (var prop in my_so.data) {
  trace("\t"+prop);
}
trace("");
my_so.clear();
trace("after my_so.clear():");
for (var prop in my_so.data) {
  trace("\t"+prop);
}
```

Cet exemple de code ActionScript affiche le message suivant dans le panneau de sortie :

```
before my_so.clear():
  name

after my_so.clear():
```

## data (propriété SharedObject.data)

```
public data : Object
```

La collecte des attributs associés à la propriété `data` de l'objet. Ces attributs peuvent être partagés et/ou stockés. Chaque attribut peut être un objet d'un quelconque tableau, nombre, valeur booléenne, etc. de type ActionScript ou JavaScript. Par exemple, les lignes suivantes affectent des valeurs à différents aspects d'un objet partagé :

```
var items_array:Array = new Array(101, 346, 483);
var currentUserIsAdmin:Boolean = true;
var currentUserName:String = "Ramona";

var my_so:SharedObject = SharedObject.getLocal("superfoo");
my_so.data.itemNumbers = items_array;
my_so.data.adminPrivileges = currentUserIsAdmin;
my_so.data.userName = currentUserName;

for (var prop in my_so.data) {
  trace(prop+": "+my_so.data[prop]);
}
```

Tous les attributs de la propriété `data` de données d'un objet partagé sont enregistrés si l'objet est persistant, et l'objet partagé contient les informations suivantes :

```
userName: Ramona
adminPrivileges: true
itemNumbers: 101,346,483
```

*Remarque :* N'affectez pas directement de valeurs à la propriété de données `data` d'un objet partagé, tel que dans `so.data = someValue`; Flash ignore ces affectations.

Pour supprimer des attributs pour des objets partagés locaux, utilisez un code tel que `delete so.data.attributeName`; le fait de définir un attribut sur `null` ou `undefined` pour un objet partagé local ne supprime pas l'attribut.

Pour créer des valeurs *privées* pour un objet partagé, des valeurs qui ne sont disponibles que pour l'occurrence du client alors que l'objet est utilisé et qui ne sont pas enregistrées avec l'objet lorsqu'il est fermé, créez des propriétés qui ne sont pas des données `data` nommées pour les enregistrer, comme l'illustre l'exemple suivant :

```
var my_so:SharedObject = SharedObject.getLocal("superfoo");
my_so.favoriteColor = "blue";
my_so.favoriteNightClub = "The Bluenote Tavern";
my_so.favoriteSong = "My World is Blue";

for (var prop in my_so) {
  trace(prop+": "+my_so[prop]);
}
```

L'objet partagé contient les données suivantes :

```
favoriteSong: My World is Blue
favoriteNightClub: The Bluenote Tavern
favoriteColor: blue
data: [object Object]
```

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

## Exemple

L'exemple suivant enregistre du texte provenant de l'occurrence de composant TextInput dans un objet partagé appelé `my_so` (pour consulter l'intégralité de l'exemple, consultez `SharedObject.getLocal()`) :

```
// Create a listener object and function for the <enter> event.
var textListener:Object = new Object();
textListener.enter = function(eventObj:Object) {
  my_so.data.myTextSaved = eventObj.target.text;
  my_so.flush();
};
```

## Voir également

# flush (méthode SharedObject.flush)

```
public flush([minDiskSpace:Number]) : Object
```

Ecrit immédiatement un objet partagé persistant localement dans un fichier local. Si vous n'utilisez pas cette méthode, Flash écrit l'objet partagé dans un fichier lorsque la session d'objet partagé se termine, c'est-à-dire lorsque le fichier SWF est fermé, lorsque la place de l'objet partagé est récupérée étant donné qu'il n'y a plus aucune référence à ce dernier ou lorsque vous appelez `SharedObject.clear()`.

Si cette méthode renvoie `"pending"`, Flash Player ouvre une boîte de dialogue demandant à l'utilisateur d'augmenter la quantité d'espace disque disponible pour les objets de ce domaine. Pour aménager un espace permettant à l'objet partagé de se développer lors d'un enregistrement ultérieur, en évitant le retour de valeurs `"pending"`, transmettez une valeur pour `minimumDiskSpace`. Lorsque Flash essaie d'écrire un fichier, il recherche le nombre d'octets transmis à `minimumDiskSpace` au lieu de rechercher l'espace nécessaire à l'enregistrement de l'objet partagé à sa taille actuelle.

Par exemple, si vous espérez qu'un objet partagé se développe jusqu'à une taille maximum de 500 octets, même s'il peut être au départ beaucoup plus petit, transmettez 500 pour `minimumDiskSpace`. Si Flash demande à l'utilisateur d'affecter de l'espace disque à l'objet partagé, il demandera 500 octets. Une fois que l'utilisateur affecte l'espace nécessaire, Flash ne demande pas davantage d'espace pour les tentatives ultérieures de purge de l'objet (tant que sa taille ne dépasse pas 500 octets).

Une fois que l'utilisateur répond à la boîte de dialogue, cette méthode est appelée de nouveau et renvoie `true` ou `false`; `SharedObject.onStatus` est également appelé avec une propriété de `code` de `SharedObject.Flush.Success` ou `SharedObject.Flush.Failed`.

Pour plus d'informations, consultez « Eléments importants de l'espace disque local » dans la présentation de la classe SharedObject.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

## Paramètres

*`minDiskSpace`*`:Number` [facultatif] - Entier spécifiant le nombre d'octets qui doivent être affectés à cet objet. La valeur par défaut est 0.

## Valeur renvoyée

`Object` - Valeur booléenne : `true` ou `false` ; ou une chaîne « en attente » `"pending"`, comme il est décrit dans la liste suivante :

- Si l'utilisateur a autorisé l'enregistrement des informations locales pour les objets de ce domaine, et si l'espace affecté est suffisant pour enregistrer l'objet, cette méthode renvoie `true`. (Si vous avez transmis une valeur pour `minimumDiskSpace`, l'espace affecté doit être au moins égal à cette valeur pour renvoyer `true`).
- Si l'utilisateur a autorisé l'enregistrement des informations locales pour les objets de ce domaine, mais si l'espace affecté est insuffisant pour enregistrer l'objet, cette méthode renvoie « en attente » `"pending"`.
- Si l'utilisateur a toujours refusé l'enregistrement d'informations locales pour les objets de ce domaine, ou si Flash ne peut pas enregistrer l'objet pour une raison quelconque, cette méthode renvoie `false`.

*Remarque* : Un contenu local peut toujours écrire des objets partagés par un domaine tiers sur un disque (domaines autres que le domaine de la barre d'adresse du navigateur actuel), quoiqu'il soit interdit à un domaine tiers d'écrire des objets partagés sur un disque.

## Exemple

La fonction suivante extrait un objet partagé, `my_so`,, et complète les propriétés accessibles en écriture avec les paramètres fournis par l'utilisateur. Enfin `flush()` est appelé pour enregistrer les paramètres et allouer un espace disque minimum de 1 000 octets.

```
this.syncSettingsCore = function(soName:String, override:Boolean,
  settings:Object) {
  var my_so:SharedObject = SharedObject.getLocal(soName, "http://
  www.mydomain.com/app/sys");
  // settings list index
  var i;
  // For each specified value in settings:
  // If override is true, set the persistent setting to the provided value.
  // If override is false, fetch the persistent setting, unless there
  // isn't one, in which case, set it to the provided value.
  for (i in settings) {
  if (override || (my_so.data[i] == null)) {
    my_so.data[i] = settings[i];
  } else {
    settings[i] = my_so.data[i];
  }
  }
  my_so.flush(1000);
};
```

### Voir également

clear (méthode SharedObject.clear), onStatus (gestionnaire SharedObject.onStatus)

## getLocal (méthode SharedObject.getLocal)

```
public static getLocal(name:String, [localPath:String], [secure:Boolean]) :
  SharedObject
```

Renvoie une référence à un objet partagé persistant localement disponible uniquement pour le client actuel. Si l'objet partagé n'existe pas encore, cette méthode en crée un. Il s'agit d'une méthode statique de la classe SharedObject. Pour affecter l'objet à une variable, utilisez une syntaxe du type suivant :

```
var so:SharedObject = SharedObject.getLocal("savedData")
```

*Remarque :* si l'utilisateur a choisi de ne jamais autoriser l'enregistrement local pour ce domaine, l'objet n'est pas enregistré localement, même si une valeur est spécifiée pour `localPath`. L'exception à cette règle est le contenu local. Un contenu local peut toujours écrire des objets partagés par un domaine tiers sur un disque (domaines autres que le domaine de la barre d'adresse du navigateur actuel), quoiqu'il soit interdit à un domaine tiers d'écrire des objets partagés sur un disque.

Pour éviter les collisions de noms, Flash examine l'emplacement du fichier SWF qui crée l'objet partagé. Par exemple, si un fichier SWF à l'adresse www.myCompany.com/apps/stockwatcher.swf crée un objet partagé nommé `portfolio`, cet objet partagé n'entre pas en conflit avec un autre objet nommé`portfolio` qui a été créé par un fichier SWF à l'adresse www.yourCompany.com/photoshoot.swf, étant donné que les fichiers SWF proviennent de répertoires différents.

Même si le paramètre `localPath` est facultatif, vous devez réfléchir à son utilisation, en particulier si les autres fichiers SWF doivent accéder à l'objet partagé. Si les données dans l'objet partagé sont spécifiques à un fichier SWF qui ne sera pas déplacé vers un autre emplacement, l'utilisation de la valeur par défaut a alors du sens. Si les autres fichiers SWF nécessitent un accès à l'objet partagé ou si le fichier SWF qui crée l'objet partagé est déplacé, la valeur de ce paramètre peut alors avoir un effet sur la capacité des fichiers SWF à accéder à l'objet partagé. Par exemple, si vous créez un objet partagé avec `localPath` défini sur la valeur par défaut du chemin complet vers le fichier SWF, aucun autre fichier SWF ne pourra accéder à cet objet partagé. Si vous déplacez par la suite le fichier SWF d'origine vers un autre emplacement, ce fichier SWF ne pourra pas accéder aux données déjà enregistrées dans l'objet partagé.

Vous pouvez réduire la probabilité de limiter involontairement l'accès à un objet partagé à l'aide du paramètre `localpath`. La meilleure option est de définir le paramètre `localPath` sur « / », ce qui rend l'objet partagé disponible pour tous les fichiers SWF du domaine, mais augmente la probabilité de doublons de noms par rapport aux autres objets partagés du domaine. Les options les plus restrictives sont disponibles tant que vous pouvez ajouter au paramètre `localPath` des noms de dossiers qui sont contenus dans le chemin complet vers le fichier SWF. Par exemple, vos options de paramètre `localPath` pour l'objet partagé du `portfolio` créé par le fichier SWF à l'adresse www.myCompany.com/apps/stockwatcher.swf sont : « / » ; « /apps » et « /apps/stockwatcher.swf ». Vous devrez déterminer l'option qui vous offre une souplesse optimale pour votre application.

Lorsque vous recourez à cette méthode, pensez au modèle de sécurité Flash Player :

- Vous ne pouvez pas accéder à des objets partagés en franchissant les limites sandbox.
- Les utilisateurs peuvent limiter l'accès aux objets partagés via la boîte de dialogue des réglages de Flash Player ou via le gestionnaire de réglages. Le maximum d'objets partagés pouvant être créés est réglé par défaut sur 100K de données par domaine. Les utilisateurs administratifs et les utilisateurs peuvent également limiter la capacité à écrire au système de fichiers.

Si vous publiez du contenu SWF à lire sous forme de fichiers locaux (soit des fichiers SWF installés localement soit des projecteurs [EXE]) et si vous devez accéder à un objet partagé spécifique à partir de plusieurs fichiers SWF locaux, vous devez tenir compte du fait que pour les fichiers locaux, deux emplacements différents peuvent être utilisés pour stocker des objets partagés. Le domaine utilisé dépend des autorisations de sécurité accordées au fichier local qui a créé l'objet partagé. Les fichiers locaux peuvent avoir trois niveaux distincts d'autorisation : 1) accès au système de fichiers local uniquement, 2) accès au réseau uniquement ou 3) accès au réseau et au système de fichiers local. Les fichiers locaux pouvant accéder au système de fichiers local (1 ou 3) stockent leurs objets partagés à un emplacement unique. Les fichiers locaux ne pouvant pas accéder au système de fichiers local (2) stockent leurs objets partagés à un autre emplacement. Pour plus d'informations, consultez les sections suivantes :

- Chapitre 17, « Understanding Security » (Comprendre la sécurité), dans *Learning ActionScript 2.0 dans Flash*
- La présentation technique de sécurité de Flash Player 8 disponible à l'adresse : http://www.macromedia.com/go/fp8_security
- La présentation technique des API relatives à la sécurité de Flash Player 8 disponible à l'adresse : http://www.macromedia.com/go/fp8_security_apis

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

## Paramètres

*name*:String - Chaîne indiquant le nom de l'objet. Le nom peut comporter des barres obliques (/); par exemple, work/addresses est un nom admissible. Les espaces ne sont pas autorisés dans un nom d'objet partagé, ainsi que les caractères suivants :

~ % & \ ; : " ' , < > ? #

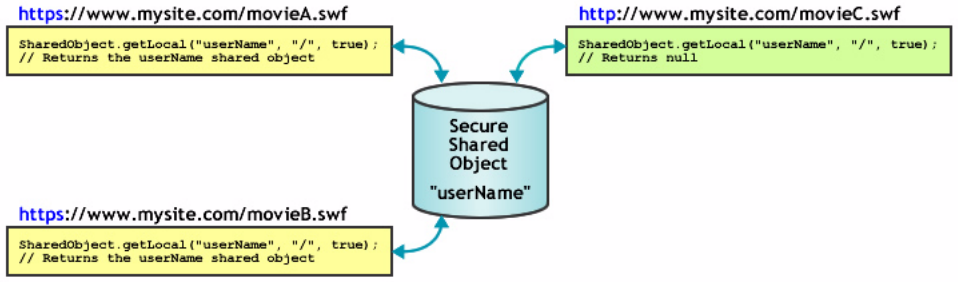
*localPath*:String [facultatif] - Chaîne qui spécifie le chemin complet ou partiel du fichier SWF qui a créé l'objet partagé, ce qui détermine l'endroit où l'objet partagé sera stocké localement. Le chemin complet constitue la valeur par défaut.

*secure*:Boolean [facultatif] - (Flash Player 8 uniquement) Détermine si l'accès à cet objet partagé est limité aux fichiers SWF reçus via une connexion HTTPS. Si votre fichier SWF est reçu via HTTPS :

- Si ce paramètre est défini sur true, Flash Player crée un nouvel objet partagé sécurisé ou obtient une référence pour un objet partagé sécurisé existant. Cet objet sécurisé partagé peut uniquement être lu par des fichiers SWF ou écrit sur des fichiers SWF reçus via des HTTPS appelant SharedObject.getLocal() avec le paramètre secure défini sur true.
- Si ce paramètre est défini sur false, Flash Player crée un nouvel objet partagé ou obtient une référence pour un objet partagé existant. Cet objet partagé peut être lu par des fichiers SWF ou écrit sur des fichiers SWF reçus via des connexions autres que HTTPS.

Si votre fichier SWF est reçu via une connexion autre que HTTPS et que vous essayez de définir ce paramètre sur true, la création d'un nouvel objet partagé (ou l'accès à un objet partagé sécurisé précédemment créé) échoue et null est renvoyé. Quelle que soit la valeur de ce paramètre, les objets partagés créés sont comptabilisés dans la quantité d'espace disque total autorisée pour un domaine. La valeur par défaut est false.

Le diagramme suivant indique comment utiliser le paramètre secure :



## Valeur renvoyée

`SharedObject` - Référence à un objet partagé qui est persistant localement et disponible uniquement pour le client actuel. Si Flash ne peut pas créer ni trouver l'objet partagé (par exemple, si `localPath` a été spécifié mais aucun répertoire de ce type n'existe ou si le paramètre `secure` utilisé est incorrect) cette méthode renvoie `null`.

Cette méthode échoue et renvoie `null` si la création d'un objet partagé persistant et le stockage de contenu Flash en provenance de tiers est interdit (ceci ne s'applique pas au contenu local). Les utilisateurs peuvent interdire les objets partagés persistants de tiers sur le panneau Paramètres globaux d'enregistrement du gestionnaire de paramètres.

## Exemple

L'exemple suivant crée un objet partagé qui stocke du texte tapé dans une occurrence de composant `TextInput`. Le fichier SWF qui en résulte charge le texte enregistré à partir de l'objet partagé dès le début de sa lecture. Lorsque l'utilisateur appuie sur Entrée, le texte contenu dans le champ texte est écrit dans l'objet partagé. Pour utiliser cet exemple, faites glisser un composant `TextInput` sur la scène, puis nommez cette occurrence `myText_ti`. Copiez le code suivant dans le scénario principal (cliquez sur une zone vide de la scène ou appuyez sur Echap pour supprimer le focus du composant) :

```
// Create the shared object and set localpath to server root.
var my_so:SharedObject = SharedObject.getLocal("savedText", "/");
// Load saved text from the shared object into the myText_ti TextInput
  component.
myText_ti.text = my_so.data.myTextSaved;
// Assign an empty string to myText_ti if the shared object is undefined
// to prevent the text input box from displaying "undefined" when
// this script is first run.
if (myText_ti.text == undefined) {
  myText_ti.text = "";
}
// Create a listener object and function for <enter> event
var textListener:Object = new Object();
textListener.enter = function(eventObj:Object) {
  my_so.data.myTextSaved = eventObj.target.text;
  my_so.flush();
};
// Register the listener with the TextInput component instance
myText_ti.addEventListener("enter", textListener);
```

L'exemple suivant enregistre la dernière image entrée par un utilisateur dans un objet local partagé, `kookie`:

```
// Get the kookie
var my_so:SharedObject = SharedObject.getLocal("kookie");
```

```
// Get the user of the kookie and go to the frame number saved for this
  user.
if (my_so.data.user != undefined) {
  this.user = my_so.data.user;
  this.gotoAndStop(my_so.data.frame);
}
```

Le bloc de code suivant est placé sur chaque image de fichier SWF :

```
// On each frame, call the rememberme function to save the frame number.
function rememberme() {
  my_so.data.frame=this._currentframe;
  my_so.data.user="John";
}
```

# getSize (méthode SharedObject.getSize)

```
public getSize() : Number
```

Obtient la taille actuelle de l'objet partagé, en octets.

Flash calcule la taille d'un objet partagé en passant par toutes ses propriétés de données ; plus un objet a de propriétés de données, plus l'estimation de sa taille prend du temps. L'estimation de la taille de l'objet peut monopoliser beaucoup de temps de traitement. Il est donc recommandé d'éviter d'utiliser cette méthode à moins de devoir combler un besoin spécifique.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

## Valeur renvoyée

`Number` - Valeur numérique spécifiant la taille de l'objet partagé, en octets.

## Exemple

L'exemple suivant obtient la taille de l'objet partagé `my_so` :

```
var items_array:Array = new Array(101, 346, 483);
var currentUserIsAdmin:Boolean = true;
var currentUserName:String = "Ramona";

var my_so:SharedObject = SharedObject.getLocal("superfoo");
my_so.data.itemNumbers = items_array;
my_so.data.adminPrivileges = currentUserIsAdmin;
my_so.data.userName = currentUserName;

var soSize:Number = my_so.getSize();
trace(soSize);
```

# onStatus (gestionnaire SharedObject.onStatus)

onStatus = function(infoObject:Object) {}

Appelé dès qu'une erreur, un avertissement ou une note d'informations est publié pour un objet partagé. Si vous souhaitez répondre à ce gestionnaire d'événements, vous devez créer une fonction pour traiter l'objet d'information généré par l'objet partagé.

L'objet d'informations a une propriété de code contenant une chaîne qui décrit le résultat du gestionnaire `onStatus` et une propriété de niveau `level` contenant une chaîne qui est soit `"Status"` soit `"Error"`.

En plus de ce gestionnaire `onStatus`, Flash fournit également une super fonction appelée `System.onStatus`. Si `onStatus` est appelé pour un objet particulier et qu'aucune fonction n'est affectée pour y répondre, Flash exécute une fonction affectée à `System.onStatus`, si elle existe.

Les événements suivants vous informent lorsque certaines activités `SharedObject` ont lieu :

| Propriété du code | Propriété de niveau | Signification |
|---|---|---|
| `SharedObject.Flush.Failed` | Erreur | La commande `SharedObject.flush()` qui a renvoyé `"pending"` (en attente) a échoué (l'utilisateur n'avait pas alloué d'espace disque supplémentaire pour l'objet partagé lorsque Flash Player a affiché la boîte de dialogue Paramètres d'enregistrement local). |
| `SharedObject.Flush.Success` | État | La commande `SharedObject.flush()` qui a renvoyé `"pending"` (en attente) s'est terminée (l'utilisateur a alloué de l'espace disque supplémentaire à l'objet partagé). |

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

## Paramètres

*infoObject*:Object - Paramètre défini en fonction du message d'état.

## Exemple

L'exemple suivant affiche différents messages selon que l'utilisateur accepte ou refuse l'écriture de l'occurrence d'objet SharedObject sur le disque.

```
var message_str:String;
this.createTextField("message_txt", this.getNextHighestDepth(), 0, 0, 300,
  22);
message_txt.html = true;
this.createTextField("status_txt", this.getNextHighestDepth(), 10, 30, 300,
  100);
status_txt.multiline = true;
status_txt.html = true;

var items_array:Array = new Array(101, 346, 483);
var currentUserIsAdmin:Boolean = true;
var currentUserName:String = "Ramona";
var my_so:SharedObject = SharedObject.getLocal("superfoo");
my_so.data.itemNumbers = items_array;
my_so.data.adminPrivileges = currentUserIsAdmin;
my_so.data.userName = currentUserName;

my_so.onStatus = function(infoObject:Object) {
  status_txt.htmlText = "<textformat tabStops='[50]'>";
  for (var i in infoObject) {
    status_txt.htmlText += "<b>"+i+"</b>"+"\t"+infoObject[i];
  }
  status_txt.htmlText += "</textformat>";
};

var flushResult = my_so.flush(1000001);
switch (flushResult) {
case 'pending' :
  message_str = "flush is pending, waiting on user interaction.";
  break;
case true :
  message_str = "flush was successful. Requested storage space approved.";
  break;
case false :
  message_str = "flush failed. User denied request for additional
  storage.";
  break;
}
message_txt.htmlText = "<a href=\"asfunction:System.showSettings,1\
"><u>"+message_str+"</u></a>";
```

La méthode `MovieClip.getNextHighestDepth()` utilisée dans cet exemple requiert Flash Player 7 ou version ultérieure. Si votre fichier SWF comporte un composant de la version 2, utilisez la classe DepthManager des composants de la version 2 au lieu de la méthode `MovieClip.getNextHighestDepth()`.

### Voir également

`getLocal (méthode SharedObject.getLocal)`, `onStatus (gestionnaire System.onStatus)`

# Sound

```
Object
  |
  +-Sound
```

`public class` ***Sound***
`extends Object`

La classe Sound vous laisse contrôler le son d'une animation. Vous pouvez ajouter des sons à un clip de la bibliothèque en cours de lecture et contrôler ces sons. Si vous ne spécifiez pas une cible lorsque vous créez un objet new Sound, vous pouvez utiliser les méthodes pour contrôler le son pour l'animation entière.

Vous devez utiliser le constructeur `new Sound` pour créer un objet Sound avant d'appeler les méthodes de la classe Sound.

**Disponibilité :** ActionScript 1.0 ; Flash Player 5

### Résumé des propriétés

| Modificateurs | Propriété | Description |
|---|---|---|
| | `duration:Number` [lecture seule] | Durée d'un son, en millisecondes. |
| | `id3:Object` [lecture seule] | Donne accès aux métadonnées faisant partie d'un fichier MP3. |
| | `position:Number` [lecture seule] | La durée de diffusion du son exprimée en millisecondes. |

*Propriétés héritées de la classe Object*

`constructeur (propriété Object.constructor)`, `__proto__ (Object.__proto__ property)`, `prototype (Object.prototype, propriété)`, `__resolve (Object.__resolve, propriété)`

## Résumé des événements

| Evénement | Description |
|---|---|
| `onID3 = function() {}` | Appelé dès que de nouvelles données ID3 sont disponibles pour un fichier MP3 que vous chargez à l'aide de `Sound.attachSound()` ou `Sound.loadSound()`. |
| `onLoad = function(success :Boolean) {}` | Appelé automatiquement lorsqu'un son est en cours de chargement. |
| `onSoundComplete = function() {}` | Appelé automatiquement lorsque la lecture d'un son est finie. |

## Résumé des constructeurs

| Signature | Description |
|---|---|
| `Sound([target:Object ])` | Crée un nouvel objet Sound pour un clip spécifique. |

## Résumé de la méthode

| Modificateurs | Signature | Description |
|---|---|---|
| | `attachSound(id:Strin g) : Void` | Associe le son spécifié par le paramètre `id` à l'objet Sound spécifié. |
| | `getBytesLoaded() : Number` | Renvoie le nombre d'octets chargés (transmis en continu) pour l'objet Sound spécifié. |
| | `getBytesTotal() : Number` | Renvoie la taille, en octets, de l'objet Sound spécifié. |
| | `getPan() : Number` | Renvoie le niveau panoramique défini dans le dernier appel `setPan()` en tant qu'entier compris entre -100 (gauche) et +100 (droite). |
| | `getTransform() : Object` | Renvoie les informations de transformation du son pour l'objet Sound spécifié défini avec le dernier appel `Sound.setTransform()`. |
| | `getVolume() : Number` | Renvoie le niveau de volume sonore en tant qu'entier compris entre 0 et 100, où 0 correspond à un son coupé et 100 à un volume maximum. |
| | `loadSound(url:String , isStreaming:Boolean) : Void` | Charge un fichier MP3 dans un objet Sound. |

| Modificateurs | Signature | Description |
|---|---|---|
| | `setPan(value:Number) : Void` | Détermine le mode de lecture du son dans les canaux gauche et droit (haut-parleurs). |
| | `setTransform(transformObject:Object) : Void` | Définit les informations de transformation du son (ou balance), pour un objet Sound. |
| | `setVolume(value:Number) : Void` | Définit le volume pour l'objet Sound. |
| | `start([secondOffset:Number], [loops:Number]) : Void` | Commence à lire le dernier son associé depuis le début si aucun paramètre n'est spécifié, ou en commençant à l'endroit spécifié par le paramètre `secondOffset`. |
| | `stop([linkageID:String]) : Void` | Arrête tous les sons en cours de lecture si aucun paramètre n'est spécifié, ou uniquement le son spécifié dans le paramètre `idName`. |

*Méthodes héritées de la classe Object*

`addProperty (méthode Object.addProperty)`, `hasOwnProperty (méthode Object.hasOwnProperty)`, `isPropertyEnumerable (méthode Object.isPropertyEnumerable)`, `isPrototypeOf (méthode Object.isPrototypeOf)`, `registerClass (méthode Object.registerClass)`, `toString (méthode Object.toString)`, `unwatch (méthode Object.unwatch)`, `valueOf (méthode Object.valueOf)`, `watch (méthode Object.watch)`

# attachSound (méthode Sound.attachSound)

`public attachSound(id:String) : Void`

Associe le son spécifié par le paramètre `id` à l'objet Sound spécifié. Le son doit être dans la bibliothèque du fichier SWF actuel et spécifié pour l'export dans la boîte de dialogue Propriétés de liaison. Vous devez appeler `Sound.start()` pour commencer la lecture du son.

Pour vous assurer que le son peut être contrôlé depuis une quelconque scène dans le fichier SWF, placez le son sur le scénario principal du fichier SWF.

**Disponibilité :** ActionScript 1.0 ; Flash Player 5

## Paramètres

*`id`*`:String` - Identifiant d'un son exporté dans la bibliothèque. L'identifiant est situé dans la boîte de dialogue Propriétés de liaison.

### Exemple

L'exemple suivant associe le son `logoff_id` à `my_sound`. L'un des sons de la bibliothèque comporte l'identifiant de liaison `logoff_id`.

```
var my_sound:Sound = new Sound();
my_sound.attachSound("logoff_id");
my_sound.start();
```

## duration (propriété Sound.duration)

```
public duration : Number [lecture seule]
```

Durée d'un son, en millisecondes.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

### Exemple

L'exemple suivant charge un son et affiche la durée du fichier audio dans le panneau de sortie. Ajoutez à votre fichier FLA ou AS le code ActionScript suivant.

```
var my_sound:Sound = new Sound();
my_sound.onLoad = function(success:Boolean) {
  var totalSeconds:Number = this.duration/1000;
  trace(this.duration+" ms ("+Math.round(totalSeconds)+" seconds)");
  var minutes:Number = Math.floor(totalSeconds/60);
  var seconds = Math.floor(totalSeconds)%60;
  if (seconds<10) {
  seconds = "0"+seconds;
  }
  trace(minutes+":"+seconds);
};
my_sound.loadSound("song1.mp3", true);
```

L'exemple suivant charge plusieurs morceaux dans un fichier SWF. Une barre de progression, créée avec l'API de dessin, indique la progression du chargement. Lorsque la musique commence, et termine son chargement, des informations s'affichent dans le panneau de sortie. Ajoutez à votre fichier FLA ou AS le code ActionScript suivant.

```
var pb_height:Number = 10;
var pb_width:Number = 100;
var pb:MovieClip = this.createEmptyMovieClip("progressBar_mc",
  this.getNextHighestDepth());
pb.createEmptyMovieClip("bar_mc", pb.getNextHighestDepth());
pb.createEmptyMovieClip("vBar_mc", pb.getNextHighestDepth());
pb.createEmptyMovieClip("stroke_mc", pb.getNextHighestDepth());
pb.createTextField("pos_txt", pb.getNextHighestDepth(), 0, pb_height,
  pb_width, 22);

pb._x = 100;
```

```
pb._y = 100;

with (pb.bar_mc) {
  beginFill(0x00FF00);
  moveTo(0, 0);
  lineTo(pb_width, 0);
  lineTo(pb_width, pb_height);
  lineTo(0, pb_height);
  lineTo(0, 0);
  endFill();
  _xscale = 0;
}
with (pb.vBar_mc) {
  lineStyle(1, 0x000000);
  moveTo(0, 0);
  lineTo(0, pb_height);
}
with (pb.stroke_mc) {
  lineStyle(3, 0x000000);
  moveTo(0, 0);
  lineTo(pb_width, 0);
  lineTo(pb_width, pb_height);
  lineTo(0, pb_height);
  lineTo(0, 0);
}

var my_interval:Number;
var my_sound:Sound = new Sound();
my_sound.onLoad = function(success:Boolean) {
  if (success) {
  trace("sound loaded");
  }
};
my_sound.onSoundComplete = function() {
  clearInterval(my_interval);
  trace("Cleared interval");
}
my_sound.loadSound("song3.mp3", true);
my_interval = setInterval(updateProgressBar, 100, my_sound);

function updateProgressBar(the_sound:Sound):Void {
  var pos:Number = Math.round(the_sound.position/the_sound.duration 100);
  pb.bar_mc._xscale = pos;
  pb.vBar_mc._x = pb.bar_mc._width;
  pb.pos_txt.text = pos+"%";
}
```

La méthode `MovieClip.getNextHighestDepth()` utilisée dans cet exemple requiert Flash Player 7 ou version ultérieure. Si votre fichier SWF comporte un composant de la version 2, utilisez la classe DepthManager des composants de la version 2 au lieu de la méthode `MovieClip.getNextHighestDepth()`.

### Voir également

`position (Sound.position, propriété)`

## getBytesLoaded (méthode Sound.getBytesLoaded)

```
public getBytesLoaded() : Number
```

Renvoie le nombre d'octets chargés (transmis en continu) pour l'objet Sound spécifié. Vous pouvez comparer la valeur de `getBytesLoaded()` à la valeur de `getBytesTotal()` pour déterminer le pourcentage d'un son qui a été chargé.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

### Valeur renvoyée

`Number` - Entier indiquant le nombre d'octets chargés.

### Exemple

L'exemple suivant crée deux champs texte qui affichent le nombre d'octets chargés et le nombre total d'octets du fichier son qui se charge dans le fichier SWF. Un champ texte affiche également un message lorsque le fichier a terminé son chargement. Ajoutez le code ActionScript suivant à votre fichier FLA ou AS :

```
this.createTextField("message_txt", this.getNextHighestDepth(),
  10,10,300,22)
this.createTextField("status_txt", this.getNextHighestDepth(), 10, 50, 300,
  40);
status_txt.autoSize = true;
status_txt.multiline = true;
status_txt.border = false;

var my_sound:Sound = new Sound();
my_sound.onLoad = function(success:Boolean) {
  if (success) {
  this.start();
  message_txt.text = "Finished loading";
  }
};
my_sound.onSoundComplete = function() {
  message_txt.text = "Clearing interval";
  clearInterval(my_interval);
```

```
};
my_sound.loadSound("song2.mp3", true);
var my_interval:Number;
my_interval = setInterval(checkProgress, 100, my_sound);
function checkProgress(the_sound:Sound):Void {
  var pct:Number = Math.round(the_sound.getBytesLoaded()/
  the_sound.getBytesTotal() 100);
  var pos:Number = Math.round(the_sound.position/the_sound.duration 100);
  status_txt.text = the_sound.getBytesLoaded()+" of
  "+the_sound.getBytesTotal()+" bytes ("+pct+"%)"+newline;
  status_txt.text += the_sound.position+" of "+the_sound.duration+"
  milliseconds ("+pos+"%)"+newline;
}
```

La méthode `MovieClip.getNextHighestDepth()` utilisée dans cet exemple requiert Flash Player 7 ou version ultérieure. Si votre fichier SWF comporte un composant de la version 2, utilisez la classe DepthManager des composants de la version 2 au lieu de la méthode `MovieClip.getNextHighestDepth()`.

### Voir également

`getBytesTotal (méthode Sound.getBytesTotal)`

## getBytesTotal (méthode Sound.getBytesTotal)

```
public getBytesTotal() : Number
```

Renvoie la taille, en octets, de l'objet Sound spécifié.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

### Valeur renvoyée

`Number` - Entier indiquant la taille totale, en octets, de l'objet Sound spécifié.

### Exemple

Consultez `Sound.getBytesLoaded()` pour obtenir un exemple d'utilisation de cette méthode.

### Voir également

`getBytesLoaded (méthode Sound.getBytesLoaded)`

# getPan (méthode Sound.getPan)

```
public getPan() : Number
```

Renvoie le niveau panoramique défini dans le dernier appel `setPan()` en tant qu'entier compris entre -100 (gauche) et +100 (droite). (0 définit de la même manière les canaux gauche et droit.) Le réglage panoramique contrôle la balance gauche-droite des sons actuels et futurs d'un fichier SWF.

Cette méthode s'ajoute à `setVolume()` ou `setTransform()`.

**Disponibilité :** ActionScript 1.0 ; Flash Player 5

## Valeur renvoyée

`Number` - Entier.

## Exemple

L'exemple suivant crée une barre de défilement avec l'API de dessin. Lorsque l'utilisateur fait glisser la barre, le niveau panoramique du son chargé change. Le niveau panoramique en vigueur s'affiche dans un champ texte créé de façon dynamique. Ajoutez le code ActionScript suivant à votre fichier FLA ou AS :

```
var bar_width:Number = 200;
this.createEmptyMovieClip("bar_mc", this.getNextHighestDepth());
with (bar_mc) {
  lineStyle(4, 0x000000);
  moveTo(0, 0);
  lineTo(bar_width+4, 0);
  lineStyle(0, 0x000000);
  moveTo((bar_width/2)+2, -8);
  lineTo((bar_width/2)+2, 8);
}
bar_mc._x = 100;
bar_mc._y = 100;

this.createEmptyMovieClip("knob_mc", this.getNextHighestDepth());
with (knob_mc) {
  lineStyle(0, 0x000000);
  beginFill(0xCCCCCC);
  moveTo(0, 0);
  lineTo(4, 0);
  lineTo(4, 10);
  lineTo(0, 10);
  lineTo(0, 0);
  endFill();
}
knob_mc._x = bar_mc._x+(bar_width/2);
knob_mc._y = bar_mc._y-(knob_mc._height/2);
```

```
knob_mc.left = knob_mc._x-(bar_width/2);
knob_mc.right = knob_mc._x+(bar_width/2);
knob_mc.top = knob_mc._y;
knob_mc.bottom = knob_mc._y;

knob_mc.onPress = function() {
  this.startDrag(false, this.left, this.top, this.right, this.bottom);
};
knob_mc.onRelease = function() {
  this.stopDrag();
  var multiplier:Number = 100/(this.right-this.left) 2;
  var pan:Number = (this._x-this.left-(bar_width/2)) multiplier;
  my_sound.setPan(pan);
  pan_txt.text = my_sound.getPan();
};

var my_sound:Sound = new Sound();
my_sound.loadSound("song2.mp3", true);
this.createTextField("pan_txt", this.getNextHighestDepth(), knob_mc._x,
  knob_mc._y+knob_mc._height, 20, 22);
pan_txt.selectable = false;
pan_txt.autoSize = "center";
pan_txt.text = my_sound.getPan();
```

La méthode `MovieClip.getNextHighestDepth()` utilisée dans cet exemple requiert Flash Player 7 ou version ultérieure. Si votre fichier SWF comporte un composant de la version 2, utilisez la classe DepthManager des composants de la version 2 au lieu de la méthode `MovieClip.getNextHighestDepth()`.

### Voir également

`setPan (méthode Sound.setPan)`

## getTransform (méthode Sound.getTransform)

```
public getTransform() : Object
```

Renvoie les informations de transformation du son pour l'objet Sound spécifié défini avec le dernier appel `Sound.setTransform()`.

**Disponibilité :** ActionScript 1.0 ; Flash Player 5

### Valeur renvoyée

`Object` - Objet avec des propriétés qui contiennent les valeurs de pourcentage de canaux pour l'objet Sound spécifié.

## Exemple

L'exemple suivant associe quatre clips provenant d'un symbole de la bibliothèque (identifiant de liaison : `knob_id`) qui sont utilisés en tant que glissières (ou boutons) pour contrôler le fichier son qui se charge dans le fichier SWF. Ces glissières contrôlent l'objet de transformation, ou balance, du fichier son. Pour plus d'informations, consultez la section `Sound.setTransform()`. Ajoutez le code ActionScript suivant à votre fichier FLA ou AS :

```
var my_sound:Sound = new Sound();
my_sound.loadSound("song1.mp3", true);
var transform_obj:Object = my_sound.getTransform();

this.createEmptyMovieClip("transform_mc", this.getNextHighestDepth());
transform_mc.createTextField("transform_txt",
  transform_mc.getNextHighestDepth, 0, 8, 120, 22);
transform_mc.transform_txt.html = true;

var knob_ll:MovieClip = transform_mc.attachMovie("knob_id", "ll_mc",
  transform_mc.getNextHighestDepth(), {_x:0, _y:30});
var knob_lr:MovieClip = transform_mc.attachMovie("knob_id", "lr_mc",
  transform_mc.getNextHighestDepth(), {_x:30, _y:30});
var knob_rl:MovieClip = transform_mc.attachMovie("knob_id", "rl_mc",
  transform_mc.getNextHighestDepth(), {_x:60, _y:30});
var knob_rr:MovieClip = transform_mc.attachMovie("knob_id", "rr_mc",
  transform_mc.getNextHighestDepth(), {_x:90, _y:30});

knob_ll.top = knob_ll._y;
knob_ll.bottom = knob_ll._y+100;
knob_ll.left = knob_ll._x;
knob_ll.right = knob_ll._x;
knob_ll._y = knob_ll._y+(100-transform_obj['ll']);
knob_ll.onPress = pressKnob;
knob_ll.onRelease = releaseKnob;
knob_ll.onReleaseOutside = releaseKnob;

knob_lr.top = knob_lr._y;
knob_lr.bottom = knob_lr._y+100;
knob_lr.left = knob_lr._x;
knob_lr.right = knob_lr._x;
knob_lr._y = knob_lr._y+(100-transform_obj['lr']);
knob_lr.onPress = pressKnob;
knob_lr.onRelease = releaseKnob;
knob_lr.onReleaseOutside = releaseKnob;

knob_rl.top = knob_rl._y;
knob_rl.bottom = knob_rl._y+100;
knob_rl.left = knob_rl._x;
knob_rl.right = knob_rl._x;
knob_rl._y = knob_rl._y+(100-transform_obj['rl']);
knob_rl.onPress = pressKnob;
```

```
knob_rl.onRelease = releaseKnob;
knob_rl.onReleaseOutside = releaseKnob;

knob_rr.top = knob_rr._y;
knob_rr.bottom = knob_rr._y+100;
knob_rr.left = knob_rr._x;
knob_rr.right = knob_rr._x;
knob_rr._y = knob_rr._y+(100-transform_obj['rr']);
knob_rr.onPress = pressKnob;
knob_rr.onRelease = releaseKnob;

knob_rr.onReleaseOutside = releaseKnob;

updateTransformTxt();

function pressKnob() {
  this.startDrag(false, this.left, this.top, this.right, this.bottom);
}
function releaseKnob() {
  this.stopDrag();
  updateTransformTxt();
}
function updateTransformTxt() {
  var ll_num:Number = 30+100-knob_ll._y;
  var lr_num:Number = 30+100-knob_lr._y;
  var rl_num:Number = 30+100-knob_rl._y;
  var rr_num:Number = 30+100-knob_rr._y;
  my_sound.setTransform({ll:ll_num, lr:lr_num, rl:rl_num, rr:rr_num});
  transform_mc.transform_txt.htmlText = "<textformat
  tabStops='[0,30,60,90]'>";
  transform_mc.transform_txt.htmlText +=
  ll_num+"\t"+lr_num+"\t"+rl_num+"\t"+rr_num;
  transform_mc.transform_txt.htmlText += "</textformat>";
}
```

La méthode `MovieClip.getNextHighestDepth()` utilisée dans cet exemple requiert Flash Player 7 ou version ultérieure. Si votre fichier SWF comporte un composant de la version 2, utilisez la classe DepthManager des composants de la version 2 au lieu de la méthode `MovieClip.getNextHighestDepth()`.

## Voir également

`setTransform (méthode Sound.setTransform)`

# getVolume (méthode Sound.getVolume)

```
public getVolume() : Number
```

Renvoie le niveau de volume sonore en tant qu'entier compris entre 0 et 100, où 0 correspond à un son coupé et 100 à un volume maximum. Le paramètre par défaut est 100.

**Disponibilité :** ActionScript 1.0 ; Flash Player 5

## Valeur renvoyée

`Number` - Entier.

## Exemple

L'exemple suivant crée une glissière avec l'API de dessin et un clip est créé lors de l'exécution. Un champ texte créé de façon dynamique affiche le niveau de volume actuel du son lu dans le fichier SWF. Ajoutez le code ActionScript suivant à votre fichier AS ou FLA :

```
var my_sound:Sound = new Sound();
my_sound.loadSound("song3.mp3", true);

this.createEmptyMovieClip("knob_mc", this.getNextHighestDepth());

knob_mc.left = knob_mc._x;
knob_mc.right = knob_mc.left+100;
knob_mc.top = knob_mc._y;
knob_mc.bottom = knob_mc._y;

knob_mc._x = my_sound.getVolume();

with (knob_mc) {
  lineStyle(0, 0x000000);
  beginFill(0xCCCCCC);
  moveTo(0, 0);
  lineTo(4, 0);
  lineTo(4, 18);
  lineTo(0, 18);
  lineTo(0, 0);
  endFill();
}

knob_mc.createTextField("volume_txt", knob_mc.getNextHighestDepth(),
  knob_mc._width+4, 0, 30, 22);
knob_mc.volume_txt.text = my_sound.getVolume();

knob_mc.onPress = function() {
  this.startDrag(false, this.left, this.top, this.right, this.bottom);
  this.isDragging = true;
};
knob_mc.onMouseMove = function() {
```

```
  if (this.isDragging) {
  this.volume_txt.text = this._x;
  }
}
knob_mc.onRelease = function() {
  this.stopDrag();
  this.isDragging = false;
  my_sound.setVolume(this._x);

};
```

La méthode `MovieClip.getNextHighestDepth()` utilisée dans cet exemple requiert Flash Player 7 ou version ultérieure. Si votre fichier SWF comporte un composant de la version 2, utilisez la classe DepthManager des composants de la version 2 au lieu de la méthode `MovieClip.getNextHighestDepth()`.

### Voir également

`setVolume (méthode Sound.setVolume)`

## id3 (Sound.id3, propriété)

`public id3 : Objet [lecture seule]`

Donne accès aux métadonnées faisant partie d'un fichier MP3.

Les fichiers son MP3 peuvent contenir des balises ID3, qui fournissent des métadonnées sur le fichier. Si un son MP3 que vous chargez à l'aide de `Sound.attachSound()` ou `Sound.loadSound()` contient des balises ID3, vous pouvez interroger ces propriétés. Seules les balises ID3 qui utilisent le jeu de caractères UTF-8 sont prises en charge.

Flash Player 6 (6.0.40.0) et les versions ultérieures utilisent la propriété `Sound.id3` pour prendre en charge les balises ID3 1.0 et ID3 1.1. Flash Player 7 ajoute une prise en charge des balises ID3 2.0, en particulier les balises 2.3 et 2.4. Le tableau suivant énumère les balises ID3 2.0 classiques et le type de contenu représenté par les balises ; vous les interrogez au format *`my_sound`*`.id3.COMM`, *`my_sound`*`.id3.TIME`, etc. Les fichiers MP3 peuvent contenir des balises différentes de celles indiquées dans ce tableau ;`Sound.id3` donne également accès à ces balises.

| Propriété | Description |
|---|---|
| TFLT | Type de fichier |
| TIME | Durée |
| TIT1 | Description du groupe de contenus |
| TIT2 | Titre/nom du morceau/description du contenu |

| Propriété | Description |
|---|---|
| TIT3 | Sous-titre/description plus précise |
| TKEY | Touche initiale |
| TLAN | Langues |
| TLEN | Durée |
| TMED | Type de média |
| TOAL | Album/film/titre du spectacle d'origine |
| TOFN | Nom de fichier d'origine |
| TOLY | Paroliers/auteurs d'origine |
| TOPE | Artiste d'origine/musiciens |
| TORY | Année de parution d'origine |
| TOWN | Propriétaire du fichier/détenteur de licence |
| TPE1 | Artiste principal/solistes |
| TPE2 | Groupe/orchestre/accompagnement |
| TPE3 | Chef d'orchestre/description plus détaillée des musiciens |
| TPE4 | Interprété, remixé ou modifié par |
| TPOS | Elément d'un ensemble |
| TPUB | Editeur |
| TRCK | Numéro de piste/position dans l'ensemble |
| TRDA | Dates d'enregistrement |
| TRSN | Nom de station de la radio Internet |
| TRSO | Propriétaire de la station de la radio Internet |
| TSIZ | Taille |
| TSRC | ISRC (international standard recording code - code standard international d'enregistrement) |
| TSSE | Logiciel/matériel et paramètres utilisés pour le codage |
| TYER | Année |
| WXXX | Structure de lien URL |

Flash Player 6 prenait en charge plusieurs balises ID31.0. Si ces balises ne sont pas dans le fichier MP3, mais si les balises ID3 2.0 correspondantes sont dans le fichier, les balises ID3 2.0 sont copiées dans les propriétés ID3 1.0, comme il est illustré dans le tableau suivant. Ce processus offre une compatibilité ascendante avec les scripts que vous pouvez avoir déjà écrits et qui lisent les propriétés ID3 1.0.

| Balise ID3 2.0 | Propriété ID3 1.0 correspondante |
|---|---|
| COMM | Sound.id3.comment |
| TALB | Sound.id3.album |
| TCON | Sound.id3.genre |
| TIT2 | Sound.id3.songname |
| TPE1 | Sound.id3.artist |
| TRCK | Sound.id3.track |
| TYER | Sound.id3.year |

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

### Exemple

L'exemple suivant permet de suivre les propriétés ID3 de song.mp3 et les affiche dans le panneau de sortie :

```
var my_sound:Sound = new Sound();
my_sound.onID3 = function(){
  for( var prop in my_sound.id3 ){
  trace( prop + " : "+ my_sound.id3[prop] );
  }
}
my_sound.loadSound("song.mp3", false);
```

### Voir également

`attachSound (méthode Sound.attachSound)`, `loadSound (méthode Sound.loadSound)`

## loadSound (méthode Sound.loadSound)

```
public loadSound(url:String, isStreaming:Boolean) : Void
```

Charge un fichier MP3 dans un objet Sound. Vous pouvez utiliser le paramètre `isStreaming` pour indiquer si le son est un événement ou un son en flux continu.

Les sons d'événement sont totalement chargés avant d'être lus. Ils sont gérés par la classe ActionScript Sound et répondent à toutes les méthodes et propriétés de cette classe.

Les sons en flux continu sont lus lors de leur téléchargement. La lecture commence lorsqu'un nombre suffisant de données a été reçu pour lancer le décompresseur.

Tous les MP3 (événement ou flux continu) chargés avec cette méthode sont enregistrés dans la mémoire cache du fichier du navigateur du système utilisateur.

Lorsque vous recourez à cette méthode, pensez au modèle de sécurité Flash Player.

Pour Flash Player 8 :

- `Sound.loadSound()` n'est pas autorisé si le fichier SWF procédant à l'appel se trouve dans le sandbox local du système de fichiers et que le son se trouve dans un sandbox du réseau.
- Un accès à partir du sandbox de confiance locale ou du sandbox local avec réseau requiert l'autorisation du site web via un fichier de régulation interdomaine.

Pour Flash Player 7 et versions suivantes :

- Les sites web peuvent autoriser l'accès à une ressource provenant de demandeurs de différents domaines via un fichier de régulation interdomaine.

Pour plus d'informations, consultez les sections suivantes :

- Chapitre 17, « Understanding Security » (Comprendre la sécurité), dans *Learning ActionScript 2.0 dans Flash*
- La présentation technique de sécurité de Flash Player 8 disponible à l'adresse : http://www.macromedia.com/go/fp8_security
- La présentation technique des API relatives à la sécurité de Flash Player 8 disponible à l'adresse : http://www.macromedia.com/go/fp8_security_apis

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

## Paramètres

*url*:`String` - Emplacement sur un serveur d'un fichier son MP3.

*isStreaming*:`Boolean` - Valeur booléenne qui indique si le son est en diffusion continue (`true`) ou un événement audio (`false`).

## Exemple

L'exemple suivant charge un événement audio, qui ne peut pas être lu avant la fin de son chargement :

```
var my_sound:Sound = new Sound();
my_sound.loadSound("song1.mp3", false);
```

L'exemple suivant charge un son à diffusion en continu :

```
var my_sound:Sound = new Sound();
my_sound.loadSound("song1.mp3", true);
```

### Voir également

onLoad (gestionnaire Sound.onLoad)

## onID3 (gestionnaire Sound.onID3)

onID3 = function() {}

Appelé dès que de nouvelles données ID3 sont disponibles pour un fichier MP3 que vous chargez à l'aide de `Sound.attachSound()` ou `Sound.loadSound()`. Ce gestionnaire donne accès aux données ID3 sans interrogation. Si les balises ID3 1.0 et ID3 2.0 existent dans un fichier, ce gestionnaire est appelé deux fois.

**Disponibilité :** ActionScript 1.0 ; Flash Player 7

### Exemple

L'exemple suivant affiche les propriétés ID3 de `song1.mp3` dans une occurrence du composant DataGrid. Ajoutez à votre document un composant DataGrid sous le nom d'occurrence `id3_dg` et ajoutez le code ActionScript suivant à votre fichier FLA ou AS :

```
import mx.controls.gridclasses.DataGridColumn;
var id3_dg:mx.controls.DataGrid;
id3_dg.move(0, 0);
id3_dg.setSize(Stage.width, Stage.height);
var property_dgc:DataGridColumn = id3_dg.addColumn(new
  DataGridColumn("property"));
property_dgc.width = 100;
property_dgc.headerText = "ID3 Property";
var value_dgc:DataGridColumn = id3_dg.addColumn(new
  DataGridColumn("value"));
value_dgc.width = id3_dg._width-property_dgc.width;
value_dgc.headerText = "ID3 Value";

var my_sound:Sound = new Sound();
my_sound.onID3 = function() {
trace("onID3 called at "+getTimer()+" ms.");
for (var prop in this.id3) {
id3_dg.addItem({property:prop, value:this.id3[prop]});
}
};
my_sound.loadSound("song1.mp3", true);
```

### Voir également

attachSound (méthode Sound.attachSound), id3 (Sound.id3, propriété), loadSound (méthode Sound.loadSound)

# onLoad (gestionnaire Sound.onLoad)

## onLoad = function(success:Boolean) {}

Appelé automatiquement lorsqu'un son est en cours de chargement. Vous devez créer une fonction qui s'exécute lorsque ce gestionnaire est appelé. Vous pouvez utiliser une fonction anonyme ou une fonction nommée (vous trouverez un exemple de chacune de ces fonctions dans `Sound.onSoundComplete`). Il est recommandé de définir ce gestionnaire avant d'appeler `mySound.loadSound()`.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

### Paramètres

*`success`*`:Boolean` - Valeur booléenne `true` si `my_sound` a été chargé avec succès, et `false` dans tous les autres cas.

### Exemple

L'exemple suivant crée un nouvel objet Sound et charge un son. Le chargement du son est traité par le gestionnaire `onLoad`, qui permet également de diffuser le morceau à l'issue de son chargement. Créez un nouveau fichier FLA et ajoutez le code ActionScript suivant à votre fichier FLA ou AS. Pour que cet exemple fonctionne, vous devez disposer d'un fichier MP3 appelé `song1.mp3` et situé dans le même répertoire que votre fichier FLA ou AS.

```
this.createTextField("status_txt", this.getNextHighestDepth(), 0,0,100,22);

// create a new Sound object
var my_sound:Sound = new Sound();
// if the sound loads, play it; if not, trace failure loading
my_sound.onLoad = function(success:Boolean) {
if (success) {
my_sound.start();
status_txt.text = "Sound loaded";
} else {
status_txt.text = "Sound failed";
}
};
// load the sound
my_sound.loadSound("song1.mp3", true);
```

La méthode `MovieClip.getNextHighestDepth()` utilisée dans cet exemple requiert Flash Player 7 ou version ultérieure. Si votre fichier SWF comporte un composant de la version 2, utilisez la classe DepthManager des composants de la version 2 au lieu de la méthode `MovieClip.getNextHighestDepth()`.

### Voir également

loadSound (méthode Sound.loadSound)

## onSoundComplete (gestionnaire Sound.onSoundComplete)

onSoundComplete = function() {}

Appelé automatiquement lorsque la lecture d'un son est finie. Vous pouvez utiliser ce gestionnaire pour déclencher les événements dans un fichier SWF lorsqu'un son s'arrête.

Vous devez créer une fonction qui s'exécute lorsque ce gestionnaire est appelé. Vous pouvez utiliser une fonction anonyme ou une fonction nommée.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

### Exemple

Usage 1 : L'exemple suivant utilise une fonction anonyme :

```
var my_sound:Sound = new Sound();
my_sound.attachSound("mySoundID");
my_sound.onSoundComplete = function() {
trace("mySoundID completed");
};
my_sound.start();
```

Usage 2 : L'exemple suivant utilise une fonction nommée :

```
function callback1() {
trace("mySoundID completed");
}
var my_sound:Sound = new Sound();
my_sound.attachSound("mySoundID");
my_sound.onSoundComplete = callback1;
my_sound.start();
```

### Voir également

onLoad (gestionnaire Sound.onLoad)

## position (Sound.position, propriété)

public position : Number [lecture seule]

La durée de diffusion du son exprimée en millisecondes. Si le son est lu en boucle, la position est réinitialisée sur 0 au début de chaque boucle.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

### Exemple

Consultez la section `Sound.duration` pour obtenir un exemple d'utilisation de cette propriété.

### Voir également

`duration (propriété Sound.duration)`

## setPan (méthode Sound.setPan)

```
public setPan(value:Number) : Void
```

Détermine le mode de lecture du son dans les canaux gauche et droit (haut-parleurs). Pour les sons mono, *`pan`* détermine le haut-parleur (gauche ou droit) par lequel le son est lu.

**Disponibilité :** ActionScript 1.0 ; Flash Player 5

### Paramètres

*`value`*`:Number` - Entier spécifiant la balance gauche-droite d'un son. Les valeurs valides vont de -100 à 100, où -100 utilise uniquement le canal gauche, 100 utilise uniquement le canal droit et 0 équilibre le son entre les deux canaux.

### Exemple

Consultez `Sound.getPan()` pour obtenir un exemple d'utilisation de cette méthode.

### Voir également

`attachSound (méthode Sound.attachSound)`, `getPan (méthode Sound.getPan)`, `setTransform (méthode Sound.setTransform)`, `setVolume (méthode Sound.setVolume)`, `start (Sound.start method)`

## setTransform (méthode Sound.setTransform)

```
public setTransform(transformObject:Object) : Void
```

Définit les informations de transformation du son (ou balance), pour un objet Sound.

Le `soundTransformObject` est un objet que vous créez à l'aide de la méthode de constructeur de la classe Object générique avec les paramètres spécifiant la manière dont le son est réparti dans les canaux gauche et droit (haut-parleurs).

Les sons nécessitent une grande quantité d'espace disque et de mémoire. Les sons stéréo utilisant deux fois plus de données que les sons mono, il est généralement préférable d'utiliser des sons monos 6 bits de 22 KHz. Vous pouvez utiliser `setTransform()` pour lire des sons monos comme des sons stéréo, lire des sons stéréos comme des sons monos et ajouter des effets intéressants aux sons.

Les propriétés pour `soundTransformObject` sont les suivantes :

`ll` - Pourcentage indiquant la quantité d'entrée gauche à lire dans le haut-parleur gauche (0-100).

`lr` - Pourcentage indiquant la quantité d'entrée droite à lire dans le haut-parleur gauche (0-100).

`rr` - Pourcentage indiquant la quantité d'entrée droite à lire dans le haut-parleur droit (0-100).

`rl` - Pourcentage indiquant la quantité d'entrée gauche à lire dans le haut-parleur droit (0-100).

Le résultat net des paramètres est représenté par la formule suivante :

```
leftOutput = left_input ~ ll + right_input ~ lr
rightOutput = right_input ~ rr + left_input ~ rl
```

Les valeurs pour `left_input` ou `right_input` sont déterminées par le type (stéréo ou mono) de son dans votre fichier SWF.

Les sons stéréos séparent l'entrée du son en parties égales entre les haut-parleurs gauche et droit et présentent les paramètres de transformation suivants par défaut :

```
ll = 100
lr = 0
rr = 100
rl = 0
```

Les sons monos lisent toutes les entrées de son dans le haut-parleur gauche et présentent les paramètres de transformation suivants par défaut :

```
ll = 100
lr = 100
rr = 0
rl = 0
```

**Disponibilité :** ActionScript 1.0 ; Flash Player 5

## Paramètres

*`transformObject`*`:Object` - Objet créé avec le constructeur pour la classe générique Object.

## Exemple

L'exemple suivant illustre comment un réglage peut être effectué avec `setTransform()`, alors que ce dernier est impossible avec `setVolume()` ou `setPan()`, même si ces derniers sont combinés.

Le code suivant crée un nouvel objet `soundTransformObject` et définit ses propriétés de façon à ce que les sons provenant des deux canaux soient diffusés uniquement dans le canal gauche.

```
var mySoundTransformObject:Object = new Object();
mySoundTransformObject.ll = 100;
mySoundTransformObject.lr = 100;
mySoundTransformObject.rr = 0;
mySoundTransformObject.rl = 0;
```

Pour appliquer l'objet`soundTransformObject` à un objet Sound, vous devez transmettre cet objet à l'objet Sound avec `setTransform()`, comme il est indiqué ci-dessous :

```
my_sound.setTransform(mySoundTransformObject);
```

L'exemple suivant lit un son stéréo en mono. L'objet `soundTransformObjectMono` prend alors les paramètres suivants :

```
var mySoundTransformObjectMono:Object = new Object();
mySoundTransformObjectMono.ll = 50;
mySoundTransformObjectMono.lr = 50;
mySoundTransformObjectMono.rr = 50;
mySoundTransformObjectMono.rl = 50;
my_sound.setTransform(mySoundTransformObjectMono);
```

Cet exemple lit le canal gauche à demi-capacité et ajoute le reste du canal gauche au canal droit. L'objet `soundTransformObjectHalf` a les paramètres suivants :

```
var mySoundTransformObjectHalf:Object = new Object();
mySoundTransformObjectHalf.ll = 50;
mySoundTransformObjectHalf.lr = 0;
mySoundTransformObjectHalf.rr = 100;
mySoundTransformObjectHalf.rl = 50;
my_sound.setTransform(mySoundTransformObjectHalf);

var mySoundTransformObjectHalf:Object = {ll:50, lr:0, rr:100, rl:50};
```

Consultez aussi l'exemple relatif à `Sound.getTransform()`.

## Voir également

`Object`, `getTransform (méthode Sound.getTransform)`

## setVolume (méthode Sound.setVolume)

```
public setVolume(value:Number) : Void
```

Définit le volume pour l'objet Sound.

**Disponibilité :** ActionScript 1.0 ; Flash Player 5

### Paramètres

*value*:Number - Nombre compris entre 0 et 100 et représentant un niveau de volume. 100 correspond au volume maximum et 0 à muet. Le paramètre par défaut est 100.

### Exemple

Consultez `Sound.getVolume()` pour obtenir un exemple d'utilisation de cette méthode.

### Voir également

setPan (méthode Sound.setPan), setTransform (méthode Sound.setTransform)

## Sound, constructeur

```
public Sound([target:Object])
```

Crée un nouvel objet Sound pour un clip spécifique. Si vous ne spécifiez pas d'occurrence cible, l'objet Sound contrôle tous les sons de l'animation.

**Disponibilité :** ActionScript 1.0 ; Flash Player 5

### Paramètres

*target*:Object [facultatif] - Occurrence de clip sur laquelle porte l'objet Sound.

### Exemple

L'exemple suivant crée un nouvel objet Sound appelé `global_sound`. La deuxième ligne appelle `setVolume()` et règle le volume de l'ensemble des sons de l'animation sur 50 %.

```
var global_sound:Sound = new Sound();
global_sound.setVolume(50);
```

L'exemple suivant crée un nouvel objet Sound, le transmet au clip cible *my_mc* et appelle la méthode `start`, ce qui active tout son présent dans *my_mc*.

```
var movie_sound:Sound = new Sound(my_mc);
movie_sound.start();
```

# start (Sound.start method)

`public start([secondOffset:Number], [loops:Number]) : Void`

Commence à lire le dernier son associé depuis le début si aucun paramètre n'est spécifié, ou en commençant à l'endroit spécifié par le paramètre `secondOffset`.

**Disponibilité :** ActionScript 1.0 ; Flash Player 5

## Paramètres

*`secondOffset`*`:Number` [facultatif] - Paramètre qui permet de lire un son à partir d'un point spécifique du fichier. Par exemple, si vous disposez d'un son de 30 secondes et souhaitez faire commencer la lecture à partir de la moitié de ce dernier, spécifiez 15 pour le paramètre `secondOffset`. Le son n'est pas retardé de 15 secondes, il est lu à partir de sa 15ième seconde.

*`loops`*`:Number` [facultatif] - Paramètre permettant de spécifier le nombre de lectures consécutives du son. Ce paramètre n'est pas disponible si le son est diffusé en continu.

## Exemple

L'exemple suivant crée un nouvel objet Sound et charge un son. Le chargement du son est traité par le gestionnaire `onLoad`, qui permet également de diffuser le morceau à l'issue de son chargement. Ensuite, le son est diffusé avec la méthode `start()`. Créez un nouveau fichier FLA et ajoutez le code ActionScript suivant à votre fichier FLA ou AS. Pour que cet exemple fonctionne, vous devez disposer d'un fichier MP3 appelé song1.mp3 et situé dans le même répertoire que votre fichier FLA ou AS.

```
this.createTextField("status_txt", this.getNextHighestDepth(), 0,0,100,22);

// create a new Sound object
var my_sound:Sound = new Sound();
// if the sound loads, play it; if not, trace failure loading
my_sound.onLoad = function(success:Boolean) {
  if (success) {
  my_sound.start();
  status_txt.text = "Sound loaded";
  } else {
  status_txt.text = "Sound failed";
  }
};
// load the sound
my_sound.loadSound("song1.mp3", true);
```

La méthode `MovieClip.getNextHighestDepth()` utilisée dans cet exemple requiert Flash Player 7 ou version ultérieure. Si votre fichier SWF comporte un composant de la version 2, utilisez la classe DepthManager des composants de la version 2 au lieu de la méthode `MovieClip.getNextHighestDepth()`.

### Voir également

stop (Sound.stop method)

## stop (Sound.stop method)

```
public stop([linkageID:String]) : Void
```

Arrête tous les sons en cours de lecture si aucun paramètre n'est spécifié, ou uniquement le son spécifié dans le paramètre `idName`.

**Disponibilité :** ActionScript 1.0 ; Flash Player 5

### Paramètres

*`linkageID`*`:String` [facultatif] - Paramètre spécifiant un son dont vous devez arrêter la diffusion. Le paramètre `idName` doit figurer entre guillemets (" ").

### Exemple

L'exemple suivant utilise deux boutons, `stop_btn` et `play_btn`, pour contrôler la lecture d'un son qui se charge dans un fichier SWF. Ajoutez deux boutons à votre document et ajoutez le code ActionScript suivant à votre fichier FLA ou AS :

```
var my_sound:Sound = new Sound();
my_sound.loadSound("song1.mp3", true);

stop_btn.onRelease = function() {
  trace("sound stopped");
  my_sound.stop();
};
play_btn.onRelease = function() {
  trace("sound started");
  my_sound.start();
};
```

### Voir également

start (Sound.start method)

# Scène

```
Object
  |
  +-Stage
```

```
public class Stage
extends Object
```

La classe Stage est une classe de niveau supérieur dont les méthodes, les propriétés, et les gestionnaires sont accessibles sans l'aide d'un constructeur. Utilisez les méthodes et propriétés de cette classe pour accéder aux informations et les modifier dans les limites d'un fichier SWF.

**Disponibilité :** ActionScript 1.0 ; Flash Player 5

## Résumé des propriétés

| Modificateurs | Propriété | Description |
|---|---|---|
| `static` | `align:String` | Indique l'alignement actuel du fichier SWF dans le lecteur ou le navigateur. |
| `static` | `height:Number` | Propriété (lecture seule) : indique la hauteur actuelle, en pixels, de la scène. |
| `static` | `scaleMode:String` | Indique le redimensionnement actuel du fichier SWF dans Flash Player. |
| `static` | `showMenu:Boolean` | Spécifie l'affichage ou le masquage des éléments par défaut dans le menu contextuel de Flash Player. |
| `static` | `width:Number` | Propriété (lecture seule) : indique la largeur actuelle, en pixels, de la scène. |

*Propriétés héritées de la classe Object*

```
constructeur (propriété Object.constructor), __proto__ (Object.__proto__
property), prototype (Object.prototype, propriété), __resolve
(Object.__resolve, propriété)
```

## Résumé des événements

| Evénement | Description |
|---|---|
| `onResize = function() {}` | Appelé lorsque `Stage.scaleMode` est défini sur noScale et le fichier SWF est redimensionné. |

### Résumé de la méthode

| Modificateurs | Signature | Description |
|---|---|---|
| static | addListener(listener:Object) : Void | Détecte le moment où un fichier SWF est redimensionné (mais uniquement si Stage.scaleMode = "noScale"). |
| static | removeListener(listener:Object) : Boolean | Supprime un objet écouteur créé avec addListener(). |

*Méthodes héritées de la classe Object*

addProperty (méthode Object.addProperty), hasOwnProperty (méthode Object.hasOwnProperty), isPropertyEnumerable (méthode Object.isPropertyEnumerable), isPrototypeOf (méthode Object.isPrototypeOf), registerClass (méthode Object.registerClass), toString (méthode Object.toString), unwatch (méthode Object.unwatch), valueOf (méthode Object.valueOf), watch (méthode Object.watch)

# addListener (méthode Stage.addListener)

```
public static addListener(listener:Object) : Void
```

Détecte le moment où un fichier SWF est redimensionné (mais uniquement si `Stage.scaleMode = "noScale"`). La méthode `addListener()` ne fonctionne pas avec le réglage de redimensionnement du clip (`showAll`) par défaut ou d'autres réglages de redimensionnement (`exactFit` et `noBorder`).

Pour utiliser `addListener()`, vous devez d'abord créer un *objet écouteur*. Les objets écouteur de scène reçoivent une notification de `Stage.onResize`.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

## Paramètres

*`listener`*`:Object` - Objet qui écoute les notifications de rappel en provenance de l'événement `Stage.onResize` event.

## Exemple

Cet exemple crée un nouvel objet écouteur appelé `stageListener`. Il utilise ensuite `stageListener` pour appeler `onResize` et définit une fonction qui sera appelée lorsque `onResize` sera déclenché. Enfin, le code ajoute l'objet `stageListener` à la liste de rappel de l'objet Stage. Les objets écouteur permettent à plusieurs objets d'écouter les notifications de redimensionnement.

```
this.createTextField("stageSize_txt", this.getNextHighestDepth(), 10, 10,
  100, 22);
var stageListener:Object = new Object();
stageListener.onResize = function() {
  stageSize_txt.text = "w:"+Stage.width+", h:"+Stage.height;
};
Stage.scaleMode = "noScale";
Stage.addListener(stageListener);
```

### Voir également

onResize (Stage.onResize, écouteur d'événements), removeListener (méthode Stage.removeListener)

## align (propriété Stage.align)

```
public static align : String
```

Indique l'alignement actuel du fichier SWF dans le lecteur ou le navigateur.

Le tableau suivant énumère les valeurs pour la propriété `align`. Toute valeur non indiquée ici centre le fichier SWF dans Flash Player ou le navigateur, ce qui constitue la valeur par défaut.

| *Valeur* | *Vertical* | *Horizontal* |
|---|---|---|
| "T" | haut | centre |
| "B" | bas | centre |
| "L" | centre | gauche |
| "R" | centre | droite |
| "TL" | haut | gauche |
| "TR" | haut | droite |
| "BL" | bas | gauche |
| "BR" | bas | droite |

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

### Exemple

L'exemple suivant illustre différents alignements du fichier SWF. Ajoutez une occurrence ComboBox à votre document avec le nom d'occurrence `stageAlign_cb`. Ajoutez le code ActionScript suivant à votre fichier FLA ou AS :

```
var stageAlign_cb:mx.controls.ComboBox;
stageAlign_cb.dataProvider = ['T', 'B', 'L', 'R', 'TL', 'TR', 'BL', 'BR'];
var cbListener:Object = new Object();
cbListener.change = function(evt:Object) {
```

```
  var align:String = evt.target.selectedItem;
  Stage.align = align;
};
stageAlign_cb.addEventListener("change", cbListener);
Stage.scaleMode = "noScale";
```

Sélectionnez différents paramètres d'alignement à partir de la zone de liste déroulante.

# height (propriété Stage.height)

```
public static height : Number
```

Propriété (lecture seule) : indique la hauteur actuelle, en pixels, de la scène. Lorsque la valeur de `Stage.scaleMode` est noScale, la propriété height représente la hauteur de Flash Player. Lorsque la valeur de `Stage.scaleMode` n'est pas noScale, height représente la hauteur du fichier SWF.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

## Exemple

Cet exemple crée un nouvel objet écouteur appelé `stageListener`. Il utilise ensuite `myListener` pour appeler `onResize` et définit une fonction qui sera appelée lorsque `onResize` sera déclenché. Enfin, le code ajoute l'objet `myListener` à la liste de rappel de l'objet Stage. Les objets écouteur permettent à plusieurs objets d'écouter les notifications de redimensionnement.

```
this.createTextField("stageSize_txt", this.getNextHighestDepth(), 10, 10,
  100, 22);
var stageListener:Object = new Object();
stageListener.onResize = function() {
  stageSize_txt.text = "w:"+Stage.width+", h:"+Stage.height;
};
Stage.scaleMode = "noScale";
Stage.addListener(stageListener);
```

## Voir également

`align (propriété Stage.align)`, `scaleMode (propriété Stage.scaleMode)`, `width (propriété Stage.width)`

# onResize (Stage.onResize, écouteur d'événements)

```
onResize = function() {}
```

Appelé lorsque `Stage.scaleMode` est défini sur noScale et le fichier SWF est redimensionné. Vous pouvez utiliser ce gestionnaire d'événements pour écrire une fonction qui positionne les objets sur la scène lorsqu'un fichier SWF est redimensionné.

```
myListener.onResize = function()[
 // your statements here
 }
```

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

### Exemple

L'exemple suivant affiche un message dans le panneau de sortie lorsque la scène est redimensionnée :

```
Stage.scaleMode = "noScale"
var myListener:Object = new Object();
myListener.onResize = function () {
  trace("Stage size is now " + Stage.width + " by " + Stage.height);
}
Stage.addListener(myListener);
// later, call Stage.removeListener(myListener)
```

### Voir également

scaleMode (propriété Stage.scaleMode), addListener (méthode Stage.addListener), removeListener (méthode Stage.removeListener)

## removeListener (méthode Stage.removeListener)

```
public static removeListener(listener:Object) : Boolean
```

Supprime un objet écouteur créé avec addListener().

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

### Paramètres

*listener*:Object - Objet ajouté à la liste de rappel d'un autre objet avec addListener().

### Valeur renvoyée

Boolean - Valeur booléenne.

### Exemple

L'exemple suivant affiche les dimensions de la scène dans un champ texte créé de façon dynamique. Lorsque vous redimensionnez la scène, les valeurs du champ texte sont mises à jour. Créez un bouton avec le nom d'occurrence remove_btn. Ajoutez le code ActionScript suivant à l'image1 du scénario :

```
this.createTextField("stageSize_txt", this.getNextHighestDepth(), 10, 10,
  100, 22);
```

```
stageSize_txt.autoSize = true;
stageSize_txt.border = true;
var stageListener:Object = new Object();
stageListener.onResize = function() {
  stageSize_txt.text = "w:"+Stage.width+", h:"+Stage.height;
};
Stage.addListener(stageListener);

remove_btn.onRelease = function() {
  stageSize_txt.text = "Removing Stage listener...";
  Stage.removeListener(stageListener);
}
```

Choisissez Contrôle > Testez l'animation pour tester cet exemple. Les valeurs qui s'affichent dans le champ texte sont mises à jour lorsque vous redimensionnez l'environnement de test. Lorsque vous cliquez sur `remove_btn`, l'écouteur est supprimé et les valeurs ne sont plus mises à jour dans le champ texte.

### Voir également

`addListener (méthode Stage.addListener)`

## scaleMode (propriété Stage.scaleMode)

```
public static scaleMode : String
```

Indique le redimensionnement actuel du fichier SWF dans Flash Player. La propriété `scaleMode` oblige le fichier SWF à passer en un mode de redimensionnement spécifique. Par défaut, le fichier SWF utilise les paramètres HTML définis dans la boîte de dialogue Paramètres de publication.

La propriété `scaleMode` peut utiliser les valeurs `"exactFit"`, `"showAll"`, `"noBorder"` et `"noScale"`. Toute autre valeur définit la propriété `scaleMode` sur la valeur par défaut `"showAll"`.

- `showAll` (valeur par défaut) rend visible la totalité du contenu Flash dans la zone définie, sans distorsion, tout en conservant les proportions d'origine de l'animation. Des bordures peuvent apparaître de part et d'autre de l'application.
- `noBorder` redimensionne le contenu Flash de façon à ce qu'il remplisse la zone définie, sans distorsion mais avec un recadrage éventuel, tout en conservant les proportions d'origine de l'application.
- `exactFit` rend tout le contenu Flash visible dans la zone spécifiée sans essayer de préserver les proportions d'origine. Une distorsion peut se produire.

- `noScale` fixe la taille du contenu Flash, de sorte qu'elle n'est pas modifiée même si la taille de la fenêtre du lecteur change. Un recadrage peut être effectué si la fenêtre du lecteur est plus petite que le contenu Flash.

*Remarque :* Le réglage par défaut est showAll, sauf en mode Tester l'animation, où le réglage par défaut est noScale

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

### Exemple

L'exemple suivant illustre différents paramètres de redimensionnement du fichier SWF. Ajoutez une occurrence ComboBox à votre document avec le nom d'occurrence `scaleMode_cb`. Ajoutez le code ActionScript suivant à votre fichier FLA ou AS :

```
var scaleMode_cb:mx.controls.ComboBox;
scaleMode_cb.dataProvider = ["showAll", "exactFit", "noBorder", "noScale"];
var cbListener:Object = new Object();
cbListener.change = function(evt:Object) {
  var scaleMode_str:String = evt.target.selectedItem;
  Stage.scaleMode = scaleMode_str;
};
scaleMode_cb.addEventListener("change", cbListener);
```

Pour afficher un autre exemple, consultez le fichier stagesize.fla dans le dossier d'exemples ActionScript. La liste suivante fournit des chemins types vers le dossier d'exemples ActionScript :

- Windows : *lecteur d'amorçage*\Program Files\Macromedia\Flash 8\Samples and Tutorials\Samples\ActionScript
- Macintosh : *disque dur Macintosh*/Applications/Macromedia Flash 8/Samples and Tutorials/Samples/ActionScript

## showMenu (propriété Stage.showMenu)

```
public static showMenu : Boolean
```

Spécifie l'affichage ou le masquage des éléments par défaut dans le menu contextuel de Flash Player. Si `showMenu` est défini sur `true` (valeur par défaut), tous les éléments du menu contextuel s'affichent. Si `showMenu` est défini sur `false`, seuls les éléments Paramètres et A propos de Macromedia Flash Player s'affichent.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

### Exemple

L'exemple suivant crée un lien texte sur lequel l'utilisateur peut cliquer pour activer ou désactiver le menu contextuel de Flash Player.

```
this.createTextField("showMenu_txt", this.getNextHighestDepth(), 10, 10,
  100, 22);
showMenu_txt.html = true;
showMenu_txt.autoSize = true;
showMenu_txt.htmlText = "<a
  href=\"asfunction:toggleMenu\"><u>Stage.showMenu = "+Stage.showMenu+"</
  u></a>";
function toggleMenu() {
  Stage.showMenu = !Stage.showMenu;
  showMenu_txt.htmlText = "<a
  href=\"asfunction:toggleMenu\"><u>Stage.showMenu = "+Stage.showMenu+"</
  u></a>";
}
```

### Voir également

`ContextMenu`, `ContextMenuItem`

## width (propriété Stage.width)

```
public static width : Number
```

Propriété (lecture seule) : indique la largeur actuelle, en pixels, de la scène. Lorsque la valeur de `Stage.scaleMode` est `"noScale"`, la propriété `width` représente la largeur de Flash Player. Ceci signifie que `Stage.width` varie en fonction du redimensionnement de la fenêtre du lecteur. Lorsque la valeur de `Stage.scaleMode` n'est pas `"noScale"`, `width` représente la largeur du fichier SWF telle que définie en contexte de création dans la boîte de dialogue Propriétés du document. Ceci signifie que la valeur de la largeur `width` reste constante lorsque vous redimensionnez la fenêtre du lecteur.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

### Exemple

Cet exemple crée un nouvel objet écouteur appelé `stageListener`. Il utilise ensuite `stageListener` pour appeler `onResize` et définit une fonction qui sera appelée lorsque `onResize` sera déclenché. Enfin, le code ajoute l'objet `stageListener` à la liste de rappel de l'objet Stage. Les objets écouteur permettent à plusieurs objets d'écouter les notifications de redimensionnement.

```
this.createTextField("stageSize_txt", this.getNextHighestDepth(), 10, 10,
  100, 22);
var stageListener:Object = new Object();
```

```
stageListener.onResize = function() {
  stageSize_txt.text = "w:"+Stage.width+", h:"+Stage.height;
};
Stage.scaleMode = "noScale";
Stage.addListener(stageListener);
```

### Voir également

`align (propriété Stage.align)`, `height (propriété Stage.height)`, `scaleMode (propriété Stage.scaleMode)`

# String

```
Object
  |
  +-String
```

`public class` ***String***
`extends Object`

La classe String est une enveloppe pour le type de données primitif de la chaîne et fournit des méthodes et des propriétés qui vous permettent de modifier les types de valeur de la chaîne primitifs. Vous pouvez convertir la valeur d'un objet en une chaîne à l'aide de la fonction `String()`.

Toutes les méthodes de la classe String, à l'exception de `concat()`, `fromCharCode()`, `slice()` et `substr()`, sont génériques, ce qui signifie que les méthodes appellent `toString()` avant d'effectuer leurs opérations, et vous pouvez utiliser ces méthodes avec d'autres objets qui ne sont pas de type String.

Tous les index de chaîne étant basés sur zéro, l'index du dernier caractère pour une chaîne `x` est `x.length - 1`.

Vous pouvez appeler l'une des méthodes de la classe String à l'aide de la méthode constructeur `new String` ou d'une valeur de littéral de chaîne. Si vous spécifiez un littéral de chaîne, l'interpréteur ActionScript le convertit automatiquement en un objet String temporaire, appelle la méthode, puis supprime l'objet String temporaire. Vous pouvez également utiliser la propriété `String.length` avec un littéral de chaîne.

Ne confondez pas un littéral de chaîne avec un objet String. Dans l'exemple suivant, la première ligne de code crée le littéral de chaîne `first_string`, et la deuxième ligne de code crée l'objet String `second_string` :

```
var first_string:String = "foo"
var second_string:String = new String("foo")
```

Utilisez des littéraux de chaîne sauf si vous avez spécifiquement besoin d'utiliser un objet String.

**Disponibilité :** ActionScript 1.0 ; Flash Player 5

## Résumé des propriétés

| Modificateurs | Propriété | Description |
|---|---|---|
| | `length:Number` | Entier spécifiant le nombre de caractères dans l'objet String spécifié. |

*Propriétés héritées de la classe Object*

`constructeur (propriété Object.constructor), __proto__ (Object.__proto__ property), prototype (Object.prototype, propriété), __resolve (Object.__resolve, propriété)`

## Résumé des constructeurs

| Signature | Description |
|---|---|
| `String(value:String)` | Crée un nouvel objet String. |

## Résumé de la méthode

| Modificateurs | Signature | Description |
|---|---|---|
| | `charAt(index:Number) : String` | Renvoie le caractère à la position spécifiée par le paramètre `index`. |
| | `charCodeAt(index:Number) : Number` | Renvoie un entier 16 bits de 0 à 65535 qui représente le caractère spécifié par `index`. |
| | `concat(value:Object) : String` | Combine la valeur de l'objet String avec les paramètres et renvoie la nouvelle chaîne formée : la valeur d'origine, `my_str`, n'est pas modifiée. |
| `static` | `fromCharCode() : String` | Renvoie une chaîne comprenant les caractères représentés par les valeurs Unicode dans les paramètres. |
| | `indexOf(value:String, [startIndex:Number]) : Number` | Recherche la chaîne et renvoie la position de la première occurrence de `value` détectée au niveau de ou après `startIndex` dans la chaîne appelante. |
| | `lastIndexOf(value:String, [startIndex:Number]) : Number` | Recherche la chaîne de droite à gauche et renvoie l'index de la dernière occurrence de `value` détectée avant `startIndex` dans la chaîne appelante. |

| Modificateurs | Signature | Description |
|---|---|---|
| | `slice(start:Number, end:Number) : String` | Renvoie une chaîne qui contient le caractère `start` et tous les autres caractères jusqu'au caractère `end`, ce dernier n'étant pas inclus. |
| | `split(delimiter:String, [limit:Number]) : Array` | Divise un objet String en sous-chaînes en le séparant aux endroits où le paramètre `delimiter` spécifié apparaît et renvoie les sous-chaînes dans un tableau. |
| | `substr(start:Number, length:Number) : String` | Renvoie les caractères dans une chaîne à partir de l'index spécifié dans le paramètre `start` par le nombre de caractères spécifié dans le paramètre `length`. |
| | `substring(start:Number, end:Number) : String` | Renvoie une chaîne comprenant les caractères entre les points spécifiés par les paramètres `start` et `end`. |
| | `toLowerCase() : String` | Renvoie une copie de l'objet `String` avec tous les caractères majuscules convertis en minuscules. |
| | `toString() : String` | Renvoie les propriétés d'un objet en tant que chaînes, que ces propriétés soient des chaînes ou non. |
| | `toUpperCase() : String` | Renvoie une copie de l'objet String, avec tous les caractères minuscules convertis en majuscules. |
| | `valueOf() : String` | Renvoie une chaîne. |

*Méthodes héritées de la classe Object*

`addProperty (méthode Object.addProperty)`, `hasOwnProperty (méthode Object.hasOwnProperty)`, `isPropertyEnumerable (méthode Object.isPropertyEnumerable)`, `isPrototypeOf (méthode Object.isPrototypeOf)`, `registerClass (méthode Object.registerClass)`, `toString (méthode Object.toString)`, `unwatch (méthode Object.unwatch)`, `valueOf (méthode Object.valueOf)`, `watch (méthode Object.watch)`

# charAt (méthode String.charAt)

```
public charAt(index:Number) : String
```

Renvoie le caractère à la position spécifiée par le paramètre `index`. Si `index` n'est pas un nombre compris entre 0 et `string.length` - 1, une chaîne vide est renvoyée.

Cette méthode est similaire à `String.charCodeAt()` sauf que la valeur renvoyée est un caractère, et non pas un code de caractère d'entier 16 bits.

**Disponibilité :** ActionScript 1.0 ; Flash Player 5

### Paramètres

*index*:Number - Entier spécifiant la position d'un caractère dans la chaîne. Le premier caractère est indiqué par 0, et le dernier par my_str .length-1.

### Valeur renvoyée

String - Caractère correspondant à l'index spécifié. Ou un objet String vide si l'index spécifié n'est pas compris dans la plage des index de cet objet String.

### Exemple

Dans l'exemple suivant, cette méthode est appelée pour la première lettre de la chaîne "Chris" :

```
var my_str:String = "Chris";
var firstChar_str:String = my_str.charAt(0);
trace(firstChar_str); // output: C
```

### Voir également

charCodeAt (méthode String.charCodeAt)

## charCodeAt (méthode String.charCodeAt)

```
public charCodeAt(index:Number) : Number
```

Renvoie un entier 16 bits de 0 à 65535 qui représente le caractère spécifié par index. Si index index n'est pas un nombre compris entre 0 et string.length - 1, NaN est renvoyé.

Cette méthode est similaire à String.charAt() à l'exception que la valeur renvoyée est un code de caractère d'entier 16 bits, pas un caractère.

**Disponibilité :** ActionScript 1.0 ; Flash Player 5

### Paramètres

*index*:Number - Entier spécifiant la position d'un caractère dans la chaîne. Le premier caractère est indiqué par 0, et le dernier par my_str.length - 1.

### Valeur renvoyée

Number - Entier qui représente le caractère spécifié par index.

### Exemple

Dans l'exemple suivant, cette méthode est appelée pour la première lettre de la chaîne « Chris » :

```
var my_str:String = "Chris";
```

```
var firstChar_num:Number = my_str.charCodeAt(0);
trace(firstChar_num); // output: 67
```

### Voir également

charAt (méthode String.charAt)

## concat (String.concat method)

```
public concat(value:Object) : String
```

Combine la valeur de l'objet String avec les paramètres et renvoie la nouvelle chaîne formée : la valeur d'origine, `my_str`, n'est pas modifiée.

**Disponibilité :** ActionScript 1.0 ; Flash Player 5

### Paramètres

*`value`*`:Object` - value1[,...valueN] Valeurs supérieures ou égales à zéro à concaténer.

### Valeur renvoyée

`String` - Chaîne.

### Exemple

L'exemple suivant crée deux chaînes et les combine avec `String.concat()`:

```
var stringA:String = "Hello";
var stringB:String = "World";
var combinedAB:String = stringA.concat(" ", stringB);
trace(combinedAB); // output: Hello World
```

## fromCharCode (méthode String.fromCharCode)

```
public static fromCharCode() : String
```

Renvoie une chaîne comprenant les caractères représentés par les valeurs Unicode dans les paramètres.

**Disponibilité :** ActionScript 1.0 ; Flash Player 5

### Valeur renvoyée

`String` - Chaîne correspondant à la valeur des codes de caractère Unicode spécifiés.

### Exemple

L'exemple suivant utilise `fromCharCode()` pour insérer un @ dans l'adresse électronique :

```
var address_str:String = "dog"+String.fromCharCode(64)+"house.net";
```

```
trace(address_str); // output: dog@house.net
```

# indexOf (méthode String.indexOf)

```
public indexOf(value:String, [startIndex:Number]) : Number
```

Recherche la chaîne et renvoie la position de la première occurrence de `value` détectée au niveau de ou après `startIndex` dans la chaîne appelante. Cet index est basé sur zéro, ce qui signifie que le premier caractère d'une chaîne est considéré comme étant à l'index 0, pas l'index 1. Si `value` n'est pas détecté, la méthode renvoie -1.

**Disponibilité :** ActionScript 1.0 ; Flash Player 5

## Paramètres

*`value`*`:String` - Chaîne ; sous-chaîne à rechercher.

*`startIndex`*`:Number` [facultatif] - Entier spécifiant l'index de départ de la recherche.

## Valeur renvoyée

`Number` - Position de la première occurrence de la sous-chaîne spécifiée ou -1.

## Exemple

Les exemples suivants utilisent `indexOf()` pour renvoyer l'index de caractères et de sous-chaînes :

```
var searchString:String = "Lorem ipsum dolor sit amet.";
var index:Number;

index = searchString.indexOf("L");
trace(index); // output: 0

index = searchString.indexOf("l");
trace(index); // output: 14

index = searchString.indexOf("i");
trace(index); // output: 6

index = searchString.indexOf("ipsum");
trace(index); // output: 6

index = searchString.indexOf("i", 7);
trace(index); // output: 19

index = searchString.indexOf("z");
trace(index); // output: -1
```

### Voir également

lastIndexOf (méthode String.lastIndexOf)

## lastIndexOf (méthode String.lastIndexOf)

```
public lastIndexOf(value:String, [startIndex:Number]) : Number
```

Recherche la chaîne de droite à gauche et renvoie l'index de la dernière occurrence de `value` détectée avant `startIndex` dans la chaîne appelante. Cet index est basé sur zéro, ce qui signifie que le premier caractère dans une chaîne est considéré comme étant à l'index 0, pas l'index 1. Si `value` n'est pas détectée, la méthode renvoie -1.

**Disponibilité :** ActionScript 1.0 ; Flash Player 5

### Paramètres

*`value`*`:String` - Chaîne à rechercher.

*`startIndex`*`:Number` [facultatif] - Entier spécifiant le point de départ de la recherche de `value`.

### Valeur renvoyée

`Number` - Position de la dernière occurrence de la sous-chaîne spécifiée ou -1.

### Exemple

L'exemple suivant indique comment utiliser `lastIndexOf()` pour renvoyer l'index d'un caractère :

```
var searchString:String = "Lorem ipsum dolor sit amet.";
var index:Number;

index = searchString.lastIndexOf("L");
trace(index); // output: 0

index = searchString.lastIndexOf("l");
trace(index); // output: 14

index = searchString.lastIndexOf("i");
trace(index); // output: 19

index = searchString.lastIndexOf("ipsum");
trace(index); // output: 6

index = searchString.lastIndexOf("i", 18);
trace(index); // output: 6

index = searchString.lastIndexOf("z");
```

```
trace(index); // output: -1
```

### Voir également

indexOf (méthode String.indexOf)

## length (propriété String.length)

```
longueur publique : Number
```

Entier spécifiant le nombre de caractères dans l'objet String spécifié.

Tous les index de chaîne étant basés sur zéro, l'index du dernier caractère pour une chaîne `x` est `x.length - 1`.

**Disponibilité :** ActionScript 1.0 ; Flash Player 5

### Exemple

L'exemple suivant crée un nouvel objet de type chaîne et utilise `String.length` pour en compter le nombre de caractères :

```
var my_str:String = "Hello world!";
trace(my_str.length); // output: 12
```

L'exemple suivant boucle de 0 à `my_str.length`. Le code vérifie les caractères au sein d'une chaîne et, si cette chaîne contient le caractère `@`, `true` est affiché dans le panneau de sortie. En l'absence du caractère `@` le programme affiche `false` dans le panneau de sortie.

```
function checkAtSymbol(my_str:String):Boolean {
  for (var i = 0; i<my_str.length; i++) {
  if (my_str.charAt(i) == "@") {
    return true;
  }
  }
  return false;
}

trace(checkAtSymbol("dog@house.net")); // output: true
trace(checkAtSymbol("Chris")); // output: false
```

Vous trouverez également un exemple dans le fichier Strings.fla du dossier HelpExamples. La liste suivante présente les chemins type vers ce dossier :

- Windows : *lecteur d'amorçage*\Program Files\Macromedia\Flash 8\Samples and Tutorials\Samples\ActionScript
- Macintosh : *disque dur Macintosh*/Applications/Macromedia Flash 8/Samples and Tutorials/Samples/ActionScript

# slice (méthode String.slice)

```
public slice(start:Number, end:Number) : String
```

Renvoie une chaîne qui contient le caractère `start` et tous les autres caractères jusqu'au caractère `end`, ce dernier n'étant pas inclus. L'objet String d'origine n'est pas modifié. Si le paramètre `end` n'est pas spécifié, la fin de la sous-chaîne correspond à la fin de la chaîne. Si le caractère indexé par `start` est identique au caractère indexé par `end` ou s'il trouve à droite de ce caractère, la méthode renvoie une chaîne vide.

**Disponibilité :** ActionScript 1.0 ; Flash Player 5

## Paramètres

*`start`*`:Number` - Index basé sur zéro du point de départ de la découpe. Si `start` correspond à un nombre négatif, le point de départ est déterminé à partir de la fin de la chaîne, -1 représentant le dernier caractère.

*`end`*`:Number` - Entier correspondant à 1+ l'index du point de terminaison de la découpe. Le caractère indexé par le paramètre `end` n'est pas inclus dans la chaîne extraite. Si ce paramètre est omis, `String.length` est utilisé. Si `end` correspond à un nombre négatif, le point de terminaison est calculé à partir de la fin de la chaîne, -1 représentant le dernier caractère.

## Valeur renvoyée

`String` - Sous-chaîne de la chaîne spécifiée.

## Exemple

L'exemple suivant crée une variable, `my_str`, lui affecte une valeur de type chaîne, puis appelle la méthode `slice()` avec différentes valeurs pour les paramètres `start` et `end`. Tout appel à `slice()` figure dans une instruction `trace()` qui affiche son résultat dans le panneau de sortie.

```
// Index values for the string literal
// positive index: 0 1 2 3 4
// string: L o r e m
// negative index: -5 -4 -3 -2 -1

var my_str:String = "Lorem";

// slice the first character
trace("slice(0,1): "+my_str.slice(0, 1)); // output: slice(0,1): L
trace("slice(-5,1): "+my_str.slice(-5, 1)); // output: slice(-5,1): L

// slice the middle three characters
trace("slice(1,4): "+my_str.slice(1, 4)); // slice(1,4): ore
trace("slice(1,-1): "+my_str.slice(1, -1)); // slice(1,-1): ore
```

```
// slices that return empty strings because start is not to the left of end
trace("slice(1,1): "+my_str.slice(1, 1)); // slice(1,1):
trace("slice(3,2): "+my_str.slice(3, 2)); // slice(3,2):
trace("slice(-2,2): "+my_str.slice(-2, 2)); // slice(-2,2):

// slices that omit the end parameter use String.length, which equals 5
trace("slice(0): "+my_str.slice(0)); // slice(0): Lorem
trace("slice(3): "+my_str.slice(3)); // slice(3): em
```

Vous trouverez également un exemple dans le fichier Strings.fla du dossier HelpExamples. La liste suivante présente les chemins type vers ce dossier :

- Windows : *lecteur d'amorçage*\Program Files\Macromedia\Flash 8\Samples and Tutorials\Samples\ActionScript
- Macintosh : *disque dur Macintosh*/Applications/Macromedia Flash 8/Samples and Tutorials/Samples/ActionScript

### Voir également

`substr (méthode String.substr)`, `substring (méthode String.substring)`

## split (méthode String.split)

```
public split(delimiter:String, [limit:Number]) : Array
```

Divise un objet String en sous-chaînes en le séparant aux endroits où le paramètre `delimiter` spécifié apparaît et renvoie les sous-chaînes dans un tableau. Si vous utilisez une chaîne vide ("") en tant que séparateur, chaque caractère dans la chaîne est placé comme un élément dans le tableau.

Si le paramètre `delimiter` n'est pas défini, l'ensemble de la chaîne est placé dans le premier élément du tableau renvoyé.

**Disponibilité :** ActionScript 1.0 ; Flash Player 5

### Paramètres

*`delimiter`*`:String` - Chaîne ; caractère ou chaîne à partir desquels `my_str` est divisé.

*`limit`*`:Number` [facultatif] - Nombre d'éléments à placer dans le tableau.

### Valeur renvoyée

`Array` - Tableau contenant les sous-chaînes de `my_str`.

## Exemple

L'exemple suivant renvoie un tableau avec cinq éléments :

```
var my_str:String = "P,A,T,S,Y";
var my_array:Array = my_str.split(",");
for (var i = 0; i<my_array.length; i++) {
  trace(my_array[i]);
}
// output:
  P
  A
  T
  S
  Y
```

L'exemple suivant renvoie un tableau avec deux éléments, `"P"` et `"A"`:

```
var my_str:String = "P,A,T,S,Y";
var my_array:Array = my_str.split(",", 2);
trace(my_array); // output: P,A
```

L'exemple suivant indique que si vous utilisez une chaîne vide ("") en tant que paramètre `delimiter` les caractères de la chaîne sont placés dans le tableau en tant qu'éléments :

```
var my_str:String = new String("Joe");
var my_array:Array = my_str.split("");
for (var i = 0; i<my_array.length; i++) {
  trace(my_array[i]);
}
// output:
  J
  o
  e
```

Vous trouverez également un exemple dans le fichier Strings.fla du dossier HelpExamples. La liste suivante présente les chemins type vers ce dossier :

- Windows : *lecteur d'amorçage*\Program Files\Macromedia\Flash 8\Samples and Tutorials\Samples\ActionScript
- Macintosh : *disque dur Macintosh*/Applications/Macromedia Flash 8/Samples and Tutorials/Samples/ActionScript

## Voir également

`join (méthode Array.join)`

# String, constructeur

```
public String(value:String)
```

Crée un nouvel objet String.

*Remarque :* Les littéraux de chaîne utilisant moins de temps système que les objets String et étant généralement plus faciles à utiliser, vous devez utiliser des littéraux de chaîne à la place du constructeur pour la classe String sauf si vous avez une bonne raison d'utiliser un objet String plutôt qu'un littéral de chaîne.

**Disponibilité :** ActionScript 1.0 ; Flash Player 5

## Paramètres

*value*:String - Valeur initiale du nouvel objet String.

# substr (méthode String.substr)

```
public substr(start:Number, length:Number) : String
```

Renvoie les caractères dans une chaîne à partir de l'index spécifié dans le paramètre `start` par le nombre de caractères spécifié dans le paramètre `length`. La méthode `substr` ne modifie pas la chaîne spécifiée par `my_str`; elle renvoie une nouvelle chaîne.

**Disponibilité :** ActionScript 1.0 ; Flash Player 5

## Paramètres

*start*:Number - Entier qui indique la position du premier caractère à utiliser pour créer la sous-chaîne à partir de `my_str`. Si `start` correspond à un nombre négatif, la position de départ est déterminée à partir de la fin de la chaîne, -1 représentant le dernier caractère.

*length*:Number - Nombre de caractères de la sous-chaîne en cours de création. Si le paramètre `length` n'est pas spécifié, la sous-chaîne contient tous les caractères, du début à la fin de la chaîne.

## Valeur renvoyée

`String` - Sous-chaîne de la chaîne spécifiée.

## Exemple

L'exemple suivant crée une nouvelle chaîne, `my_str` et utilise `substr()` pour renvoyer le deuxième mot de cette chaîne ; tout d'abord avec un paramètre `start` positif, puis un paramètre `start` négatif :

```
var my_str:String = new String("Hello world");
var mySubstring:String = new String();
```

```
mySubstring = my_str.substr(6,5);
trace(mySubstring); // output: world

mySubstring = my_str.substr(-5,5);
trace(mySubstring); // output: world
```

Vous trouverez également un exemple dans le fichier Strings.fla du dossier HelpExamples. La liste suivante présente les chemins type vers ce dossier :

- Windows : *lecteur d'amorçage*\Program Files\Macromedia\Flash 8\Samples and Tutorials\Samples\ActionScript
- Macintosh : *disque dur Macintosh*/Applications/Macromedia Flash 8/Samples and Tutorials/Samples/ActionScript

# substring (méthode String.substring)

```
public substring(start:Number, end:Number) : String
```

Renvoie une chaîne comprenant les caractères entre les points spécifiés par les paramètres `start` et `end`. Si le paramètre `end` n'est pas spécifié, la fin de la sous-chaîne correspond à la fin de la chaîne. Si la valeur de `start` est égale à la valeur `end`, la méthode renvoie une chaîne vide. Si la valeur de `start` est supérieure à la valeur `end`, es paramètres sont automatiquement permutés avant que la fonction s'exécute et la valeur d'origine n'est pas modifiée.

**Disponibilité :** ActionScript 1.0 ; Flash Player 5

## Paramètres

*`start`*`:Number` - Entier qui indique la position du premier caractère à utiliser pour créer la sous-chaîne à partir de `my_str`. Les valeurs valides pour `start` vont de 0 à `String.length` - 1. Si `start` a une valeur négative, 0 est utilisé.

*`end`*`:Number` - Entier qui correspond à 1+ l'index du dernier caractère à extraire dans `my_str`. Les valeurs valides pour `end` vont de 1 à `String.length`. Le caractère indexé par le paramètre `end` n'est pas inclus dans la chaîne extraite. Si ce paramètre est omis, `String.length` est utilisé. Si ce paramètre correspond à une valeur négative, 0 est utilisé.

## Valeur renvoyée

`String` - Sous-chaîne de la chaîne spécifiée.

## Exemple

L'exemple suivant indique comment utiliser `substring()`:

```
var my_str:String = "Hello world";
var mySubstring:String = my_str.substring(6,11);
trace(mySubstring); // output: world
```

L'exemple suivant indique ce qui se produit lorsqu'un paramètre `start` négatif est utilisé :

```
var my_str:String = "Hello world";
var mySubstring:String = my_str.substring(-5,5);
trace(mySubstring); // output: Hello
```

Vous trouverez également un exemple dans le fichier Strings.fla du dossier HelpExamples. La liste suivante présente les chemins type vers ce dossier :

- Windows : *lecteur d'amorçage*\Program Files\Macromedia\Flash 8\Samples and Tutorials\Samples\ActionScript
- Macintosh : *disque dur Macintosh*/Applications/Macromedia Flash 8/Samples and Tutorials/Samples/ActionScript

# toLowerCase (méthode String.toLowerCase)

```
public toLowerCase() : String
```

Renvoie une copie de l'objet `String` avec tous les caractères majuscules convertis en minuscules. La valeur d'origine n'est pas modifiée.

**Disponibilité :** ActionScript 1.0 ; Flash Player 5

## Valeur renvoyée

`String` - Chaîne.

## Exemple

L'exemple suivant crée une chaîne de caractères en majuscules, puis copie cette chaîne avec `toLowerCase()` pour la convertir en minuscules :

```
var upperCase:String = "LOREM IPSUM DOLOR";
var lowerCase:String = upperCase.toLowerCase();
trace("upperCase: " + upperCase); // output: upperCase: LOREM IPSUM DOLOR
trace("lowerCase: " + lowerCase); // output: lowerCase: lorem ipsum dolor
```

Vous trouverez également un exemple dans le fichier Strings.fla du dossier HelpExamples. La liste suivante présente les chemins type vers ce dossier :

- Windows : *lecteur d'amorçage*\Program Files\Macromedia\Flash 8\Samples and Tutorials\Samples\ActionScript
- Macintosh : *disque dur Macintosh*/Applications/Macromedia Flash 8/Samples and Tutorials/Samples/ActionScript

## Voir également

`toUpperCase (méthode String.toUpperCase)`

# toString (méthode String.toString)

```
public toString() : String
```

Renvoie les propriétés d'un objet en tant que chaînes, que ces propriétés soient des chaînes ou non.

**Disponibilité :** ActionScript 1.0 ; Flash Player 5

## Valeur renvoyée

`String` - Chaîne.

## Exemple

Le code suivant renvoie une chaîne en majuscules qui donne la liste de l'ensemble des propriétés d'un objet, que les propriétés soient des chaînes ou non :

```
var employee:Object = new Object();
employee.name = "bob";
employee.salary = 60000;
employee.id = 284759021;

var employeeData:String = new String();
for (prop in employee)
{
  employeeData += employee[prop].toString().toUpperCase() + " ";
}
trace(employeeData);
```

Si la méthode `toString()` n'était pas incluse dans ce code et que la ligne dans la boucle `for` utilisait `employee[prop].toUpperCase()`, la valeur de sortie serait "`undefined undefined BOB`". L'inclusion de la méthode `toString()` produit la valeur de sortie souhaitée : "`284759021 60000 BOB`".

# toUpperCase (méthode String.toUpperCase)

```
public toUpperCase() : String
```

Renvoie une copie de l'objet String, avec tous les caractères minuscules convertis en majuscules. La valeur d'origine n'est pas modifiée.

**Disponibilité :** ActionScript 1.0 ; Flash Player 5

## Valeur renvoyée

`String` - Chaîne.

### Exemple

L'exemple suivant crée une chaîne en minuscules, puis crée une copie de cette chaîne pour la convertir en majuscules avec `toUpperCase()`:

```
var lowerCase:String = "lorem ipsum dolor";
var upperCase:String = lowerCase.toUpperCase();
trace("lowerCase: " + lowerCase); // output: lowerCase: lorem ipsum dolor
trace("upperCase: " + upperCase); // output: upperCase: LOREM IPSUM DOLOR
```

Vous trouverez également un exemple dans le fichier Strings.fla du dossier d'exemples ActionScript. La liste suivante présente les chemins type vers ce dossier :

- Windows : *lecteur d'amorçage*\Program Files\Macromedia\Flash 8\Samples and Tutorials\Samples\ActionScript
- Macintosh : *disque dur Macintosh*/Applications/Macromedia Flash 8/Samples and Tutorials/Samples/ActionScript

### Voir également

`toLowerCase (méthode String.toLowerCase)`

## valueOf (méthode String.valueOf)

```
public valueOf() : String
```

Renvoie une chaîne.

**Disponibilité :** ActionScript 1.0 ; Flash Player 5

### Valeur renvoyée

`String` - Valeur de la chaîne.

### Exemple

L'exemple suivant crée une nouvelle occurrence de l'objet String et montre ensuite que la méthode `valueOf` renvoie une référence à la valeur *primitive* plutôt qu'une occurrence de l'objet.

```
var str:String = new String("Hello World");
var value:String = str.valueOf();
trace(str instanceof String); // true
trace(value instanceof String); // false
trace(str === value); // false
```

# StyleSheet (TextField.StyleSheet)

```
Object
  |
  +-TextField.StyleSheet
```

```
public class StyleSheet
extends Object
```

La classe StyleSheet permet de créer un objet feuille de style contenant des règles de formatage de texte pour la taille et la couleur de la police ainsi que d'autres styles de formatage. Vous pouvez ensuite appliquer des styles définis par une feuille de style à un objet TextField qui contient du texte au format HTML ou XML. Le texte contenu dans l'objet TextField est ensuite automatiquement mis en forme en fonction des styles de balises définis par l'objet StyleSheet. Vous pouvez utiliser des styles de texte pour définir de nouvelles balises de formatage, redéfinir des balises HTML intégrées ou créer des classes de style qui peuvent être appliquées à certaines balises HTML.

Pour appliquer des styles à un objet TextField, attribuez l'objet feuille de style à une propriété `styleSheet`.

Flash Player prend en charge un sous-ensemble de propriétés dans la spécification CSS1 d'origine (www.w3.org/TR/REC-CSS1). Le tableau suivant présente les propriétés et les valeurs prises en charge de la feuille de style en cascade (CSS) et les noms de propriétés ActionScript correspondants. (Chaque nom de propriété ActionScript est tiré du nom de propriété CSS correspondant. Si le nom contient un trait d'union, le trait d'union est omis et le caractère suivant est une majuscule.)

| Propriété CSS | Propriété ActionScript | Utilisation et valeurs prises en charge |
|---|---|---|
| `color` | `color` | Seules les valeurs hexadécimales de couleur sont prises en charge. Les couleurs nommées (comme `blue`) ne sont pas prises en charge. Les couleurs sont écrites au format suivant : `#FF0000`. |
| `display` | `display` | Les valeurs prises en charge sont `inline`, `block` et `none`. |

| Propriété CSS | Propriété ActionScript | Utilisation et valeurs prises en charge |
|---|---|---|
| `font-family` | `fontFamily` | Liste des polices à utiliser, séparées par des virgules, classées par ordre de choix décroissant. Tous les noms de familles de polices peuvent être utilisés. Si vous spécifiez un nom de police générique, il est converti dans la police de périphérique appropriée. Les conversions de police suivantes sont disponibles : `mono` est converti en `_typewriter`, `sans-serif` est converti en `_sans` et `serif` est converti en `_serif`. |
| `font-size` | `fontSize` | Seule la partie numérique de la valeur est utilisée. Les unités (px, pt) ne sont pas analysées. Les pixels et les points sont équivalents. |
| `font-style` | `fontStyle` | Les valeurs reconnues sont `normal` et `italic`. |
| `font-weight` | `fontWeight` | Les valeurs reconnues sont `normal` et `bold`. |
| `kerning` | `kerning` | Les valeurs reconnues sont `true` et `false`. Le crénage est pris en charge uniquement pour les polices incorporées. Certaines polices, telles que Courier New, ne prennent pas en charge le crénage. La propriété de crénage n'est prise en charge que dans les fichiers SWF créés dans Windows, pas dans les fichiers SWF créés sur Macintosh. Cependant, ces fichiers SWF peuvent être lus dans des versions de Flash Player autres que Windows et le crénage s'applique encore. |

| Propriété CSS | Propriété ActionScript | Utilisation et valeurs prises en charge |
|---|---|---|
| `letter-spacing` | `letterSpacing` | La quantité d'espace répartie uniformément entre les caractères. Cette valeur spécifie le nombre de pixels à ajouter après chaque caractère. Une valeur négative comprime l'espace entre les caractères. Seule la partie numérique de la valeur est utilisée. Les unités (px, pt) ne sont pas analysées. Les pixels et les points sont équivalents. |
| `margin-left` | `marginLeft` | Seule la partie numérique de la valeur est utilisée. Les unités (px, pt) ne sont pas analysées. Les pixels et les points sont équivalents. |
| `margin-right` | `marginRight` | Seule la partie numérique de la valeur est utilisée. Les unités (px, pt) ne sont pas analysées. Les pixels et les points sont équivalents. |
| `text-align` | `textAlign` | Les valeurs reconnues sont `left`, `center`, `right` et `justify`. |
| `text-decoration` | `textDecoration` | Les valeurs reconnues sont `none` et `underline`. |
| `text-indent` | `textIndent` | Seule la partie numérique de la valeur est utilisée. Les unités (px, pt) ne sont pas analysées. Les pixels et les points sont équivalents. |

**Disponibilité :** ActionScript 1.0 ; Flash Player 7

## Résumé des propriétés

*Propriétés héritées de la classe Object*

```
constructeur (propriété Object.constructor), __proto__ (Object.__proto__
property), prototype (Object.prototype, propriété), __resolve
(Object.__resolve, propriété)
```

## Résumé des événements

| Evénement | Description |
|---|---|
| `onLoad = function(success:Boolean) {}` | Appelé lorsqu'une opération `load()` est terminée. |

## Résumé des constructeurs

| Signature | Description |
|---|---|
| `StyleSheet()` | Crée un objet StyleSheet. |

## Résumé de la méthode

| Modificateurs | Signature | Description |
|---|---|---|
| | `clear() : Void` | Supprime l'ensemble des styles de l'objet StyleSheet spécifié. |
| | `getStyle(name:String) : Object` | Renvoie une copie de l'objet style associé au style spécifié (`name`). |
| | `getStyleNames() : Array` | Renvoie un tableau qui contient les noms (sous formes de chaînes) de tous les styles enregistrés dans cette feuille de style. |
| | `load(url:String) : Boolean` | Procède au chargement du fichier CSS dans la StyleSheet. |
| | `parseCSS(cssText:String) : Boolean` | Analyse le CSS dans `cssText` et charge la StyleSheet simultanément. |
| | `setStyle(name:String, style:Object) : Void` | Ajoute un nouveau style avec le nom spécifié à l'objet feuille de style. |
| | `transform(style:Object) : TextFormat` | Développe la capacité d'analyse du fichier CSS. |

*Méthodes héritées de la classe Object*

`addProperty (méthode Object.addProperty), hasOwnProperty (méthode Object.hasOwnProperty), isPropertyEnumerable (méthode Object.isPropertyEnumerable), isPrototypeOf (méthode Object.isPrototypeOf), registerClass (méthode Object.registerClass), toString (méthode Object.toString), unwatch (méthode Object.unwatch), valueOf (méthode Object.valueOf), watch (méthode Object.watch)`

# clear (méthode StyleSheet.clear)

```
public clear() : Void
```

Supprime l'ensemble des styles de l'objet StyleSheet spécifié.

**Disponibilité :** ActionScript 1.0 ; Flash Player 7

## Exemple

L'exemple suivant charge une StyleSheet appelée styles.css dans un fichier SWF, puis affiche les styles qui ont été chargés dans le panneau de sortie. Lorsque vous cliquez sur `clear_btn`, tous les styles sont supprimés de l'objet `my_styleSheet`.

```
// Create a new StyleSheet object
import TextField.StyleSheet;
var my_styleSheet:StyleSheet = new StyleSheet();

my_styleSheet.onLoad = function(success:Boolean) {
  if (success) {
    trace("Styles loaded.");
    var styles_array:Array = my_styleSheet.getStyleNames();
    for (var i = 0; I < styles_array.length; i++) {
      trace("\t"+styles_array[i]);
    }
    trace("");
  } else {
    trace("Error loading CSS");
  }
};

// Start the loading operation
my_styleSheet.load("styles.css");

clear_btn.onRelease = function() {
  my_styleSheet.clear();
  trace("Styles cleared.");
  var styles_array:Array = my_styleSheet.getStyleNames();
  for (var i = 0; i<styles_array.length; i++) {
    trace("\t"+styles_array[i]);
  }
  trace("");
};
```

# getStyle (méthode StyleSheet.getStyle)

```
public getStyle(name:String) : Object
```

Renvoie une copie de l'objet style associé au style spécifié (`name`). Si aucun objet style n'est associé à `name`, la méthode renvoie `null`.

**Disponibilité :** ActionScript 1.0 ; Flash Player 7

## Paramètres

*name*:String - Nom du style à extraire.

## Valeur renvoyée

Object - Objet style ; dans tous les autres cas null.

## Exemple

L'exemple suivant charge des styles à partir d'un fichier CSS, analyse la StyleSheet, puis affiche les noms de style et les valeurs des propriétés dans le panneau de sortie. Créez un fichier ActionScript appelé StyleSheetTracer.as et entrez le code suivant dans le fichier :

```
import TextField.StyleSheet;
class StyleSheetTracer {
  // StyleSheetTracer.displayFromURL
  // This method displays the CSS style sheet at
  // the specified URL in the Output panel.
  static function displayFromURL(url:String):Void {
    // Create a new StyleSheet object
    var my_styleSheet:StyleSheet = new StyleSheet();
    // The load operation is asynchronous, so set up
    // a callback function to display the loaded StyleSheet.
    my_styleSheet.onLoad = function(success:Boolean) {
      if (success) {
        StyleSheetTracer.display(this);
      } else {
        trace("Error loading style sheet "+url);
      }
    };
    // Start the loading operation.
    my_styleSheet.load(url);
  }
  static function display(my_styleSheet:StyleSheet):Void {
    var styleNames:Array = my_styleSheet.getStyleNames();
    if (!styleNames.length) {
      trace("This is an empty style sheet.");
    } else {
      for (var i = 0; i<styleNames.length; i++) {
        var styleName:String = styleNames[i];
        trace("Style "+styleName+":");
        var styleObject:Object = my_styleSheet.getStyle(styleName);
        for (var propName in styleObject) {
          var propValue = styleObject[propName];
          trace("\t"+propName+": "+propValue);
        }
        trace("");
```

```
        }
      }
    }
}
```

Créez un document CSS appelé styles.css, qui a deux styles appelés `.heading` et `.mainBody` définissant les propriétés pour `font-family`, `font-size` et `font-weight`. Entrez le code suivant dans le document CSS :

```
/~ In styles.css ~/
.heading {
font-family: Arial, Helvetica, sans-serif;
font-size: 24px;
font-weight: bold;
}
.mainBody {
font-family: Arial, Helvetica, sans-serif;
font-size: 12px;
font-weight: normal;
}
```

Enfin, dans un fichier FLA ou ActionScript, tapez le code ActionScript suivant pour charger la feuille de style externe, styles.css :

```
StyleSheetTracer.displayFromURL("styles.css");
```

Cet exemple donne les résultats suivants dans le panneau de sortie :

```
Style .heading:
fontWeight: bold
fontSize: 24px
fontFamily: Arial, Helvetica, sans-serif

Style .mainBody:
fontWeight: normal
fontSize: 12px
fontFamily: Arial, Helvetica, sans-serif
```

### Voir également

## getStyleNames (méthode StyleSheet.getStyleNames)

```
public getStyleNames() : Array
```

Renvoie un tableau qui contient les noms (sous formes de chaînes) de tous les styles enregistrés dans cette feuille de style.

**Disponibilité :** ActionScript 1.0 ; Flash Player 7

### Valeur renvoyée

`Array` - Tableau de noms de style (en tant que chaînes).

### Exemple

Cet exemple crée un nom d'objet StyleSheet `styleSheet` qui contient deux styles, `heading` et `bodyText`. Il appelle ensuite la méthode `getStyleNames()` de l'objet StyleSheet, place les résultats dans le tableau `names_array` et affiche le contenu du tableau dans le panneau de sortie.

```
import TextField.StyleSheet;
var my_styleSheet:StyleSheet = new StyleSheet();
my_styleSheet.setStyle("heading", {fontsize:'24px'});
my_styleSheet.setStyle("bodyText", {fontsize:'12px'});
var names_array:Array = my_styleSheet.getStyleNames();
trace(names_array.join("\n"));
```

Les informations suivantes apparaissent dans le panneau de sortie :

```
bodyText
heading
```

### Voir également

`getStyle (méthode StyleSheet.getStyle)`

## load (méthode StyleSheet.load)

```
public load(url:String) : Boolean
```

Procède au chargement du fichier CSS dans la StyleSheet. L'opération de chargement est asynchrone : utilisez le gestionnaire de rappel `onLoad()` pour déterminer si le chargement du fichier est terminé. Le fichier CSS doit résider dans le même domaine que le fichier SWF qui le charge.

**Disponibilité :** ActionScript 1.0 ; Flash Player 7

### Paramètres

*`url`*`:String` - URL du fichier CSS à charger. L'URL doit se trouver dans le même domaine que l'URL du fichier SWF.

### Valeur renvoyée

`Boolean` - `false` si aucun paramètre (`null`) n'est transmis ; `true` dans tous les autres cas. Utilisez le gestionnaire de rappel `onLoad()` pour vérifier si le chargement d'une StyleSheet a réussi.

### Exemple

Pour consulter un exemple de chargement asynchrone des feuilles de style avec ActionScript 2.0, consultez l'exemple pour `getStyle()`.

L'exemple suivant charge le fichier CSS appelé styles.css dans l'objet StyleSheet `my_styleSheet`. Lorsque le fichier a été chargé avec succès, l'objet StyleSheet est appliqué à un objet TextField appelé `news_txt`.

```
import TextField.StyleSheet;
this.createTextField("news_txt", 999, 10, 10, 320, 240);
news_txt.multiline = true;
news_txt.wordWrap = true;
news_txt.html = true;

var my_styleSheet:StyleSheet = new StyleSheet();
my_styleSheet.onLoad = function(success:Boolean) {
  if (success) {
  news_txt.styleSheet = my_styleSheet;
  news_txt.htmlText = "<p class=\"heading\">Heading goes here!</p>"
    + "<p class=\"mainBody\">Lorem ipsum dolor sit amet, consectetuer "
    + "adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet
  "
    + "dolore magna aliquam erat volutpat.</p>";
  }
};
my_styleSheet.load("styles.css");
```

Vous trouverez l'intégralité du code de styles.css dans l'exemple pour `getStyle()`.

### Voir également

`onLoad (gestionnaire StyleSheet.onLoad)`, `getStyle (méthode StyleSheet.getStyle)`

## onLoad (gestionnaire StyleSheet.onLoad)

```
onLoad = function(success:Boolean) {}
```

Appelé lorsqu'une opération `load()` est terminée. Si la StyleSheet a été chargée avec succès, le paramètre `success` est `true`. Si le document n'a pas été reçu, ou si une erreur est survenue au cours de la réception de la réponse provenant du serveur, le paramètre `success` est `false`.

**Disponibilité :** ActionScript 1.0 ; Flash Player 7

### Paramètres

*`success`*`:Boolean` - Valeur booléenne indiquant si le fichier CSS a été chargé avec succès (`true`) ou pas (`false`).

### Exemple

L'exemple suivant charge le fichier CSS appelé styles.css dans l'objet StyleSheet `my_styleSheet`. Lorsque le chargement du fichier s'est terminé avec succès, l'objet StyleSheet est appliqué à un objet TextField appelé `news_txt`.

```
import TextField.StyleSheet;
this.createTextField("news_txt", 999, 10, 10, 320, 240);
news_txt.multiline = true;
news_txt.wordWrap = true;
news_txt.html = true;

var my_styleSheet:StyleSheet = new StyleSheet();
my_styleSheet.onLoad = function(success:Boolean) {
  if (success) {
  news_txt.styleSheet = my_styleSheet;
  news_txt.htmlText = "<p class=\"heading\">Heading goes here!"
    + "</p><p class=\"mainBody\">Lorem ipsum dolor "
    + "sit amet, consectetuer adipiscing elit, sed diam nonummy "
    + "nibh euismod tincidunt ut laoreet dolore magna aliquam "
    + "erat volutpat.</p>";
  }
};
my_styleSheet.load("styles.css");
```

Vous trouverez l'intégralité du code de styles.css dans l'exemple pour `getStyle()`. Pour consulter un exemple de chargement asynchrone des feuilles de style avec ActionScript 2.0, consultez l'exemple pour `getStyle()`.

### Voir également

`load (méthode StyleSheet.load)`, `getStyle (méthode StyleSheet.getStyle)`

## parseCSS (méthode StyleSheet.parseCSS)

```
public parseCSS(cssText:String) : Boolean
```

Analyse le CSS dans `cssText` et charge la StyleSheet simultanément. Si un style figurant dans `cssText` est déjà présent dans styleSheet, les propriétés de styleSheet sont retenues et seules les propriétés de `cssText` sont ajoutées ou modifiées.

Pour développer la capacité d'analyse CSS d'origine, vous pouvez annuler cette méthode en créant une sous-classe de la classe StyleSheet.

**Disponibilité :** ActionScript 1.0 ; Flash Player 7

### Paramètres

*`cssText`*`:String` - Texte CSS à analyser.

### Valeur renvoyée

`Boolean` - Valeur booléenne indiquant si le texte a été analysé avec succès (`true`) ou pas (`false`).

### Exemple

L'exemple suivant analyse la CSS dans `css_str` Le script affiche le résultat de son analyse de la CSS, puis affiche la CSS analysée dans le panneau de sortie.

```
import TextField.StyleSheet;
var css_str:String = ".heading {font-family: Arial, Helvetica, sans-serif;
  font-size: 24px; font-weight: bold; }";
var my_styleSheet:StyleSheet = new StyleSheet();
if (my_styleSheet.parseCSS(css_str)) {
  trace("parsed successfully");
  dumpStyles(my_styleSheet);
} else {
  trace("unable to parse CSS");
}
//
function dumpStyles(styles:StyleSheet):Void {
  var styleNames_array:Array = styles.getStyleNames();
  for (var i = 0; i<styleNames_array.length; i++) {
    var styleName_str:String = styleNames_array[i];
    var styleObject:Object = styles.getStyle(styleName_str);
    trace(styleName_str);
    for (var prop in styleObject) {
      trace("\t"+prop+": "+styleObject[prop]);
    }
    trace("");
  }
}
```

## setStyle (méthode StyleSheet.setStyle)

```
public setStyle(name:String, style:Object) : Void
```

Ajoute un nouveau style avec le nom spécifié à l'objet feuille de style. Si le style nommé n'existe pas déjà dans la StyleSheet, il est ajouté. Si le style nommé n'existe pas déjà dans la StyleSheet, il est remplacé. Si le paramètre `style` est `null`, le style nommé est supprimé.

Flash Player crée une copie de l'objet style que vous transmettez à cette méthode.

Vous trouverez une liste des styles pris en charge dans le tableau de description de la classe StyleSheet.

**Disponibilité :** ActionScript 1.0 ; Flash Player 7

## Paramètres

*name*:String - Nom du style à ajouter à la StyleSheet.

*style*:Object - Objet qui décrit le style, ou null.

## Exemple

L'exemple suivant ajoute un style appelé emphasized à la StyleSheet myStyleSheet. Ce style comporte deux propriétés de style : color et fontWeight. L'objet style est défini avec l'opérateur {}.

```
myStyleSheet.setStyle("emphasized", {color:'#000000',fontWeight:'bold'});
```

Vous pouvez également créer un objet style à partir d'une occurrence de la classe Object, puis transmettre cet objet (styleObj) en tant que paramètre style comme il est indiqué dans l'exemple suivant :

```
import TextField.StyleSheet;
var my_styleSheet:StyleSheet = new StyleSheet();

var styleObj:Object = new Object();
styleObj.color = "#000000";
styleObj.fontWeight = "bold";
my_styleSheet.setStyle("emphasized", styleObj);
delete styleObj;

var styleNames_array:Array = my_styleSheet.getStyleNames();
for (var i=0;i<styleNames_array.length;i++) {
  var styleName:String = styleNames_array[i];
  var thisStyle:Object = my_styleSheet.getStyle(styleName);
  trace(styleName);
  for (var prop in thisStyle) {
    trace("\t"+prop+": "+thisStyle[prop]);
  }
  trace("");
}
```

Les informations suivantes apparaissent dans le panneau de sortie :

```
emphasized
fontWeight: bold
color: #000000
```

*Remarque :* Dans la mesure où Flash Player crée une copie de l'objet style transmis à setStyle(), la commande `delete styleObj` dans l'exemple de code réduit l'usage de la mémoire en supprimant l'objet style d'origine transmis à setStyle().

### Voir également

Opérateur {} (initialiseur d'objet), StyleSheet (TextField.StyleSheet)

## StyleSheet, constructeur

```
public StyleSheet()
```

Crée un objet StyleSheet.

**Disponibilité :** ActionScript 1.0 ; Flash Player 7

### Exemple

L'exemple suivant charge une feuille de style et présente les styles à charger dans le document. Ajoutez le code ActionScript suivant à votre fichier ActionScript ou FLA :

```
import TextField.StyleSheet;
var my_styleSheet:StyleSheet = new StyleSheet();
my_styleSheet.onLoad = function(success:Boolean) {
  if (success) {
    trace("Styles loaded:");
    var styles_array:Array = my_styleSheet.getStyleNames();
    trace(styles_array.join(newline));
  } else {
    trace("Error loading CSS");
  }
};
my_styleSheet.load("styles.css");
```

Le fichier styles.css contient deux styles appelés `.heading` et `.mainbody` pour afficher les informations suivantes dans le panneau de sortie :

```
Styles loaded:
.heading
.mainBody
```

Vous trouverez l'intégralité du code de styles.css dans l'exemple pour `getStyle()`.

### Voir également

getStyle (méthode StyleSheet.getStyle)

## transform (méthode StyleSheet.transform)

```
public transform(style:Object) : TextFormat
```

Développe la capacité d'analyse du fichier CSS. Les développeurs avancés peuvent annuler cette méthode en développant la classe StyleSheet.

**Disponibilité :** ActionScript 1.0 ; Flash Player 7

### Paramètres

*style*:Object - Objet qui décrit le style, contenant des règles de style en tant que propriétés de l'objet, ou null.

### Valeur renvoyée

TextFormat - Objet TextFormat contenant le résultat du mappage des règles CSS en propriétés de format de texte.

### Exemple

L'exemple suivant développe la méthode transform() :

```
import TextField.StyleSheet;
class AdvancedCSS extends StyleSheet {
  public function AdvancedCSS() {
    trace("AdvancedCSS instantiated");
  }

  public function transform(styleObject):TextFormat {
    trace("tranform called");
  }
}
```

# System

```
Object
  |
  +-System
```

public class *System*
extends Object

La classe System regroupe les propriétés liées à certaines opérations qui ont lieu sur l'ordinateur de l'utilisateur, telles qu'opérations au niveau des objets partagés, des paramètres locaux pour les caméras et les microphones, ainsi que de l'utilisation du Presse-papiers. Des propriétés et des méthodes supplémentaires suivantes figurent dans des classes spécifiques dans le package System : la classe capabilities, la classe security et la classe IME.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

### Voir également

capabilities (System.capabilities), security (System.security), IME (System.IME)

## Résumé des propriétés

| Modificateurs | Propriété | Description |
|---|---|---|
| `static` | `exactSettings:Boolean` | Valeur booléenne qui spécifie si les règles de concordance avec le superdomaine (`false`) ou le domaine exact (`true`) s'appliquent lorsque vous accédez aux paramètres locaux (tels que les autorisations d'accès à la caméra ou au microphone) ou aux données persistantes locales (objets partagés). |
| `static` | `useCodepage:Boolean` | Valeur booléenne qui indique à Flash Player s'il faut utiliser Unicode ou la page de code classique du système d'exploitation exécutant le lecteur pour interpréter des fichiers texte externes. |

*Propriétés héritées de la classe Object*

`constructeur (propriété Object.constructor), __proto__ (Object.__proto__ property), prototype (Object.prototype, propriété), __resolve (Object.__resolve, propriété)`

## Résumé des événements

| Evénement | Description |
|---|---|
| `onStatus = function(infoObject:Object) {}` | Gestionnaire d'événements : fournit un gestionnaire de super événements pour certains objets. |

## Résumé de la méthode

| Modificateurs | Signature | Description |
|---|---|---|
| `static` | `setClipboard(text:String) : Void` | Remplace le contenu du presse-papiers par une chaîne de texte spécifiée. |
| `static` | `showSettings([tabID:Number]) : Void` | Affiche le panneau Paramètres de Flash Player spécifié. |

*Méthodes héritées de la classe Object*

`addProperty (méthode Object.addProperty), hasOwnProperty (méthode Object.hasOwnProperty), isPropertyEnumerable (méthode Object.isPropertyEnumerable), isPrototypeOf (méthode Object.isPrototypeOf), registerClass (méthode Object.registerClass), toString (méthode Object.toString), unwatch (méthode Object.unwatch), valueOf (méthode Object.valueOf), watch (méthode Object.watch)`

# exactSettings (propriété System.exactSettings)

```
public static exactSettings : Boolean
```

Valeur booléenne qui spécifie si les règles de concordance avec le superdomaine (`false`) ou le domaine exact (`true`) s'appliquent lorsque vous accédez aux paramètres locaux (tels que les autorisations d'accès à la caméra ou au microphone) ou aux données persistantes locales (objets partagés). La valeur par défaut est `true` pour les fichiers publiés pour Flash Player 7 ou une version plus récente et `false` pour les fichiers publiés pour Flash Player 6.

Si cette valeur est `true` (vrai), les paramètres et les données d'un fichier SWF hébergé à ici.xyz.com sont stockés dans un répertoire ici.xyz.com, les paramètres et les données d'un fichier SWF hébergé à labas.xyz.com sont stockés dans un répertoire appelé labas.xyz.com, etc. Si cette valeur est `false`, les paramètres et données de fichiers SWF résidant sur ici.xyz.com, labas.xyz.com et xyz.com sont partagés et tous enregistrés sur xyz.com.

Si une partie de vos fichiers définit cette propriété sur `false` et les autres sur `true`, vous risquez d'obtenir des fichiers SWF dans différents sous-domaines qui partagent des paramètres et des données. Par exemple, si cette propriété est `false` dans un fichier SWF hébergé à l'adresse here.xyz.com et `true` dans un fichier SWF hébergé à l'adresse xyz.com, les deux fichiers utiliseront les mêmes paramètres et données, c'est-à-dire, ceux du répertoire xyz.com. Si le comportement n'est pas celui attendu, assurez-vous de définir cette propriété dans chaque fichier pour représenter correctement la manière dont vous souhaitez enregistrer les paramètres et les données.

Si vous voulez modifier la valeur par défaut de cette propriété, faites-le dans le premier cadre de votre document. Si vous voulez modifier la valeur par défaut de cette propriété, faites-le à proximité du début de votre script. La propriété ne peut pas être modifiée après une activité nécessitant un accès aux paramètres locaux telle que `System.showSettings()` ou `SharedObject.getLocal()`.

Si vous utilisez `loadMovie()`, `MovieClip.loadMovie()`, ou `MovieClipLoader.loadClip()` pour charger un fichier SWF dans un autre, tous les fichiers publiés pour Flash Player 7 partagent une seule valeur pour `System.exactSettings` et tous les fichiers publiés pour Flash Player 6 partagent une seule valeur pour `System.exactSettings`. Si vous utilisez `MovieClip.loadMovie()` ou `MovieClipLoader.loadClip()` pour charger un fichier SWF dans un autre, tous les fichiers partagent une seule valeur pour `System.exactSettings`. Par conséquent, si vous spécifiez une valeur pour cette propriété dans un fichier publié pour une version du Player particulière, vous devez le faire dans tous les fichiers à charger. Si vous chargez plusieurs fichiers, le paramètre spécifié dans le dernier fichier chargé remplace tout paramètre précédemment spécifié.

Généralement, vous devez trouver que la valeur par défaut de `System.exactSettings` est fine. Souvent, la seule condition requise est que, lorsqu'un fichier SWF enregistre un objet partagé dans une session, le même fichier SWF puisse récupérer le même objet partagé dans une session ultérieure. Cette situation sera toujours vraie, quelle que soit la valeur de `System.exactSettings`. Mais vous pouvez modifier `System.exactSettings` à partir de sa valeur par défaut de sorte qu'un fichier SWF publié pour Flash Player 7 ou une version ultérieure puisse récupérer les objets partagés créés à l'origine par un fichier SWF publié pour Flash Player 6. Etant donné que le lecteur a enregistré les objets partagés créés par le fichier SWF Flash Player 6 dans un dossier spécifique au superdomaine de ce fichier SWF, vous devez utiliser les règles du superdomaine pour la récupération de l'objet partagé dans votre fichier SWF Flash Player 7. Cette étape requiert la spécification de `System.exactSettings = false` dans votre fichier SFW Flash Player 7. Il est également possible d'avoir des fichiers SWF publiés pour Flash Player 6 et des fichiers SWF pour Flash Player 7 partageant les mêmes données d'objet partagé. Dans ce cas, choisissez simplement une valeur pour `System.exactSettings` (soit `true` soit `false`) et utilisez-la invariablement dans vos fichiers SWF Flash Player 6 et Flash Player 7.

**Disponibilité :** ActionScript 1.0 ; Flash Player 7

### Exemple

L'exemple suivant indique comment spécifier les règles de concordance du superdomaine :

### Voir également

`loadMovie (méthode MovieClip.loadMovie)`, `loadClip (méthode MovieClipLoader.loadClip)`, `getLocal (méthode SharedObject.getLocal)`, `exactSettings (propriété System.exactSettings)`

## onStatus (gestionnaire System.onStatus)

```
onStatus = function(infoObject:Object) {}
```

Gestionnaire d'événements : fournit un gestionnaire de super événements pour certains objets.

Les classes LocalConnection, NetStream, et SharedObject fournissent un gestionnaire d'événements onStatus qui utilise un objet information pour fournir des informations, des messages d'état ou d'erreur. Pour répondre à ce gestionnaire d'événements, vous devez créer une fonction permettant de traiter l'objet information et vous devez connaître le format et le contenu de l'objet information renvoyé.

En plus de ces méthodes onStatus spécifiques, Flash fournit également une super fonction appelée `System.onStatus`, qui sert de gestionnaire de messages d'erreur secondaire. Si une occurrence de la classe LocalConnection, NetStream, ou SharedObject transmet un objet information avec une propriété de niveau « error », mais si vous n'avez pas défini une fonction onStatus pour cette occurrence particulière, Flash utilise alors la fonction que vous définissez à la place pour `System.onStatus`.

Remarque : Les classes Camera et Microphone ont également des gestionnaires onStatus mais ne transmettent pas d'objets information avec un niveau de propriété « error ». Par conséquent, `System.onStatus` n'est pas appelé si vous ne spécifiez pas de fonction pour ces gestionnaires.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

## Paramètres

*`infoObject`*`:Object` - Paramètre défini en fonction du message d'état.

## Exemple

L'exemple suivant indique comment créer une fonction System.onStatus pour traiter les objets d'information lorsqu'une fonction onStatus, propre à une classe, n'existe pas :

```
// Create generic function
System.onStatus = function(genericError:Object){
  // Your script would do something more meaningful here
  trace("An error has occurred. Please try again.");
}
```

L'exemple suivant indique comment créer une fonction onStatus pour une occurrence de la classe NetStream :

```
// Create function for NetStream object

videoStream_ns.onStatus = function(infoObject:Object) {
  if (infoObject.code == "NetStream.Play.StreamNotFound") {
    trace("Could not find video file.");
  }
}
```

## Voir également

`onStatus (gestionnaire Camera.onStatus)`, `onStatus (gestionnaire LocalConnection.onStatus)`, `onStatus (gestionnaire Microphone.onStatus)`, `onStatus (gestionnaire NetStream.onStatus)`, `onStatus (gestionnaire SharedObject.onStatus)`

# setClipboard (méthode System.setClipboard)

```
public static setClipboard(text:String) : Void
```

Remplace le contenu du presse-papiers par une chaîne de texte spécifiée.

*Remarque :* Pour des raisons de sécurité, il est impossible de lire le contenu du système Clipboard. En d'autres termes, il n'existe pas de méthode `System.getClipboard()` correspondante.

**Disponibilité :** ActionScript 1.0 ; Flash Player 7

## Paramètres

*`text`*`:String` - Chaîne au format texte seul à placer dans le Presse-papiers du système, remplaçant son contenu actuel (si ce dernier contient déjà des données).

## Exemple

L'exemple suivant place la phrase `"Hello World"` dans le Presse-papiers système :

```
System.setClipboard("Hello world");
```

L'exemple suivant crée deux champs texte pendant l'exécution, appelés `in_txt` et `out_txt`. Lorsque vous sélectionnez du texte dans le champ `in_txt` vous pouvez cliquer sur `copy_btn` pour copier les données dans le Presse-papiers. Vous pouvez ensuite coller le texte dans le champ`out_txt`.

```
this.createTextField("in_txt", this.getNextHighestDepth(), 10, 10, 160,
  120);
in_txt.multiline = true;
in_txt.border = true;
in_txt.text = "lorum ipsum...";
this.createTextField("out_txt", this.getNextHighestDepth(), 10, 140, 160,
  120);
out_txt.multiline = true;
out_txt.border = true;
out_txt.type = "input";

copy_btn.onRelease = function() {
  System.setClipboard(in_txt.text);
  Selection.setFocus("out_txt");
};
```

Si votre fichier SWF comporte un composant de la version 2, utilisez la classe DepthManager des composants de la version 2 au lieu de la méthode `MovieClip.getNextHighestDepth()`, utilisée dans cet exemple.

# showSettings (méthode System.showSettings)

```
public static showSettings([tabID:Number]) : Void
```

Affiche le panneau Paramètres de Flash Player spécifié. Le panneau permet aux utilisateurs d'effectuer les actions suivantes :

- Autoriser ou refuser l'accès à la caméra et au microphone
- Spécifier l'espace disque disponible pour les objets partagés
- Sélectionner une caméra et un microphone par défaut
- Spécifier le gain du microphone et les paramètres de suppression d'écho

Par exemple, si votre application implique l'usage d'une caméra, vous pouvez demander à l'utilisateur de sélectionner Autoriser dans le panneau Paramètres de contrôle, puis émettre la commande `System.showSettings(0)`. (Vérifiez que la taille de votre Scène est au moins égale à 215 x 138 pixels )

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

## Paramètres

*`tabID`*`:Number` [facultatif] - Nombre qui spécifie le panneau Paramètres de Flash Player à afficher, comme il est indiqué dans le tableau suivant :

| **Valeur transmise au** `panel` | **Panneau Paramètres affiché** |
|---|---|
| Aucun (le paramètre est omis) ou une valeur non prise en charge | Le panneau qui était ouvert lorsque l'utilisateur a fermé le panneau Paramètres de Flash Player. |
| 0 | Confidentialité |
| 1 | Stockage local |
| 2 | Microphone |
| 3 | Caméra |

## Exemple

L'exemple suivant indique comment afficher le panneau Paramètres d'enregistrement local de Flash Player :

### Voir également

`get (méthode Camera.get)`, `obtenir (méthode Microphone.get)`, `getLocal (méthode SharedObject.getLocal)`

## useCodepage (propriété System.useCodepage)

```
public static useCodepage : Boolean
```

Valeur booléenne qui indique à Flash Player s'il faut utiliser Unicode ou la page de code classique du système d'exploitation exécutant le lecteur pour interpréter des fichiers texte externes. La valeur par défaut de `System.useCodepage` est false.

- Lorsque la propriété est définie sur false, Flash Player interprète les fichiers externes comme de l'Unicode. (Ces fichiers doivent être codés en Unicode lorsque vous les enregistrez.)
- Lorsque la propriété est définie sur true, Flash Player interprète les fichiers texte externes à l'aide de la page de code classique du système d'exploitation exécutant le lecteur.

Le texte que vous chargez comme un fichier externe (à l'aide des instructions `loadVariables()` ou `getURL()` statements, ou de la classe LoadVars ou XML) doit être codé en Unicode lorsque vous enregistrez le fichier texte afin que Flash Player le reconnaisse comme Unicode. Pour coder des fichiers externes comme Unicode, enregistrez les fichiers dans une application qui prend en charge l'Unicode, tel que Notepad sous Windows 2000.

Si vous chargez des fichiers externes qui ne sont pas codés en Unicode, vous devez définir `System.useCodepage` sur true. Ajoutez le code suivant sur la première ligne de code dans la première image du fichier SWF qui charge les données :

```
System.useCodepage = true;
```

Lorsque ce code est présent, Flash Player interprète du texte externe à l'aide de la page de code classique du système d'exploitation exécutant Flash Player. Ce code est généralement CP1252 pour un système d'exploitation Windows anglais et Shift-JIS pour un système d'exploitation japonais. Si vous définissez `System.useCodepage` sur true, Flash Player 6 et les versions ultérieures traitent le texte comme Flash Player 5. (Flash Player 5 traitait l'ensemble du texte comme s'il se trouvait dans la page de code classique du système d'exploitation exécutant le lecteur.)

Si vous définissez `System.useCodepage` sur true, souvenez-vous que la page de code classique du système d'exploitation exécutant le lecteur doit inclure les caractères utilisés dans votre fichier texte externe afin d'afficher le texte. Par exemple, si vous chargez un fichier texte externe contenant des caractères chinois, ceux-ci ne peuvent s'afficher sur un système utilisant la page de code CP1252 car celle-ci ne comprend pas les caractères chinois.

Pour que les utilisateurs de toutes les plates-formes puissent afficher les fichiers texte externes utilisés dans vos fichiers SWF, vous devez coder tous les fichiers texte externes en Unicode et conserver la propriété `System.useCodepage` définie sur false par défaut. Ainsi, Flash Player 6 et les versions ultérieures interprètent le texte comme de l'Unicode.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

# TextField

```
Object
  |
  +-TextField
```

`public dynamic class` ***TextField***
`extends Object`

La classe TextField permet de créer des zones d'affichage et d'entrée du texte. Tous les champs texte de saisie et dynamique dans un fichier SWF sont des occurrences de la classe TextField. Vous pouvez donner un nom d'occurrence à un champ texte dans l'inspecteur des propriétés, puis utiliser les méthodes et les propriétés de la classe TextField pour la modifier avec ActionScript. Les noms d'occurrence de TextField s'affichent dans l'explorateur d'animations et dans la boîte de dialogue Insérer un chemin cible du panneau Actions.

Pour créer dynamiquement un champ texte, vous ne devez pas utiliser l'opérateur`new`. Utilisez plutôt `MovieClip.createTextField()` à sa place.

Les méthodes de la classe TextField permettent de définir, sélectionner et manipuler du texte dans un champ texte dynamique ou de saisie que vous créez en cours de programmation ou à l'exécution.

ActionScript offre différentes manières de formater vos textes à l'exécution. La classe TextFormat permet de définir le formatage des caractères et des paragraphes pour les objets TextField. A partir de Flash Player 7, vous pouvez appliquer des styles CSS (feuilles de style en cascade) aux champs texte à l'aide de la propriété TextField.StyleSheet et de la classe StyleSheet. Vous pouvez utiliser le style CSS pour l'appliquer aux balises HTML intégrées, définir de nouvelles balises de format ou appliquer des styles. Vous pouvez directement affecter du texte au format HTML, pouvant éventuellement utiliser des styles CSS, à un champ texte. Dans Flash Player 7 et les versions ultérieures, le texte HTML que vous assignez à un champ texte peut contenir des supports intégrés (clips vidéo, fichiers SWF, JPEG, GIF et PNG). Le texte entoure le média intégré comme dans un document HTML dans un navigateur Web.

Flash Player prend en charge un sous-ensemble de balises HTML à utiliser pour formater le texte.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

## Voir également

`Object`, `createTextField (méthode MovieClip.createTextField)`

## Résumé des propriétés

| Modificateurs | Propriété | Description |
|---|---|---|
| | `_alpha:Number` | Définit ou extrait la valeur de transparence alpha du champ texte. |
| | `antiAliasType:String` | Le type d'anti-aliasing appliqué à cette instance de TextField. |
| | `autoSize:Object` | Commande le dimensionnement et l'alignement automatiques des champs texte. |
| | `background:Boolean` | Spécifie si le champ texte a un remplissage d'arrière-plan. |
| | `backgroundColor:Number` | Couleur de l'arrière-plan du champ texte. |
| | `border:Boolean` | Spécifie si le champ texte comporte une bordure. |
| | `borderColor:Number` | Couleur de la bordure du champ texte. |
| | `bottomScroll:Number` [lecture seule] | Entier (index de base un) qui indique la ligne la plus basse visible dans le champ texte. |
| | `condenseWhite:Boolean` | Valeur booléenne qui spécifie si les espaces blancs (espaces, sauts de ligne, etc.) dans un champ texte HTML doivent être supprimés lorsque le champ est restitué dans un navigateur. |
| | `embedFonts:Boolean` | Spécifie si le rendu doit utiliser des polices vectorielles incorporées. |
| | `filters:Array` | Un tableau indexé contenant tous les objets filtre associés au champ texte. |
| | `gridFitType:String` | Le type d'adaptation à la grille appliqué à cette occurrence de TextField. |
| | `_height:Number` | Hauteur du champ texte, en pixels. |
| | `_highquality:Number` | *Déconseillé* à partir de Flash Player 7. Cette propriété est déconseillée, préférez-lui `TextField._quality`. Spécifie le niveau d'anti-aliasing appliqué au fichier SWF actuel. |
| | `hscroll:Number` | Indique la position de défilement horizontal actuelle. |

| Modificateurs | Propriété | Description |
|---|---|---|
| | `html:Boolean` | Indicateur qui signale si le champ texte contient une représentation HTML. |
| | `htmlText:String` | Si le champ texte est un champ texte HTML, cette propriété contient la représentation HTML du contenu du champ texte. |
| | `length:Number` [lecture seule] | Indique le nombre de caractères d'un champ texte. |
| | `maxChars:Number` | Indique le nombre maximum de caractères qu'un champ texte peut contenir. |
| | `maxhscroll:Number` [lecture seule] | Indique la valeur maximale de `TextField.hscroll`. |
| | `maxscroll:Number` [lecture seule] | Indique la valeur maximale de `TextField.scroll`. |
| | `menu:ContextMenu` | Associe l'objet ContextMenu *`contextMenu`* au champ texte *`my_txt`*. |
| | `mouseWheelEnabled:Boolean` | Valeur booléenne qui indique si Flash Player doit automatiquement faire défiler des champs texte multiligne lorsque le pointeur de la souris clique sur un champ texte et l'utilisateur actionne la molette. |
| | `multiline:Boolean` | Indique si le champ texte est un champ texte multiligne. |
| | `_name:String` | Le nom de l'occurrence du champ texte. |
| | `_parent:MovieClip` | Référence au clip ou à l'objet contenant le champ texte ou l'objet actuel. |
| | `password:Boolean` | Indique si le champ texte est un champ texte de mot de passe. |
| | `_quality:String` | La qualité de rendu utilisée pour un fichier SWF. |
| | `restrict:String` | Indique le jeu de caractères qu'un utilisateur peut rentrer dans le champ texte. |
| | `_rotation:Number` | Rotation du champ texte, en degrés, à partir de son orientation d'origine. |
| | `scroll:Number` | La position verticale du texte dans un champ texte. |
| | `selectable:Boolean` | Valeur booléenne indiquant si le champ texte est sélectionnable. |
| | `sharpness:Number` | La netteté des bords du glyphe dans cette occurrence TextField. |

| Modificateurs | Propriété | Description |
|---|---|---|
| | `_soundbuftime:Number` | Le nombre de secondes pendant lequel les sons sont chargés en mémoire tampon avant d'être diffusés en continu. |
| | `styleSheet:StyleSheet` | Associe une feuille de style au champ texte. |
| | `tabEnabled:Boolean` | Spécifie si le champ texte est inclus dans l'ordre de tabulation automatique. |
| | `tabIndex:Number` | Permet de personnaliser l'ordre de tabulation des objets dans un fichier SWF. |
| | `_target:String` [lecture seule] | Le chemin cible de l'occurrence du champ texte. |
| | `text:String` | Indique le texte actuel dans le champ texte. |
| | `textColor:Number` | Indique la couleur du texte dans un champ texte. |
| | `textHeight:Number` | Indique la hauteur du texte. |
| | `textWidth:Number` | Indique la largeur du texte. |
| | `thickness:Number` | L'épaisseur des bords du glyphe dans cette occurrence TextField. |
| | `type:String` | Spécifie le type de champ texte. |
| | `_url:String` [lecture seule] | Récupère l'URL du fichier SWF qui a créé le champ texte. |
| | `variable:String` | Nom de la variable à laquelle le champ texte est associé. |
| | `_visible:Boolean` | Valeur booléenne indiquant si le champ texte *`my_txt`* est visible. |
| | `_width:Number` | Largeur du champ texte, en pixels. |
| | `wordWrap:Boolean` | Valeur booléenne indiquant si le champ texte comporte un retour à la ligne. |
| | `_x:Number` | Entier qui définit la coordonnée x d'un champ texte par rapport aux coordonnées locales du clip parent. |
| | `_xmouse:Number` [lecture seule] | Renvoie la coordonnée x de la position de la souris par rapport au champ texte. |
| | `_xscale:Number` | Détermine le redimensionnement horizontal du champ texte tel qu'il est appliqué à partir du point d'alignement du champ texte, exprimé en pourcentage. |

| Modificateurs | Propriété | Description |
|---|---|---|
| | `_y:Number` | Coordonnée y d'un champ texte par rapport aux coordonnées locales du clip parent. |
| | `_ymouse:Number` [lecture seule] | Indique la coordonnée y de la position de la souris par rapport au champ texte. |
| | `_yscale:Number` | Redimensionnement vertical du champ texte tel qu'il est appliqué à partir du point d'alignement du champ texte, exprimé en pourcentage. |

*Propriétés héritées de la classe Object*

`constructeur (propriété Object.constructor), __proto__ (Object.__proto__ property), prototype (Object.prototype, propriété), __resolve (Object.__resolve, propriété)`

## Résumé des événements

| Evénement | Description |
|---|---|
| `onChanged = function(changedField:TextField) {}` | Gestionnaire d'événements/écouteur : appelé lorsque le contenu d'un champ texte est modifié. |
| `onKillFocus = function(newFocus:Object) {}` | Appelé lorsqu'un champ texte perd le focus clavier. |
| `onScroller = function(scrolledField:TextField) {}` | Gestionnaire d'événements/écouteur : appelé lorsque l'une des propriétés de défilement du champ texte est modifiée. |
| `onSetFocus = function(oldFocus:Object) {}` | Appelé lorsqu'un champ texte reçoit le focus clavier. |

## Résumé de la méthode

| Modificateurs | Signature | Description |
|---|---|---|
| | `addListener(listener:Object) : Boolean` | Enregistre un objet pour recevoir les notifications d'événement TextField. |
| | `getDepth() : Number` | Renvoie la profondeur d'un champ texte. |
| `static` | `getFontList() : Array` | Renvoie les noms des polices sur le système hôte du lecteur sous forme de tableau. |
| | `getNewTextFormat() : TextFormat` | Renvoie un objet TextFormat contenant une copie de l'objet TextFormat du champ texte. |
| | `getTextFormat([beginIndex:Number], [endIndex:Number]) : TextFormat` | Renvoie un objet TextFormat pour un caractère, une plage de caractères ou l'ensemble d'un objet TextField. |
| | `removeListener(listener:Object) : Boolean` | Supprime un objet écouteur précédemment enregistré dans une occurrence de champ texte avec `TextField.addListener()`. |
| | `removeTextField() : Void` | Supprime le champ texte. |
| | `replaceSel(newText:String) : Void` | Remplace la sélection actuelle par le contenu du paramètre `newText`. |
| | `replaceText(beginIndex:Number, endIndex:Number, newText:String) : Void` | Remplace une plage de caractères, spécifiée par les paramètres `beginIndex` et `endIndex` dans le champ texte spécifié, par le contenu du paramètre `newText`. |
| | `setNewTextFormat(tf:TextFormat) : Void` | Définit le format par défaut du nouveau texte dans un champ texte. |
| | `setTextFormat([beginIndex:Number], [endIndex:Number], textFormat:TextFormat) : Void` | Applique la mise en forme du texte spécifié par le paramètre `textFormat` à une partie ou à l'ensemble du texte dans un champ texte. |

addProperty (méthode Object.addProperty), hasOwnProperty (méthode Object.hasOwnProperty), isPropertyEnumerable (méthode Object.isPropertyEnumerable), isPrototypeOf (méthode Object.isPrototypeOf), registerClass (méthode Object.registerClass), toString (méthode Object.toString), unwatch (méthode Object.unwatch), valueOf (méthode Object.valueOf), watch (méthode Object.watch)

# addListener (méthode TextField.addListener)

```
public addListener(listener:Object) : Boolean
```

Enregistre un objet pour recevoir les notifications d'événement TextField. L'objet reçoit les notifications d'événement lorsque les gestionnaires d'événements `onChanged` et `onScroller` sont appelés. Lorsqu'un champ texte est modifié ou défile, les gestionnaires d'événements `TextField.onChanged` et `TextField.onScroller` sont appelés, suivis des gestionnaires d'événements `onChanged` et `onScroller` de tous les objets enregistrés comme écouteurs. Plusieurs objets peuvent être enregistrés comme écouteurs.

Pour supprimer un objet listener d'un champ texte, appelez `TextField.removeListener()`.

Une référence à l'occurrence du champ texte est transmise en tant que paramètre aux gestionnaires `onScroller` et `onChanged` par la source de l'événement. Vous pouvez capturer ces données en plaçant un paramètre dans la méthode du gestionnaire d'événements. Par exemple, le code suivant utilise `txt` en tant que paramètre à transmettre au gestionnaire d'événements `onScroller`. Le paramètre est ensuite utilisé dans une instruction `trace` pour envoyer le nom de l'occurrence du champ texte vers le panneau de sortie.

```
my_txt.onScroller = function(textfield_txt:TextField) {
  trace(textfield_txt._name+" scrolled");
};
```

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

## Paramètres

*`listener`*`:Object` - Objet avec un gestionnaire d'événements `onChanged` ou `onScroller`.

## Valeur renvoyée

`Boolean` -

### Exemple

L'exemple suivant définit un gestionnaire `onChanged` pour le champ d'entrée `my_txt`. Il définit ensuite un nouvel objet écouteur,`txtListener`, et définit un gestionnaire `onChanged` pour cet objet. Ce gestionnaire sera appelé lorsque le champ texte `my_txt` est modifié. La ligne finale du code appelle `TextField.addListener` pour enregistrer l'objet écouteur `txtListener` avec le champ texte `my_txt` de façon à le notifier lorsque `my_txt` change.

```
this.createTextField("my_txt", this.getNextHighestDepth(), 10, 10, 100,
  22);
my_txt.border = true;
my_txt.type = "input";

my_txt.onChanged = function(textfield_txt:TextField) {
  trace(textfield_txt._name+" changed");
};
var txtListener:Object = new Object();
txtListener.onChanged = function(textfield_txt:TextField) {
  trace(textfield_txt._name+" changed and notified myListener");
};
my_txt.addListener(txtListener);
```

La méthode `MovieClip.getNextHighestDepth()` utilisée dans cet exemple requiert Flash Player 7 ou version ultérieure. Si votre fichier SWF comporte un composant de la version 2, utilisez la classe DepthManager des composants de la version 2 au lieu de la méthode `MovieClip.getNextHighestDepth()`.

### Voir également

`onChanged (gestionnaire TextField.onChanged)`, `onScroller (gestionnaire TextField.onScroller)`, `removeListener (méthode TextField.removeListener)`

## _alpha (propriété TextField._alpha)

```
public _alpha : Number
```

Définit ou extrait la valeur de transparence alpha du champ texte. Les valeurs valides sont comprises entre 0 (entièrement transparent) et 100 (entièrement opaque). La valeur par défaut est 100. Les valeurs de transparence ne sont pas prises en charge pour les champs texte qui utilisent les polices de périphérique. Vous devez utiliser des polices intégrées pour utiliser la propriété de transparence `_alpha` avec un champ texte.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

### Exemple

Le code suivant définit la propriété `_alpha` d'un champ texte appelé `my_txt` sur 20%. Créez un symbole de police dans la bibliothèque en sélectionnant Nouvelle police dans le menu d'options de la bibliothèque. Définit ensuite la liaison de la police sur `my font`. Définit la liaison pour un symbole de police sur `my font`. Ajoutez le code ActionScript suivant à votre fichier FLA ou AS :

```
var my_fmt:TextFormat = new TextFormat();
my_fmt.font = "my font";
// where 'my font' is the linkage name of a font in the Library
this.createTextField("my_txt", this.getNextHighestDepth(), 10, 10, 100,
  22);
my_txt.border = true;
my_txt.embedFonts = true;
my_txt.text = "Hello World";
my_txt.setTextFormat(my_fmt);
my_txt._alpha = 20;
```

La méthode `MovieClip.getNextHighestDepth()` utilisée dans cet exemple requiert Flash Player 7 ou version ultérieure. Si votre fichier SWF comporte un composant de la version 2, utilisez la classe DepthManager des composants de la version 2 au lieu de la méthode `MovieClip.getNextHighestDepth()`.

### Voir également

`_alpha (propriété Button._alpha)`, `_alpha (propriété MovieClip._alpha)`

## antiAliasType (propriété TextField.antiAliasType)

```
public antiAliasType : String
```

Le type d'anti-aliasing appliqué à cette instance de TextField. L'anti-aliasing avancé est disponible uniquement à partir de Flash Player 8. Vous ne pouvez contrôler cette définition que si la police est intégrée (avec la propriété `embedFonts` définie sur `true`). Pour Flash Player 8, le paramètre par défaut est `"advanced"`.

Pour définir les valeurs pour cette propriété, utilisez les valeurs de chaîne suivantes :

| Valeur de chaîne | Description |
|---|---|
| `"normal"` | Applique un anti-aliasing ordinaire au texte. Ceci correspond au type d'anti-aliasing utilisé par Flash Player 7 et ses versions précédentes. |
| `"advanced"` | Applique un anti-aliasing avancé qui rend le texte plus lisible. (Cette fonction est disponible à partir de Flash Player 8.) Anti-aliasing avancé permet d'obtenir une qualité du rendu des polices de petite taille. Cette option convient particulièrement aux applications comportant beaucoup de texte de petite taille. L'anti-aliasing avancé n'est pas recommandé pour les polices de plus de 48 points. |

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

## Exemple

Cet exemple crée deux champs texte et n'applique l'anti-aliasing avancé qu'au premier. Il suppose que vous avez une police intégrée à la librairie avec l'identifiant de liaison défini sur `"Times-12"`. Pour intégrer la police, procédez comme suit :

- Ouvrez votre Bibliothèque
- Cliquez sur le menu d'options de la Bibliothèque dans le coin supérieur droit de la Bibliothèque
- Sélectionnez « Nouvelle Police » dans la liste déroulante
- Nommez la police « Times-12 »
- Sélectionnez « Times New Roman » dans le menu déroulant

- Appuyez sur le bouton « OK »
- Cliquez du bouton droit sur la nouvelle police créée et sélectionnez « Liaison... »
- Cochez la case « Exporter pour ActionScript »
- Acceptez l'identifiant par défaut « Times-12 » en appuyant sur le bouton « OK »

```
var my_format:TextFormat = new TextFormat();
my_format.font = "Times-12";

var my_text1:TextField = this.createTextField("my_text1",
  this.getNextHighestDepth(), 10, 10, 300, 30);
my_text1.text = "This text uses advanced anti-aliasing.";
my_text1.antiAliasType = "advanced";
my_text1.border = true;
my_text1.embedFonts = true;
my_text1.setTextFormat(my_format);

var my_text2:TextField = this.createTextField("my_text2",
  this.getNextHighestDepth(), 10, 50, 300, 30);
my_text2.text = "This text uses normal anti-aliasing."
my_text2.antiAliasType = "normal";
my_text2.border = true;
my_text2.embedFonts = true;
my_text2.setTextFormat(my_format);
```

Si votre fichier SWF comporte un composant de la version 2, utilisez la classe DepthManager des composants de la version 2 au lieu de la méthode `MovieClip.getNextHighestDepth()`, utilisée dans cet exemple.

### Voir également

`TextRenderer (flash.text.TextRenderer)`, `gridFitType (propriété TextField.gridFitType)`, `thickness (propriété TextField.thickness)`, `sharpness (propriété TextField.sharpness)`

## autoSize (propriété TextField.autoSize)

```
public autoSize : Object
```

Commande le dimensionnement et l'alignement automatiques des champs texte. Les valeurs acceptables pour autoSize sont `"none"` (par défaut), `"left"`, `"right"` et `"center"`. Lorsque vous définissez la propriété `autoSize` , `true` est synonyme de `"left"` et `false` est synonyme de `"none"`.

Les valeurs de `autoSize` et `TextField.wordWrap` déterminent si un champ texte est agrandi ou réduit vers la gauche, la droite ou le bas. La valeur par défaut pour chacune de ces propriétés est `false`.

Si `autoSize` est défini sur `"none"` (valeur par défaut) ou `false`, il n'y a aucun redimensionnement.

Si `autoSize` est défini sur `"left"` ou `true`, le texte est alors traité comme du texte cadré à gauche, ce qui signifie que le côté gauche du texte reste fixe et tout redimensionnement d'un champ texte sur une seule ligne se fera à droite. Si le texte contient un saut de ligne (par exemple `"\n" or "\r")` le bas est alors également redimensionné pour s'adapter à la ligne suivante du texte. Si `wordWrap` est également défini sur `true`, seul le bas du champ texte est redimensionné et le côté droit reste fixe.

Si `autoSize` est défini sur `"right"`, le texte est alors traité comme du texte cadré à droite, ce qui signifie que le côté droit du texte reste fixe et tout redimensionnement d'un champ texte sur une seule ligne se fera à gauche. Si le texte contient un saut de ligne (par exemple `"\n" or "\r")` le bas est alors également redimensionné pour s'adapter à la ligne suivante du texte. Si `wordWrap` est également défini sur `true`, seul le bas du champ texte est redimensionné et le côté gauche reste fixe.

Si `autoSize` est défini sur `"center"`, le texte est traité comme du texte centré, ce qui signifie que tout redimensionnement d'un champ texte sur une seule ligne est uniformément réparti sur les côtés droit et gauche. Si le texte contient un saut de ligne (par exemple `"\n" or "\r")` le bas est alors également redimensionné pour s'adapter à la ligne suivante du texte. Si `wordWrap` est également défini sur `true`, seul le bas du champ texte est redimensionné et les côtés gauche et droit restent fixes.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

## Exemple

Utilisez le code suivant et entrez différentes valeurs pour `autoSize` de façon à voir comment le champ se redimensionne en fonction de ces valeurs. Un clic de souris pendant la lecture du fichier SWF remplace la chaîne `"short text"` de chaque champ par un texte plus long et appliquant différents paramètres pour `autoSize`.

```
this.createTextField("left_txt", 997, 10, 10, 70, 30);
this.createTextField("center_txt", 998, 10, 50, 70, 30);
this.createTextField("right_txt", 999, 10, 100, 70, 30);
this.createTextField("true_txt", 1000, 10, 150, 70, 30);
this.createTextField("false_txt", 1001, 10, 200, 70, 30);

left_txt.text = "short text";
left_txt.border = true;

center_txt.text = "short text";
center_txt.border = true;

right_txt.text = "short text";
```

```
right_txt.border = true;

true_txt.text = "short text";
true_txt.border = true;

false_txt.text = "short text";
false_txt.border = true;

// create a mouse listener object to detect mouse clicks
var myMouseListener:Object = new Object();
// define a function that executes when a user clicks the mouse
myMouseListener.onMouseDown = function() {
  left_txt.autoSize = "left";
  left_txt.text = "This is much longer text";
  center_txt.autoSize = "center";
  center_txt.text = "This is much longer text";
  right_txt.autoSize = "right";
  right_txt.text = "This is much longer text";
  true_txt.autoSize = true;
  true_txt.text = "This is much longer text";
  false_txt.autoSize = false;
  false_txt.text = "This is much longer text";
};
// register the listener object with the Mouse object
Mouse.addListener(myMouseListener);
```

# background (propriété TextField.background)

```
public background : Boolean
```

Spécifie si le champ texte a un remplissage d'arrière-plan. Si `true`, le champ texte a un remplissage d'arrière-plan. Si `false`, le champ texte n'a pas de remplissage d'arrière-plan.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

## Exemple

L'exemple suivant crée un champ texte avec une couleur d'arrière-plan qui s'active ou se désactive lorsque vous appuyez sur l'une des touches du clavier.

```
this.createTextField("my_txt", this.getNextHighestDepth(), 10, 10, 320,
  240);
my_txt.border = true;
my_txt.text = "Lorum ipsum";
my_txt.backgroundColor = 0xFF0000;

var keyListener:Object = new Object();
keyListener.onKeyDown = function() {
  my_txt.background = !my_txt.background;
};
```

```
Key.addListener(keyListener);
```

La méthode `MovieClip.getNextHighestDepth()` utilisée dans cet exemple requiert Flash Player 7 ou version ultérieure. Si votre fichier SWF comporte un composant de la version 2, utilisez la classe DepthManager des composants de la version 2 au lieu de la méthode `MovieClip.getNextHighestDepth()`.

# backgroundColor (propriété TextField.backgroundColor)

```
public backgroundColor : Number
```

Couleur de l'arrière-plan du champ texte. La valeur par défaut est `0xFFFFFF` (blanc). Cette propriété peut être récupérée ou définie, même s'il n'y a actuellement aucun arrière-plan, mais la couleur n'est visible que si le champ texte a une bordure.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

## Exemple

Consultez l'exemple relatif à `TextField.background`.

## Voir également

`background (propriété TextField.background)`

# border (propriété TextField.border)

```
public border : Boolean
```

Spécifie si le champ texte comporte une bordure. Si `true`, le champ texte comporte une bordure. Si `false`, le champ texte ne comporte pas de bordure.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

## Exemple

L'exemple suivant crée un champ texte appelé `my_txt`, définit la propriété border sur `true` et affiche du texte dans ce champ.

```
this.createTextField("my_txt", this.getNextHighestDepth(), 10, 10, 320,
  240);
my_txt.border = true;
my_txt.text = "Lorum ipsum";
```

La méthode `MovieClip.getNextHighestDepth()` utilisée dans cet exemple requiert Flash Player 7 ou version ultérieure. Si votre fichier SWF comporte un composant de la version 2, utilisez la classe DepthManager des composants de la version 2 au lieu de la méthode `MovieClip.getNextHighestDepth()`.

## borderColor (propriété TextField.borderColor)

```
public borderColor : Number
```

Couleur de la bordure du champ texte. La valeur par défaut est `0x000000` (noir). Cette propriété peut être récupérée ou définie, même s'il n'y a actuellement pas de bordure.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

### Exemple

L'exemple suivant crée un champ texte appelé `my_txt`, définit la propriété border sur `true` et affiche du texte dans ce champ.

```
this.createTextField("my_txt", this.getNextHighestDepth(), 10, 10, 320,
  240);
my_txt.border = true;
my_txt.borderColor = 0x00FF00;
my_txt.text = "Lorum ipsum";
```

La méthode `MovieClip.getNextHighestDepth()` utilisée dans cet exemple requiert Flash Player 7 ou version ultérieure. Si votre fichier SWF comporte un composant de la version 2, utilisez la classe DepthManager des composants de la version 2 au lieu de la méthode `MovieClip.getNextHighestDepth()`.

### Voir également

`border (propriété TextField.border)`

## bottomScroll (propriété TextField.bottomScroll)

```
public bottomScroll : Number [lecture seule]
```

Entier (index de base un) qui indique la ligne la plus basse visible dans le champ texte. Considérez le champ texte comme une fenêtre sur un bloc de texte. La propriété `TextField.scroll` est l'index basé sur un de la ligne la plus haute visible dans la fenêtre.

L'ensemble du texte entre les lignes `TextField.scroll` et `TextField.bottomScroll` est actuellement visible dans le champ texte.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

### Exemple

L'exemple suivant crée un champ texte et y insère du texte. Vous devez insérer un bouton (avec le nom d'occurrence « my_btn »). Lorsque vous cliquez dessus, les propriétés `scroll` et `bottomScroll` du champ texte font l'objet d'une instruction trace pour le champ `comment_txt`.

```
this.createTextField("comment_txt", this.getNextHighestDepth(), 0, 0, 160,
  120);
comment_txt.html = true;
comment_txt.selectable = true;
comment_txt.multiline = true;
comment_txt.wordWrap = true;
comment_txt.htmlText = "<b>What is hexadecimal?</b><br>"
  + "The hexadecimal color system uses six digits to represent color
  values. "
  + "Each digit has sixteen possible values or characters. The characters
  range"
  + " from 0 to 9 and then A to F. Black is represented by (#000000) and
  white, "
  + "at the opposite end of the color system, is (#FFFFFF).";
my_btn.onRelease = function() {
  trace("scroll: "+comment_txt.scroll);
  trace("bottomScroll: "+comment_txt.bottomScroll);
};
```

La méthode `MovieClip.getNextHighestDepth()` utilisée dans cet exemple requiert Flash Player 7 ou version ultérieure. Si votre fichier SWF comporte un composant de la version 2, utilisez la classe DepthManager des composants de la version 2 au lieu de la méthode `MovieClip.getNextHighestDepth()`.

## condenseWhite (propriété TextField.condenseWhite)

```
public condenseWhite : Boolean
```

Valeur booléenne qui spécifie si les espaces blancs (espaces, sauts de ligne, etc.) dans un champ texte HTML doivent être supprimés lorsque le champ est restitué dans un navigateur. La valeur par défaut est `false`.

Si vous définissez cette valeur sur `true`, vous devez utiliser les commandes HTML classiques telles que `<BR>` et `<P>` pour placer des sauts de ligne dans le champ texte.

Si la propriété `.html` du champ texte est `false`, cette propriété est ignorée.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

### Exemple

L'exemple suivant crée deux champs texte pendant l'exécution, appelés `first_txt` et `second_txt`. L'espace blanc est supprimé du deuxième champ texte. Ajoutez le code ActionScript suivant à votre fichier FLA ou AS :

```
var my_str:String = "Hello\tWorld\nHow are you?\t\t\tEnd";

this.createTextField("first_txt", this.getNextHighestDepth(), 10, 10, 160,
  120);
first_txt.html = true;
first_txt.multiline = true;
first_txt.wordWrap = true;
first_txt.condenseWhite = false;
first_txt.border = true;
first_txt.htmlText = my_str;

this.createTextField("second_txt", this.getNextHighestDepth(), 180, 10,
  160, 120);
second_txt.html = true;
second_txt.multiline = true;
second_txt.wordWrap = true;
second_txt.condenseWhite = true;
second_txt.border = true;
second_txt.htmlText = my_str;
```

La méthode `MovieClip.getNextHighestDepth()` utilisée dans cet exemple requiert Flash Player 7 ou version ultérieure. Si votre fichier SWF comporte un composant de la version 2, utilisez la classe DepthManager des composants de la version 2 au lieu de la méthode `MovieClip.getNextHighestDepth()`.

### Voir également

`html (propriété TextField.html)`

## embedFonts (propriété TextField.embedFonts)

```
public embedFonts : Boolean
```

Spécifie si le rendu doit utiliser des polices vectorielles incorporées. Valeur booléenne qui, lorsqu'elle est définie sur `true`, effectue le rendu du champ texte à l'aide des polices vectorielles intégrées. Si `false`, elle restitue le champ texte à l'aide de polices de périphérique.

Si vous définissez `embedFonts` sur `true` pour un champ texte, vous devez spécifier la police du texte par l'intermédiaire de la propriété `font` d'un objet `TextFormat` appliqué au champ texte. Si la police spécifiée n'existe *pas* dans la bibliothèque (avec le nom d'occurrence de liaison correspondant), le texte ne s'affiche pas.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

### Exemple

Dans cet exemple, vous devez créer un champ texte dynamique appelé `my_txt`, puis utiliser le code ActionScript suivant pour incorporer des polices et faire pivoter le champ texte. La référence à `my font` renvoie à un symbole de police dans la bibliothèque, avec une liaison définie sur `my font`. L'exemple suivant suppose que vous disposez d'un symbole Font dans la bibliothèque appelé `my font`, avec des propriétés de liaison définies de la façon suivante : l'identifiant défini sur `my font` et Export pour ActionScript et Export dans la première image sélectionnée.

```
var my_fmt:TextFormat = new TextFormat();
my_fmt.font = "my font";

this.createTextField("my_txt", this.getNextHighestDepth(), 10, 10, 160,
  120);
my_txt.wordWrap = true;
my_txt.embedFonts = true;
my_txt.text = "Hello world";
my_txt.setTextFormat(my_fmt);
my_txt._rotation = 45;
```

La méthode `MovieClip.getNextHighestDepth()` utilisée dans cet exemple requiert Flash Player 7 ou version ultérieure. Si votre fichier SWF comporte un composant de la version 2, utilisez la classe DepthManager des composants de la version 2 au lieu de la méthode `MovieClip.getNextHighestDepth()`.

## filters (propriété TextField.filters)

```
public filters : Array
```

Un tableau indexé contenant tous les objets filtre associés au champ texte. Le package flash.filters contient plusieurs classes qui définissent des filtres spécifiques.

Ces filtres peuvent s'appliquer dans l'outil de programmation de Flash pendant la phase de conception ou d'exécution du code ActionScript. Pour appliquer un filtre avec ActionScript, vous devez créer une copie temporaire de l'intégralité du tableau `TextField.filters` modifier le tableau temporaire, puis reporter les valeurs de ce tableau temporaire dans le tableau `TextField.filters`. Vous ne pouvez pas appliquer directement un nouvel objet filtre au tableau `TextField.filters`. Le code suivant n'a aucun effet sur le clip cible, appelé `myTextField`:

```
myTextField.filters[0].push(myDropShadow);
```

Pour ajouter un filtre avec ActionScript, vous devez procéder de la façon décrite ci-dessous (supposez que le clip cible s'appelle `myTextField`):

- Créez un objet filtre avec la fonction constructeur de la classe de filtre retenue.

- Affectez la valeur du tableau `myTextField.filters` à un tableau temporaire, par exemple à un tableau intitulé `myFilters`.
- Ajoutez le nouvel objet filtre au tableau temporaire, `myFilters`.
- Affectez la valeur du tableau temporaire au tableau `myTextField.filters`.

Si le tableau `filters` est vide, il n'est pas nécessaire d'utiliser un tableau temporaire. Par contre, vous pouvez affecter directement un littéral de tableau contenant un ou plusieurs des objets filtres que vous avez créés.

Pour modifier un objet filtre existant, que ce dernier ait été créé pendant la phase de conception ou d'exécution, vous devez appliquer la technique de modification d'une copie du tableau `filters` de la façon suivante :

- Affectez la valeur du tableau `myTextField.filters` à un tableau temporaire, par exemple à un tableau intitulé `myFilters`.
- Modifiez la propriété avec le tableau temporaire `myFilters`. Par exemple, si vous souhaitez définir la propriété `quality` du premier filtre du tableau, vous pouvez utiliser le code suivant : `myList[0].quality = 1;`
- Affectez la valeur du tableau temporaire au tableau `myTextField.filters`.

Pour supprimer les filtres pour un champ texte, définissez `filters` sur un tableau vide (`[]`).

Si vous utilisez un tableau `filters` contenant plusieurs filtres et devez suivre le type de filtre affecté à chaque index de tableau, vous pouvez conserver votre propre tableau `filters` et utiliser une structure de données distincte pour suivre le type de filtre associé à chaque index de tableau. Il n'existe aucune méthode simple permettant de déterminer le type de filtre associé à chaque index de tableau `filters`.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

## Exemple

L'exemple suivant ajoute un filtre d'ombre portée à un clip appelé `myTextField`.

```
var myDropFilter = new flash.filters.DropShadowFilter();
var myFilters:Array = myTextField.filters;
myFilters.push(myDropFilter);
myTextField.filters = myFilters;
```

L'exemple suivant donne au paramètre `quality` du premier filtre du tableau la valeur 15 (cet exemple ne peut fonctionner que si au moins un objet filtre a été associé au champ texte `myTextField`).

```
var myList:Array = myTextField.filters;
myList[0].quality = 15;
myTextField.filters = myList;
```

### Voir également

## getDepth (méthode TextField.getDepth)

```
public getDepth() : Number
```

Renvoie la profondeur d'un champ texte.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

### Valeur renvoyée

`Number` - Entier qui représente la profondeur du champ texte.

### Exemple

L'exemple suivant porte sur les champs texte résidant à différentes profondeurs. Créez un champ texte dynamique sur la Scène et ajoutez le code ActionScript suivant à votre fichier FLA ou AS. Le code crée de façon dynamique deux champs texte lors de l'exécution et renvoie leur profondeur.

```
this.createTextField("first_mc", this.getNextHighestDepth(), 10, 10, 100,
  22);
this.createTextField("second_mc", this.getNextHighestDepth(), 10, 10, 100,
  22);
for (var prop in this) {
  if (this[prop] instanceof TextField) {
  var this_txt:TextField = this[prop];
  trace(this_txt._name+" is a TextField at depth: "+this_txt.getDepth());
  }
}
```

La méthode `MovieClip.getNextHighestDepth()` utilisée dans cet exemple requiert Flash Player 7 ou version ultérieure. Si votre fichier SWF comporte un composant de la version 2, utilisez la classe DepthManager des composants de la version 2 au lieu de la méthode `MovieClip.getNextHighestDepth()`.

## getFontList (méthode TextField.getFontList)

```
public static getFontList() : Array
```

Renvoie les noms des polices sur le système hôte du lecteur sous forme de tableau. (Cette méthode ne renvoie pas les noms de toutes les polices dans les fichiers SWF chargés.) Les noms sont du type `String`. Il s'agit d'une méthode statique de la classe globale TextField. Vous ne pouvez pas spécifier d'occurrence de champ texte lorsque vous appelez cette méthode.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

### Valeur renvoyée

`Array` - Tableau de noms de polices.

### Exemple

Le code suivant affiche une liste de polices renvoyées par `getFontList()` :

```
var font_array:Array = TextField.getFontList();
font_array.sort();
trace("You have "+font_array.length+" fonts currently installed");
trace("---------------------------------------");
for (var i = 0; i<font_array.length; i++) {
  trace("Font #"+(i+1)+":\t"+font_array[i]);
}
```

## getNewTextFormat (méthode TextField.getNewTextFormat)

```
public getNewTextFormat() : TextFormat
```

Renvoie un objet TextFormat contenant une copie de l'objet TextFormat du champ texte. L'objet TextFormat est le format que reçoit le nouveau texte inséré, tel que le texte inséré avec la méthode `replaceSel()` ou le texte entré par un utilisateur. Lorsque `getNewTextFormat()` est appelé, toutes les propriétés de l'objet TextFormat renvoyé sont définies. Aucune propriété n'est `null`.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

### Valeur renvoyée

`TextFormat` - Objet TextFormat.

### Exemple

L'exemple suivant affiche l'objet format de texte du champ texte spécifié (`my_txt`).

```
this.createTextField("my_txt", this.getNextHighestDepth(), 10, 10, 160,
  120);
var my_fmt:TextFormat = my_txt.getNewTextFormat();
trace("TextFormat has the following properties:");
for (var prop in my_fmt) {
  trace(prop+": "+my_fmt[prop]);
}
```

La méthode `MovieClip.getNextHighestDepth()` utilisée dans cet exemple requiert Flash Player 7 ou version ultérieure. Si votre fichier SWF comporte un composant de la version 2, utilisez la classe DepthManager des composants de la version 2 au lieu de la méthode `MovieClip.getNextHighestDepth()`.

# getTextFormat (méthode TextField.getTextFormat)

`public getTextFormat([beginIndex:Number], [endIndex:Number]) : TextFormat`

Renvoie un objet TextFormat pour un caractère, une plage de caractères ou l'ensemble d'un objet TextField.

Le tableau suivant décrit trois utilisations possibles :

| Utilisation | Description |
|---|---|
| `my_textField.getTextFormat()` | Renvoie un objet TextFormat contenant des informations de mise en forme pour l'ensemble du texte d'un champ texte. Seules les propriétés communes à l'ensemble du texte d'un champ texte sont définies dans l'objet TextFormat obtenu. Toute propriété qui est *mixte*, ce qui signifie qu'elle a différentes valeurs à différents endroits du texte, a la valeur `null`. |
| `my_textField.getTextFormat(beginIndex:Number)` | Renvoie un objet TextFormat contenant une copie du format de texte du champ texte sur la position `beginIndex`. |
| `my_textField.getTextFormat(beginIndex:Number,endIndex:Number)` | Renvoie un objet TextFormat contenant des informations de mise en forme pour la plage de texte de `beginIndex` à `endIndex`. Seules les propriétés communes à l'ensemble du texte de la plage spécifiée sont définies dans l'objet TextFormat obtenu. Toute propriété qui est mixte (c.-à-d. a différentes valeurs à différents endroits de la plage) a sa valeur définie sur `null`. |

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

## Paramètres

*`beginIndex`*`:Number` [facultatif] - Entier qui spécifie un caractère dans une chaîne. Si vous ne spécifiez pas `beginIndex` et `endIndex`, l'objet TextFormat renvoyé est pour l'ensemble du TextField.

*`endIndex`*`:Number` [facultatif] - Entier qui spécifie la position finale d'une plage de texte. Si vous spécifiez `beginIndex` mais ne spécifiez pas `endIndex`, le TextFormat renvoyé est pour le seul caractère spécifié par `beginIndex`.

## Valeur renvoyée

`TextFormat` - Objet TextFormat qui représente les propriétés de mise en forme du texte spécifié.

### Exemple

Le code ActionScript suivant suit l'ensemble des informations de formatage d'un champ texte qui est créé pendant l'exécution.

```
this.createTextField("dyn_txt", this.getNextHighestDepth(), 0, 0, 100,
  200);
dyn_txt.text = "Frank";
dyn_txt.setTextFormat(new TextFormat());
var my_fmt:TextFormat = dyn_txt.getTextFormat();
for (var prop in my_fmt) {
  trace(prop+": "+my_fmt[prop]);
}
```

La méthode `MovieClip.getNextHighestDepth()` utilisée dans cet exemple requiert Flash Player 7 ou version ultérieure. Si votre fichier SWF comporte un composant de la version 2, utilisez la classe DepthManager des composants de la version 2 au lieu de la méthode `MovieClip.getNextHighestDepth()`.

### Voir également

`getNewTextFormat (méthode TextField.getNewTextFormat)`, `setNewTextFormat (méthode TextField.setNewTextFormat)`, `setTextFormat (méthode TextField.setTextFormat)`

## gridFitType (propriété TextField.gridFitType)

```
public gridFitType : String
```

Le type d'adaptation à la grille appliqué à cette occurrence de TextField. Cette propriété ne s'applique que si la propriété `antiAliasType` du champ texte est définie sur `"advanced"`.

Pour la propriété `gridFitType`, vous pouvez utiliser les valeurs de chaînes suivantes :

| Valeur de chaîne | Description |
|---|---|
| `"none"` | Ne spécifie pas d'adaptation à la grille. Les lignes horizontales et verticales des glyphes ne sont pas alignées sur la grille de pixels. Ce paramètre est généralement retenu pour les animations ou les grandes polices. |

| Valeur de chaîne | Description |
|---|---|
| `"pixel"` | Spécifie que les lignes horizontales et verticales fortes sont adaptées à la grille de pixels. Ce paramètre convient uniquement aux champs texte alignés à gauche. Pour utiliser ce paramètre, la propriété `antiAliasType` du champ texte doit être définie sur `"advanced"`. Ce paramètre rend généralement le texte justifié à gauche plus lisible. |
| `"subpixel"` | Spécifie que les lignes horizontales et verticales fortes sont adaptées à la grille de sous-pixels sur les écrans à cristaux liquides. Pour utiliser ce paramètre, la propriété `antiAliasType` du champ texte doit être définie sur `"advanced"`. Ce paramètre `"subpixel"` est généralement préférable pour le texte dynamique aligné à droite ou centré et permet parfois d'établir un bon compromis entre la qualité d'animation et la qualité du texte. |

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

## Exemple

Cet exemple montre trois champs texte utilisant des paramètres `gridFitType` différents. Il suppose que vous avez une police intégrée à la librairie avec l'identifiant de liaison défini sur `"Times-12"`. Pour intégrer la police, procédez comme suit :

- Ouvrez votre Bibliothèque
- Cliquez sur le menu d'options de la Bibliothèque dans le coin supérieur droit de la Bibliothèque
- Sélectionnez « Nouvelle Police » dans la liste déroulante
- Nommez la police « Times-12 »
- Sélectionnez « Times New Roman » dans le menu déroulant
- Appuyez sur le bouton « OK »
- Cliquez du bouton droit sur la nouvelle police créée et sélectionnez « Liaison... »
- Cochez la case « Exporter pour ActionScript »
- Acceptez l'identifiant par défaut « Times-12 » en appuyant sur le bouton « OK »

```
var my_format:TextFormat = new TextFormat();
my_format.font = "Times-12";

var my_text1:TextField = this.createTextField("my_text1",
  this.getNextHighestDepth(), 9.5, 10, 400, 100);
my_text1.text = "this.gridFitType = none";
my_text1.embedFonts = true;
```

```
my_text1.antiAliasType = "advanced";
my_text1.gridFitType = "none";
my_text1.setTextFormat(my_format);

var my_text2:TextField = this.createTextField("my_text2",
  this.getNextHighestDepth(), 9.5, 40, 400, 100);
my_text2.text = "this.gridFitType = advanced";
my_text2.embedFonts = true;
my_text2.antiAliasType = "advanced";
my_text2.gridFitType = "pixel";
my_text2.setTextFormat(my_format);

var my_text3:TextField = this.createTextField("my_text3",
  this.getNextHighestDepth(), 9.5, 70, 400, 100);
my_text3.text = "this.gridFitType = subpixel";
my_text3.embedFonts = true;
my_text3.antiAliasType = "advanced";
my_text3.gridFitType = "subpixel";
my_text3.setTextFormat(my_format);
```

Si votre fichier SWF comporte un composant de la version 2, utilisez la classe DepthManager des composants de la version 2 au lieu de la méthode `MovieClip.getNextHighestDepth()`, utilisée dans cet exemple.

### Voir également

`TextRenderer (flash.text.TextRenderer)`, `antiAliasType (propriété TextField.antiAliasType)`, `sharpness (propriété TextField.sharpness)`

## _height (propriété TextField._height)

```
public _height : Number
```

Hauteur du champ texte, en pixels.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

### Exemple

L'exemple de code suivant définit la hauteur et la largeur d'un champ texte :

```
my_txt._width = 200;
my_txt._height = 200;
```

## _height (propriété TextField._height)

```
public _highquality : Number
```

*Déconseillé* à partir de Flash Player 7. Cette propriété est déconseillée, préférez-lui `TextField._quality`.

Spécifie le niveau d'anti-aliasing appliqué au fichier SWF actuel. Spécifiez 2 (meilleure qualité) pour bénéficier de la meilleure qualité possible et activer le lissage de façon permanente. Spécifiez 1 (haute qualité) pour procéder à l'anti-aliasing ; ceci permet de lisser les bitmaps si le fichier SWF ne contient pas d'animation et constitue la valeur par défaut. Spécifiez 0 (faible qualité) pour empêcher l'anti-aliasing.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

### Voir également

`_quality (propriété TextField._quality)`

## hscroll (propriété TextField.hscroll)

```
public hscroll : Number
```

Indique la position de défilement horizontal actuelle. Si la propriété `hscroll` est 0, le texte ne défile pas horizontalement.

Les unités du défilement horizontal sont les pixels, alors que les unités du défilement vertical sont les lignes. Le défilement horizontal est mesuré en pixels étant donné que la plupart des polices que vous utilisez généralement sont proportionnellement espacées, c'est-à-dire que les caractères peuvent avoir différentes largeurs. Flash propose un défilement vertical par ligne étant donné que les utilisateurs souhaitent que l'ensemble de la ligne de texte soit visible et non une partie de la ligne seulement. Même s'il existe plusieurs polices sur une ligne, la hauteur de la ligne s'adapte à la plus grande police utilisée.

*Remarque :* La propriété hscroll se base sur zéro et non sur un, comme c'est le cas de la propriété de défilement vertical `TextField.scroll`.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

### Exemple

L'exemple suivant fait défiler le champ texte `my_txt` à l'horizontale à l'aide de deux boutons appelés `scrollLeft_btn` et `scrollRight_btn`. Le montant de défilement s'affiche dans un champ texte appelé `scroll_txt`. Ajoutez le code ActionScript suivant à votre fichier FLA ou AS :

```
this.createTextField("scroll_txt", this.getNextHighestDepth(), 10, 10, 160,
  20);
this.createTextField("my_txt", this.getNextHighestDepth(), 10, 30, 160,
  22);
my_txt.border = true;
my_txt.multiline = false;
my_txt.wordWrap = false;
my_txt.text = "Lorem ipsum dolor sit amet, consectetuer adipiscing...";
```

```
scrollLeft_btn.onRelease = function() {
  my_txt.hscroll -= 10;
  scroll_txt.text = my_txt.hscroll+" of "+my_txt.maxhscroll;
};
scrollRight_btn.onRelease = function() {
  my_txt.hscroll += 10;
  scroll_txt.text = my_txt.hscroll+" of "+my_txt.maxhscroll;
};
```

La méthode `MovieClip.getNextHighestDepth()` utilisée dans cet exemple requiert Flash Player 7 ou version ultérieure. Si votre fichier SWF comporte un composant de la version 2, utilisez la classe DepthManager des composants de la version 2 au lieu de la méthode `MovieClip.getNextHighestDepth()`.

### Voir également

`maxhscroll (propriété TextField.maxhscroll)`, `scroll (propriété TextField.scroll)`

## html (propriété TextField.html)

```
public html : Boolean
```

Indicateur qui signale si le champ texte contient une représentation HTML. Si la propriété `html` est `true`, le champ texte est un champ texte HTML. Si `html` est `false`, le champ texte n'est pas un champ texte HTML.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

### Exemple

L'exemple suivant crée un champ texte qui définit la propriété `html` sur `true`. Le texte au format HTML s'affiche dans le champ texte.

```
this.createTextField("my_txt", this.getNextHighestDepth(), 10, 10, 160,
  22);
my_txt.html = true;
my_txt.htmlText = "<b> this is bold text </b>";
```

La méthode `MovieClip.getNextHighestDepth()` utilisée dans cet exemple requiert Flash Player 7 ou version ultérieure. Si votre fichier SWF comporte un composant de la version 2, utilisez la classe DepthManager des composants de la version 2 au lieu de la méthode `MovieClip.getNextHighestDepth()`.

### Voir également

`htmlText (propriété TextField.htmlText)`

# htmlText (propriété TextField.htmlText)

```
public htmlText : String
```

Si le champ texte est un champ texte HTML, cette propriété contient la représentation HTML du contenu du champ texte. Si le champ texte n'est pas un champ texte HTML, son comportement est identique à la propriété du `text`. Vous pouvez indiquer qu'un champ texte est un champ texte HTML dans l'inspecteur des propriétés ou en définissant la propriété `html` du champ texte sur `true`.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

## Exemple

L'exemple suivant crée un champ texte qui définit la propriété `html` sur `true`. Le texte au format HTML s'affiche dans le champ texte.

```
this.createTextField("my_txt", this.getNextHighestDepth(), 10, 10, 160,
  22);
my_txt.html = true;
my_txt.htmlText = "<b> this is bold text </b>";
```

La méthode `MovieClip.getNextHighestDepth()` utilisée dans cet exemple requiert Flash Player 7 ou version ultérieure. Si votre fichier SWF comporte un composant de la version 2, utilisez la classe DepthManager des composants de la version 2 au lieu de la méthode `MovieClip.getNextHighestDepth()`.

## Voir également

`html (propriété TextField.html)`, `Protocole asfunction`

# length (propriété TextField.length)

```
longueur publique : Number [lecture seule]
```

Indique le nombre de caractères d'un champ texte. Cette propriété renvoie la même valeur que `text.length`, mais est plus rapide. Un caractère tel que tab (`\t`) compte comme un seul caractère.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

## Exemple

L'exemple suivant renvoie le nombre de caractères du champ texte `date_txt` text field, ce qui affiche la date actuelle.

```
var today:Date = new Date();
this.createTextField("date_txt", this.getNextHighestDepth(), 10, 10, 100,
  22);
```

```
date_txt.autoSize = true;
date_txt.text = today.toString();
trace(date_txt.length);
```

La méthode `MovieClip.getNextHighestDepth()` utilisée dans cet exemple requiert Flash Player 7 ou version ultérieure. Si votre fichier SWF comporte un composant de la version 2, utilisez la classe DepthManager des composants de la version 2 au lieu de la méthode `MovieClip.getNextHighestDepth()`.

# maxChars (propriété TextField.maxChars)

```
public maxChars : Number
```

Indique le nombre maximum de caractères qu'un champ texte peut contenir. Un script peut insérer plus de texte que `maxChars` ne le permet ; la propriété `maxChars` n'indique que la quantité de texte qu'un utilisateur peut entrer. Si la valeur de cette propriété est `null`, la quantité de texte qu'un utilisateur peut entrer est illimitée.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

## Exemple

L'exemple suivant crée un champ texte appelé `age_txt` qui permet aux utilisateurs d'entrer jusqu'à deux chiffres dans ce champ.

```
this.createTextField("age_txt", this.getNextHighestDepth(), 10, 10, 30,
  22);
age_txt.type = "input";
age_txt.border = true;
age_txt.maxChars = 2;
age_txt.restrict = "0-9";
```

La méthode `MovieClip.getNextHighestDepth()` utilisée dans cet exemple requiert Flash Player 7 ou version ultérieure. Si votre fichier SWF comporte un composant de la version 2, utilisez la classe DepthManager des composants de la version 2 au lieu de la méthode `MovieClip.getNextHighestDepth()`.

# maxhscroll (propriété TextField.maxhscroll)

```
public maxhscroll : Number [lecture seule]
```

Indique la valeur maximale de `TextField.hscroll`.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

## Exemple

Consultez l'exemple relatif à `TextField.hscroll`.

# maxscroll (propriété TextField.maxscroll)

`public maxscroll : Number [lecture seule]`

Indique la valeur maximale de `TextField.scroll`.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

## Exemple

L'exemple suivant définit la valeur maximale du champ texte à défilement `my_txt`. Créez deux boutons, `scrollUp_btn` et `scrollDown_btn`, pour faire défiler le champ texte Ajoutez à votre fichier FLA ou AS le code ActionScript suivant.

```
this.createTextField("scroll_txt", this.getNextHighestDepth(), 10, 10, 160,
  20);
this.createTextField("my_txt", this.getNextHighestDepth(), 10, 30, 320,
  240);
my_txt.multiline = true;
my_txt.wordWrap = true;
for (var i = 0; i<10; i++) {
  my_txt.text += "Lorem ipsum dolor sit amet, consectetuer adipiscing elit,
  sed diam nonummy nibh "
      + "euismod tincidunt ut laoreet dolore magna aliquam erat volutpat.";
}
scrollUp_btn.onRelease = function() {
  my_txt.scroll--;
  scroll_txt.text = my_txt.scroll+" of "+my_txt.maxscroll;
};
scrollDown_btn.onRelease = function() {
  my_txt.scroll++;
  scroll_txt.text = my_txt.scroll+" of "+my_txt.maxscroll;
};
```

La méthode `MovieClip.getNextHighestDepth()` utilisée dans cet exemple requiert Flash Player 7 ou version ultérieure. Si votre fichier SWF comporte un composant de la version 2, utilisez la classe DepthManager des composants de la version 2 au lieu de la méthode `MovieClip.getNextHighestDepth()`.

# menu (propriété TextField.menu)

`public menu : ContextMenu`

Associe l'objet ContextMenu *`contextMenu`* au champ texte *`my_txt`*. La classe ContextMenu permet de modifier le menu contextuel qui s'affiche lorsque l'utilisateur clique avec le bouton droit de la souris (Windows) ou en appuyant sur la touche Contrôle (Macintosh) dans Flash Player.

Cette propriété fonctionne uniquement avec des champs texte (modifiables) ; elle n'a aucun effet sur les champs texte non sélectionnables.

**Disponibilité :** ActionScript 1.0 ; Flash Player 7

## Exemple

L'exemple suivant associe l'objet de ContextMenu `menu_cm` au champ texte `news_txt`. L'objet ContextMenu contient un élément de menu personnalisé appelé « Resize » associé à un gestionnaire de rappel appelé `doResize()`, qui peut permettre d'ajouter une fonctionnalité de redimensionnement (non illustrée) :

```
this.createTextField("news_txt", this.getNextHighestDepth(), 10, 10, 320,
  240);
news_txt.border = true;
news_txt.wordWrap = true;
news_txt.multiline = true;
news_txt.text = "To see the custom context menu item, right click (PC) or ";
news_txt.text += "control click (Mac) within the text field.";
var menu_cm:ContextMenu = new ContextMenu();
menu_cm.customItems.push(new ContextMenuItem("Resize", doResize));

function doResize(obj:TextField, item:ContextMenuItem):Void {
  // "Resize" code here
  trace("you selected: "+item.caption);
}
news_txt.menu = menu_cm;
```

Lorsque vous cliquez du bouton droit ou effectuez un Contrôle-clic sur le champ texte, l'élément menu personnalisé s'affiche.

*Remarque* : Vous ne pouvez pas utiliser un élément de menu qui est déjà utilisé par Flash. Par exemple `Print...` (avec trois petits points) est réservé par Flash, vous ne pouvez donc pas utiliser cet élément de menu. Vous pourriez toutefois utiliser `Print..` (avec deux petits points) ou tout élément de menu qui n'est pas encore utilisé par Flash.

Si votre fichier SWF comporte un composant de la version 2, utilisez la classe DepthManager des composants de la version 2 au lieu de la méthode `MovieClip.getNextHighestDepth()`, utilisée dans cet exemple.

## Voir également

`Button`, `ContextMenu`, `ContextMenuItem`, `MovieClip`

# mouseWheelEnabled (propriété TextField.mouseWheelEnabled)

```
public mouseWheelEnabled : Boolean
```

Valeur booléenne qui indique si Flash Player doit automatiquement faire défiler des champs texte multiligne lorsque le pointeur de la souris clique sur un champ texte et l'utilisateur actionne la molette. Par défaut, cette valeur est `true`. Cette propriété est utile si vous souhaitez empêcher le défilement des champs texte en actionnant la molette de la souris, ou implémenter votre propre défilement de champs texte.

**Disponibilité :** ActionScript 1.0 ; Flash Player 7

## Exemple

L'exemple suivant crée deux champs texte. Le champ `scrollable_txt` a la propriété `mouseWheelEnabled` définie sur vrai, ce qui veut dire que `scrollable_txt` défile lorsque vous cliquez sur ce champ et utilisez la roulette de la souris. Le champ `nonscrollable_txt` ne défile pas si vous cliquez sur ce champ et utilisez la roulette de la souris.

```
var font_array:Array = TextField.getFontList().sort();

this.createTextField("scrollable_txt", this.getNextHighestDepth(), 10, 10,
  240, 320);
scrollable_txt.border = true;
scrollable_txt.wordWrap = true;
scrollable_txt.multiline = true;
scrollable_txt.text = font_array.join("\n");

this.createTextField("nonscrollable_txt", this.getNextHighestDepth(), 260,
  10, 240, 320);
nonscrollable_txt.border = true;
nonscrollable_txt.wordWrap = true;
nonscrollable_txt.multiline = true;
nonscrollable_txt.mouseWheelEnabled = false;
nonscrollable_txt.text = font_array.join("\n");

Mouse.onMouseWheel
```

Si votre fichier SWF comporte un composant de la version 2, utilisez la classe DepthManager des composants de la version 2 au lieu de la méthode `MovieClip.getNextHighestDepth()`, utilisée dans cet exemple.

## Voir également

`mouseWheelEnabled (propriété TextField.mouseWheelEnabled)`

# multiline (propriété TextField.multiline)

```
public multiline : Boolean
```

Indique si le champ texte est un champ texte multiligne. Si la valeur est `true`, le champ texte est multiligne ; si la valeur est `false`, le champ texte est un champ texte sur une seule ligne.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

## Exemple

L'exemple suivant crée un champ texte multiligne appelé `fontList_txt` qui affiche une longue liste de polices sur plusieurs lignes.

```
var font_array:Array = TextField.getFontList().sort();

this.createTextField("fontList_txt", this.getNextHighestDepth(), 10, 10,
  240, 320);
fontList_txt.border = true;
fontList_txt.wordWrap = true;
fontList_txt.multiline = true;
fontList_txt.text = font_array.join("\n");
```

La méthode `MovieClip.getNextHighestDepth()` utilisée dans cet exemple requiert Flash Player 7 ou version ultérieure. Si votre fichier SWF comporte un composant de la version 2, utilisez la classe DepthManager des composants de la version 2 au lieu de la méthode `MovieClip.getNextHighestDepth()`.

# _name (propriété TextField._name)

```
public _name : String
```

Le nom de l'occurrence du champ texte.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

## Exemple

L'exemple suivant porte sur les champs texte résidant à différentes profondeurs. Créez un champ texte dynamique sur la Scène. Ajoutez le code ActionScript suivant à votre fichier FLA ou AS, qui crée de façon dynamique deux champs texte lors de l'exécution et affiche leur profondeur dans le panneau de sortie.

```
this.createTextField("first_mc", this.getNextHighestDepth(), 10, 10, 100,
  22);
this.createTextField("second_mc", this.getNextHighestDepth(), 10, 10, 100,
  22);
for (var prop in this) {
  if (this[prop] instanceof TextField) {
  var this_txt:TextField = this[prop];
```

```
  trace(this_txt._name+" is a TextField at depth: "+this_txt.getDepth());
  }
}
```

Lorsque vous testez le document, le nom et la profondeur de l'occurrence s'affichent dans le panneau de sortie.

La méthode `MovieClip.getNextHighestDepth()` utilisée dans cet exemple requiert Flash Player 7 ou version ultérieure. Si votre fichier SWF comporte un composant de la version 2, utilisez la classe DepthManager des composants de la version 2 au lieu de la méthode `MovieClip.getNextHighestDepth()`.

# onChanged (gestionnaire TextField.onChanged)

## onChanged = function(changedField:TextField) {}

Gestionnaire d'événements/écouteur : appelé lorsque le contenu d'un champ texte est modifié. Par défaut, il a la valeur undefined ; vous pouvez le définir dans un script.

Une référence vers l'occurrence de champ texte est transmise en tant que paramètre au gestionnaire `onChanged`. Vous pouvez capturer ces données en plaçant un paramètre dans la méthode du gestionnaire d'événements. Par exemple, le code suivant utilise `textfield_txt` en tant que paramètre à transmettre au gestionnaire d'événements `onChanged`. Le paramètre est ensuite utilisé dans une instruction `trace()` pour envoyer le nom de l'occurrence du champ texte vers le panneau de sortie :

```
this.createTextField("myInputText_txt", 99, 10, 10, 300, 20);
myInputText_txt.border = true;
myInputText_txt.type = "input";

myInputText_txt.onChanged = function(textfield_txt:TextField) {
trace("the value of "+textfield_txt._name+" was changed. New value is:
  "+textfield_txt.text);
};
```

Le gestionnaire `onChanged` est appelé uniquement lorsque la modification résulte d'une interaction de l'utilisateur ; par exemple, lorsque l'utilisateur tape quelque chose sur le clavier, modifie quelque chose dans le champ texte à l'aide de la souris, ou sélectionne un élément du menu. Des modifications par programme dans le champ texte ne déclenchent pas l'événement `onChanged` étant donné que le code reconnaît les modifications apportées au champ texte.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

### Paramètres

*`changedField`*`:TextField` - Champ déclenchant l'événement.

### Voir également

TextFormat, setNewTextFormat (méthode TextField.setNewTextFormat)

## onKillFocus (gestionnaire TextField.onKillFocus)

onKillFocus = function(newFocus:Object) {}

Appelé lorsqu'un champ texte perd le focus clavier. La méthode `onKillFocus` reçoit un paramètre, `newFocus` : il s'agit d'un objet représentant le nouvel objet recevant le focus. Si aucun objet ne reçoit le focus, `newFocus` contient la valeur `null`.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

### Paramètres

*`newFocus`*`:Object` - Objet qui reçoit le focus.

### Exemple

L'exemple suivant crée deux champs texte, appelés `first_txt` et `second_txt`. Lorsque vous donnez le focus à un champ texte, les informations relatives à ce champ texte et au champ texte ayant perdu le focus s'affichent dans le panneau de sortie.

```
this.createTextField("first_txt", 1, 10, 10, 300, 20);
first_txt.border = true;
first_txt.type = "input";
this.createTextField("second_txt", 2, 10, 40, 300, 20);
second_txt.border = true;
second_txt.type = "input";
first_txt.onKillFocus = function(newFocus:Object) {
  trace(this._name+" lost focus. New focus changed to: "+newFocus._name);
};
first_txt.onSetFocus = function(oldFocus:Object) {
  trace(this._name+" gained focus. Old focus changed from:
  "+oldFocus._name);
}
```

### Voir également

onSetFocus (gestionnaire TextField.onSetFocus)

## onScroller (gestionnaire TextField.onScroller)

onScroller = function(scrolledField:TextField) {}

Gestionnaire d'événements/écouteur : appelé lorsque l'une des propriétés de défilement du champ texte est modifiée.

Une référence vers l'occurrence de champ texte est transmise en tant que paramètre au gestionnaire `onScroller`. Vous pouvez capturer ces données en plaçant un paramètre dans la méthode du gestionnaire d'événements. Par exemple, le code suivant utilise `my_txt` en tant que paramètre à transmettre au gestionnaire d'événements `onScroller`. Le paramètre est ensuite utilisé dans une instruction `trace()` pour envoyer le nom de l'occurrence du champ texte vers le panneau de sortie.

```
myTextField.onScroller = function (my_txt:TextField) {
  trace (my_txt._name + " scrolled");
};
```

Le gestionnaire d'événements `TextField.onScroller` est généralement utilisé pour implémenter des barres de défilement. Les barres de défilement comportent généralement un curseur de défilement ou un autre indicateur qui spécifie la position de défilement horizontal ou vertical actuel dans un champ texte. Les champs texte peuvent être parcourus à l'aide de la souris et du clavier, ce qui entraîne une modification de la position de défilement. Le code de la barre de défilement doit être notifié si la position de défilement change suite à ce type d'interaction utilisateur, ce qui est le but de `TextField.onScroller`.

`onScroller` est appelé lorsque la position de défilement a changé suite à l'interaction des utilisateurs avec le champ texte ou à des modifications programmatiques. Le gestionnaire `onChanged` se déclenche uniquement si une interaction de l'utilisateur entraîne une modification. Ces deux options sont nécessaires étant donné qu'une partie du code change souvent la position de défilement, tandis que le code de la barre de défilement n'est pas associé et ne sait pas que la position de défilement a été modifiée sans être notifiée.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

## Paramètres

*`scrolledField`*`:TextField` - Rférence à l'objet TextField dont la position de défilement a changé.

## Exemple

L'exemple suivant crée un champ texte appelé `my_txt`, et utilise deux boutons appelés `scrollUp_btn` et `scrollDown_btn` pour faire défiler le contenu du champ texte. Lorsque le gestionnaire d'événements `onScroller` est appelé, une instruction trace permet d'afficher des informations dans le panneau de sortie. Créez deux boutons dont les noms d'occurrence sont `scrollUp_btn` et `scrollDown_btn`, et ajoutez le code ActionScript suivant à votre fichier FLA ou AS :

```
this.createTextField("scroll_txt", this.getNextHighestDepth(), 10, 10, 160,
  20);
this.createTextField("my_txt", this.getNextHighestDepth(), 10, 30, 320,
  240);
```

```
my_txt.multiline = true;
my_txt.wordWrap = true;

for (var i = 0; i<10; i++) {
  my_txt.text += "Lorem ipsum dolor sit amet, consectetuer adipiscing elit,
  sed diam "
    + "nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat
  volutpat.";
}
scrollUp_btn.onRelease = function() {
  my_txt.scroll--;
};
scrollDown_btn.onRelease = function() {
  my_txt.scroll++;
};
my_txt.onScroller = function() {
    trace("onScroller called");
  scroll_txt.text = my_txt.scroll+" of "+my_txt.maxscroll;
};
```

La méthode `MovieClip.getNextHighestDepth()` utilisée dans cet exemple requiert Flash Player 7 ou version ultérieure. Si votre fichier SWF comporte un composant de la version 2, utilisez la classe DepthManager des composants de la version 2 au lieu de la méthode `MovieClip.getNextHighestDepth()`.

### Voir également

`hscroll (propriété TextField.hscroll)`, `maxhscroll (propriété TextField.maxhscroll)`, `maxscroll (propriété TextField.maxscroll)`, `scroll (propriété TextField.scroll)`

## onSetFocus (gestionnaire TextField.onSetFocus)

onSetFocus = function(oldFocus:Object) {}

Appelé lorsqu'un champ texte reçoit le focus clavier. Le paramètre `oldFocus` est l'objet qui perd le focus. Par exemple, si l'utilisateur appuie sur la touche Tab pour déplacer le focus d'entrée d'un bouton vers un champ texte, le paramètre oldFocus contient l'occurrence de bouton. Si aucun objet n'avait précédemment reçu le focus, `oldFocus` contient la valeur null.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

### Paramètres

*`oldFocus`*`:Object` - Objet perdant le focus.

### Exemple

Consultez l'exemple relatif à `TextField.onKillFocus`.

### Voir également

`onKillFocus (gestionnaire TextField.onKillFocus)`

## _parent (propriété TextField._parent)

```
public _parent : MovieClip
```

Référence au clip ou à l'objet contenant le champ texte ou l'objet actuel. L'objet actuel est celui qui contient le code ActionScript qui fait référence à `_parent`.

Utilisez `_parent` pour spécifier un chemin relatif vers les clips ou les objets situés au-dessus du champ texte actuel. Vous pouvez utiliser `_parent` pour remonter de plusieurs niveaux dans l'arborescence de la liste d'affichage, comme dans l'exemple suivant :

```
_parent._parent._alpha = 20;
```

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

### Exemple

Le code ActionScript suivant crée deux champs texte et renvoie des informations sur la propriété `_parent` de chaque objet. Le premier champ texte, `first_txt`, est créé sur le scénario principal. Le deuxième champ texte, `second_txt`, est créé dans le clip appelé `holder_mc`.

```
this.createTextField("first_txt", this.getNextHighestDepth(), 10, 10, 160,
  22);
first_txt.border = true;
trace(first_txt._name+"'s _parent is: "+first_txt._parent);

this.createEmptyMovieClip("holder_mc", this.getNextHighestDepth());
holder_mc.createTextField("second_txt", holder_mc.getNextHighestDepth(),
  10, 40, 160, 22);
holder_mc.second_txt.border = true;
trace(holder_mc.second_txt._name+"'s _parent is:
  "+holder_mc.second_txt._parent);
```

Les informations suivantes apparaissent dans le panneau de sortie :

```
first_txt's _parent is: _level0
second_txt's _parent is: _level0.holder_mc
```

La méthode `MovieClip.getNextHighestDepth()` utilisée dans cet exemple requiert Flash Player 7 ou version ultérieure. Si votre fichier SWF comporte un composant de la version 2, utilisez la classe DepthManager des composants de la version 2 au lieu de la méthode `MovieClip.getNextHighestDepth()`.

### Voir également

`_parent (propriété Button._parent)`, `_parent (propriété MovieClip._parent)`, ,

## password (propriété TextField.password)

```
public password : Boolean
```

Indique si le champ texte est un champ texte de mot de passe. Si la valeur de password est `true`, le champ texte est un champ texte à mot de passe et masque les caractères d'entrée en utilisant les astérisques à la place des caractères actuels. Si `false`, le champ texte n'est pas un champ texte de mot de passe. Lorsque le mode mot de passe est activé, les commandes *Couper* et *Copier* et leurs raccourcis clavier ne fonctionnent pas. Ce mécanisme de sécurité empêche un utilisateur malhonnête d'utiliser les raccourcis pour découvrir le mot de passe d'un ordinateur sans surveillance.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

### Exemple

L'exemple suivant crée deux champs texte : `username_txt` et `password_txt`. Du texte est placé dans les deux champs texte ; toutefois, les propriétés de mot de passe `password_txt` sont définies sur `true`. Par conséquent, les caractères s'affichent sous forme d'astérisques dans le champ `password_txt`.

```
this.createTextField("username_txt", this.getNextHighestDepth(), 10, 10,
  100, 22);
username_txt.border = true;
username_txt.type = "input";
username_txt.maxChars = 16;
username_txt.text = "hello";

this.createTextField("password_txt", this.getNextHighestDepth(), 10, 40,
  100, 22);
password_txt.border = true;
password_txt.type = "input";
password_txt.maxChars = 16;
password_txt.password = true;
password_txt.text = "world";
```

La méthode `MovieClip.getNextHighestDepth()` utilisée dans cet exemple requiert Flash Player 7 ou version ultérieure. Si votre fichier SWF comporte un composant de la version 2, utilisez la classe DepthManager des composants de la version 2 au lieu de la méthode `MovieClip.getNextHighestDepth()`.

# _quality (propriété TextField._quality)

```
public _quality : String
```

La qualité de rendu utilisée pour un fichier SWF. Les polices de périphérique sont toujours aliasées, ce qui implique qu'elles ne sont pas affectées par la propriété `_quality`.

*Remarque :* Bien que vous puissiez spécifier cette propriété pour un objet TextField, il s'agit en fait d'une propriété globale : il vous suffit donc de définir sa valeur sur `_quality`. Pour plus d'informations, consultez la propriété globale `_quality`.

La propriété `_quality` peut être définie sur les valeurs suivantes :

- `"LOW"` Qualité de rendu inférieure. Les images ne sont pas anti-aliasées et les bitmaps ne sont pas lissés.
- `"MEDIUM"` Qualité de rendu moyenne. Les images sont anti-aliasées selon une grille de 2 x 2 pixels, mais les bitmaps ne sont pas lissés. Convient aux animations qui ne contiennent pas de texte.
- `"HIGH"` Qualité de rendu supérieure. Les images sont anti-aliasées en appliquant une grille de 4 x 4 pixels et les bitmaps sont lissés lorsque l'animation est statique. Il s'agit du paramètre de qualité de rendu par défaut de Flash.
- `"BEST"` Qualité de rendu optimale. Les graphiques sont anti-aliasés selon une grille de 4 x 4 pixels et les bitmaps sont toujours lissés.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

## Exemple

L'exemple suivant définit la qualité du rendu sur `LOW`:

```
my_txt._quality = "LOW";
```

## Voir également

`_quality, propriété`

# removeListener (méthode TextField.removeListener)

`public removeListener(listener:Object) : Boolean`

Supprime un objet écouteur précédemment enregistré dans une occurrence de champ texte avec `TextField.addListener()`.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

## Paramètres

*`listener`*`:Object` - Objet qui ne recevra plus de notifications en provenance de `TextField.onChanged` ou de `TextField.onScroller`.

## Valeur renvoyée

`Boolean` - Si `listener` a été supprimé avec succès, la méthode renvoie une valeur `true` . Si `listener` n'a pas été supprimé avec succès (par exemple, si `listener` n'était pas sur la liste d'écouteurs de l'objet TextField, la méthode renvoie une valeur de `false`.

## Exemple

L'exemple suivant crée un champ texte de saisie appelé `my_txt`. Lorsque l'utilisateur tape du texte dans le champ, les informations sur le nombre de caractères du champ texte s'affichent dans le panneau de sortie. Si l'utilisateur clique sur l'occurrence `removeListener_btn` l'écouteur est supprimé et les informations ne sont plus affichées.

```
this.createTextField("my_txt", this.getNextHighestDepth(), 10, 10, 160,
  20);
my_txt.border = true;
my_txt.type = "input";

var txtListener:Object = new Object();
txtListener.onChanged = function(textfield_txt:TextField) {
  trace(textfield_txt+" changed. Current length is:
  "+textfield_txt.length);
};
my_txt.addListener(txtListener);

removeListener_btn.onRelease = function() {
  trace("Removing listener...");
  if (!my_txt.removeListener(txtListener)) {
  trace("Error! Unable to remove listener");
  }
};
```

La méthode `MovieClip.getNextHighestDepth()` utilisée dans cet exemple requiert Flash Player 7 ou version ultérieure. Si votre fichier SWF comporte un composant de la version 2, utilisez la classe DepthManager des composants de la version 2 au lieu de la méthode `MovieClip.getNextHighestDepth()`.

# removeTextField (méthode TextField.removeTextField)

```
public removeTextField() : Void
```

Supprime le champ texte. Cette opération ne peut être réalisée que sur un champ texte qui a été créé avec `MovieClip.createTextField()`. Lorsque vous appelez cette méthode, le champ texte est supprimé. Cette méthode est similaire à `MovieClip.removeMovieClip()`.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

## Exemple

L'exemple suivant crée un champ texte que vous pouvez supprimer de la Scène lorsque vous cliquez sur l'occurrence remove_btn. Créez un bouton et appelez-le `remove_btn`,puis ajoutez le code ActionScript suivant à votre fichier FLA ou AS.

```
this.createTextField("my_txt", this.getNextHighestDepth(), 10, 10, 300,
  22);
my_txt.text = new Date().toString();
my_txt.border = true;

remove_btn.onRelease = function() {
  my_txt.removeTextField();
};
```

La méthode `MovieClip.getNextHighestDepth()` utilisée dans cet exemple requiert Flash Player 7 ou version ultérieure. Si votre fichier SWF comporte un composant de la version 2, utilisez la classe DepthManager des composants de la version 2 au lieu de la méthode `MovieClip.getNextHighestDepth()`.

# replaceSel (méthode TextField.replaceSel)

```
public replaceSel(newText:String) : Void
```

Remplace la sélection actuelle par le contenu du paramètre `newText`. Le texte est inséré au niveau de la sélection actuelle, à l'aide du format de caractère par défaut actuel et du format de paragraphe par défaut. Le texte n'est pas traité comme du code HTML, même si le champ texte est un champ texte HTML.

Vous pouvez utiliser la méthode `replaceSel()` pour insérer et effacer du texte sans perturber la mise en forme des caractères et du paragraphe du reste du texte.

Vous devez utiliser `Selection.setFocus()` pour choisir le champ avant de transmettre cette commande.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

## Paramètres

*`newText`*`:String` - Chaîne.

## Exemple

L'exemple de code suivant crée un champ texte multiligne avec du texte figurant sur la Scène. Lorsque vous sélectionnez du texte, puis cliquez du bouton droit de la souris ou effectuez un Contrôle-clic sur ce champ, vous pouvez sélectionner `Enter current date` dans le menu contextuel. Cette sélection appelle une fonction qui remplace le texte sélectionné par la date actuelle.

```
this.createTextField("my_txt", this.getNextHighestDepth(), 10, 10, 320,
  240);
my_txt.border = true;
my_txt.wordWrap = true;
my_txt.multiline = true;
my_txt.type = "input";
my_txt.text = "Select some sample text from the text field and then right-
  click/control click "
    + "and select 'Enter current date' from the context menu to replace the
  "
    + "currently selected text with the current date.";

var my_cm:ContextMenu = new ContextMenu();
my_cm.customItems.push(new ContextMenuItem("Enter current date",
  enterDate));
function enterDate(obj:Object, menuItem:ContextMenuItem) {
  var today_str:String = new Date().toString();
  var date_str:String = today_str.split(" ", 3).join(" ");
  my_txt.replaceSel(date_str);
}
my_txt.menu = my_cm;
```

La méthode `MovieClip.getNextHighestDepth()` utilisée dans cet exemple requiert Flash Player 7 ou version ultérieure. Si votre fichier SWF comporte un composant de la version 2, utilisez la classe DepthManager des composants de la version 2 au lieu de la méthode `MovieClip.getNextHighestDepth()`.

### Voir également

`setFocus (méthode Selection.setFocus)`

## replaceText (méthode TextField.replaceText)

```
public replaceText(beginIndex:Number, endIndex:Number, newText:String) :
  Void
```

Remplace une plage de caractères, spécifiée par les paramètres `beginIndex` et `endIndex` dans le champ texte spécifié, par le contenu du paramètre `newText`.

**Disponibilité :** ActionScript 1.0 ; Flash Player 7

### Paramètres

*`beginIndex`*`:Number` - Valeur de début de l'index pour la plage de remplacement.

*`endIndex`*`:Number` - Valeur de fin de l'index pour la plage de remplacement.

*`newText`*`:String` - Texte à utiliser pour remplacer la plage de caractères spécifiée.

### Exemple

L'exemple suivant crée un champ texte appelé `my_txt` et lui associe le texte `dog@house.net`. La méthode `indexOf()` permet de rechercher la première occurrence du symbole spécifié (@). Si le symbole est trouvé, le texte spécifié (entre l'index de 0 et le symbole) remplace la chaîne `bird`. Si le symbole n'est pas trouvé, un message d'erreur est affiché sur le panneau de sortie.

```
this.createTextField("my_txt", this.getNextHighestDepth(), 10, 10, 320,
  22);
my_txt.autoSize = true;
my_txt.text = "dog@house.net";

var symbol:String = "@";
var symbolPos:Number = my_txt.text.indexOf(symbol);
if (symbolPos>-1) {
  my_txt.replaceText(0, symbolPos, "bird");
} else {
  trace("symbol '"+symbol+"' not found.");
}
```

Si votre fichier SWF comporte un composant de la version 2, utilisez la classe DepthManager des composants de la version 2 au lieu de la méthode `MovieClip.getNextHighestDepth()`, utilisée dans cet exemple.

# restrict (propriété TextField.restrict)

```
public restrict : String
```

Indique le jeu de caractères qu'un utilisateur peut rentrer dans le champ texte. Si la valeur de la propriété `restrict` est `null`, vous pouvez entrer n'importe quel caractère. Si la valeur de la propriété `restrict` est une chaîne vide, aucun caractère ne peut être entré. Si la valeur de la propriété `restrict` est une chaîne de caractères, vous ne pouvez entrer que les caractères dans la chaîne du champ texte. La chaîne est lue de gauche à droite. Une plage peut être spécifiée en utilisant un tiret (-). Ceci ne limite que l'interaction avec l'utilisateur ; un script peut mettre n'importe quel texte dans le champ texte. Cette propriété ne se synchronise pas avec les cases à cocher de polices vectorielles intégrées de l'inspecteur des propriétés.

Si la chaîne commence par un caret, tous les caractères sont initialement acceptés et les caractères suivants de la chaîne sont exclus du jeu de caractères acceptés. Si la chaîne ne commence pas par un caret, aucun caractère n'est initialement accepté et les caractères suivants de la chaîne sont inclus dans le jeu de caractères acceptés.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

## Exemple

L'exemple suivant autorise uniquement les caractères en majuscules, les espaces et les nombres dans le champ texte :

```
my_txt.restrict = "A-Z 0-9";
```

L'exemple suivant exclut uniquement les caractères en minuscules :

```
my_txt.restrict = "^a-z";
```

Insérez une barre oblique pour saisir un ^ ou un -. Les séquences acceptables sont \-, \^ ou \\. Si la barre oblique doit être un caractère réel dans la chaîne, vous devez également la faire précéder d'une autre barre oblique dans le code ActionScript. Par exemple, le code suivant inclut uniquement le tiret (-) et le signe circonflexe (^) :

```
my_txt.restrict = "\\-\\^";
```

Le caractère ^ peut être utilisé n'importe où dans la chaîne pour faire alterner l'inclusion et l'exclusion des caractères. Le code suivant inclut uniquement des lettres en majuscules, mais exclut la lettre Q en majuscules :

```
my_txt.restrict = "A-Z^Q";
```

Vous pouvez utiliser la séquence d'échappement `\u` pour créer des chaînes `restrict`. Le code suivant inclut uniquement les caractères ASCII allant de 32 à 126 (tilde).

```
my_txt.restrict = "\u0020-\u007E";
```

# _rotation (propriété TextField._rotation)

```
public _rotation : Number
```

Rotation du champ texte, en degrés, à partir de son orientation d'origine. Les valeurs comprises entre 0 et 180 représentent la rotation en sens horaire ; les valeurs comprises entre 0 et -180 représentent la rotation en sens anti-horaire. Les valeurs hors de cette plage sont ajoutées ou soustraites de 360 pour obtenir une valeur comprise dans la plage. Par exemple les instructions `my_txt._rotation = 450` et `my_txt._rotation = 90` sont les mêmes.

Les valeurs de rotation ne sont pas prises en charge pour les champs texte qui utilisent des polices de périphérique. Vous devez utiliser des polices intégrées pour associer `_rotation` à un champ texte.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

## Exemple

Dans cet exemple, vous devez créer un champ texte dynamique appelé `my_txt`, puis utiliser le code ActionScript suivant pour incorporer des polices et faire pivoter le champ texte. La référence à `my font` renvoie à un symbole de police dans la bibliothèque, avec une liaison définie sur `my font`.

```
var my_fmt:TextFormat = new TextFormat();
my_fmt.font = "my font";

this.createTextField("my_txt", this.getNextHighestDepth(), 10, 10, 160,
  120);
my_txt.wordWrap = true;
my_txt.embedFonts = true;
my_txt.text = "Hello world";
my_txt.setTextFormat(my_fmt);
my_txt._rotation = 45;
```

Appliquez une mise en forme supplémentaire au champ texte avec la classe `TextFormat class`.

La méthode `MovieClip.getNextHighestDepth()` utilisée dans cet exemple requiert Flash Player 7 ou version ultérieure. Si votre fichier SWF comporte un composant de la version 2, utilisez la classe DepthManager des composants de la version 2 au lieu de la méthode `MovieClip.getNextHighestDepth()`.

## Voir également

`_rotation (propriété Button._rotation)`, `_rotation (propriété MovieClip._rotation)`, `TextFormat`

# scroll (propriété TextField.scroll)

```
public scroll : Number
```

La position verticale du texte dans un champ texte. La propriété scroll est utile pour diriger les utilisateurs vers un paragraphe spécifique dans un long passage, ou pour créer des champs texte défilants. Cette propriété peut être récupérée et modifiée.

Les unités du défilement horizontal sont les pixels, alors que les unités du défilement vertical sont les lignes. Le défilement horizontal est mesuré en pixels étant donné que la plupart des polices que vous utilisez généralement sont proportionnellement espacées, c'est-à-dire que les caractères peuvent avoir différentes largeurs. Flash propose un défilement vertical par ligne étant donné que les utilisateurs souhaitent que l'ensemble de la ligne de texte soit visible et non une partie de la ligne seulement. Même s'il existe plusieurs polices sur une ligne, la hauteur de la ligne s'adapte à la plus grande police utilisée.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

## Exemple

L'exemple suivant définit la valeur maximale du champ texte à défilement `my_txt`. Créez deux boutons, `scrollUp_btn` et `scrollDown_btn` pour faire défiler le champ texte. Ajoutez le code ActionScript suivant à votre fichier FLA ou AS :

```
this.createTextField("scroll_txt", this.getNextHighestDepth(), 10, 10, 160,
  20);
this.createTextField("my_txt", this.getNextHighestDepth(), 10, 30, 320,
  240);
my_txt.multiline = true;
my_txt.wordWrap = true;
for (var i = 0; i<10; i++) {
  my_txt.text += "Lorem ipsum dolor sit amet, consectetuer adipiscing elit,
  sed diam nonummy "
    + "nibh euismod tincidunt ut laoreet dolore magna aliquam erat
  volutpat.";
}
scrollUp_btn.onRelease = function() {
  my_txt.scroll--;
  scroll_txt.text = my_txt.scroll+" of "+my_txt.maxscroll;
};
scrollDown_btn.onRelease = function() {
  my_txt.scroll++;
  scroll_txt.text = my_txt.scroll+" of "+my_txt.maxscroll;
};
```

La méthode `MovieClip.getNextHighestDepth()` utilisée dans cet exemple requiert Flash Player 7 ou version ultérieure. Si votre fichier SWF comporte un composant de la version 2, utilisez la classe DepthManager des composants de la version 2 au lieu de la méthode `MovieClip.getNextHighestDepth()`.

### Voir également

hscroll (propriété TextField.hscroll), maxscroll (propriété TextField.maxscroll)

## selectable (propriété TextField.selectable)

```
public selectable : Boolean
```

Valeur booléenne indiquant si le champ texte est sélectionnable. La valeur `true` indique que le texte est sélectionnable. La propriété `selectable` commande si un champ texte est sélectionnable, et non si un champ texte est modifiable. Un champ texte dynamique peut être sélectionnable même s'il n'est pas modifiable. Si un champ texte dynamique n'est pas sélectionnable, ceci signifie que vous ne pouvez pas sélectionner son texte.

Si selectable est défini sur `false`, le texte du champ texte ne répond pas aux commandes de sélection de la souris ou du clavier, et le texte ne peut pas être copié à l'aide de la commande Copy. Si selectable est défini sur `true`, le texte du champ texte peut être sélectionné à l'aide de la souris ou du clavier. Vous pouvez sélectionner le texte de cette manière même si le champ texte est un champ texte dynamique et non un champ texte de saisie. Le texte peut être copié à l'aide de la commande Copy.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

### Exemple

L'exemple suivant crée un champ texte sélectionnable qui se met constamment à jour en fonction de la date et de l'heure.

```
this.createTextField("date_txt", this.getNextHighestDepth(), 10, 10, 100,
  22);
date_txt.autoSize = true;
date_txt.selectable = true;

var date_interval:Number = setInterval(updateTime, 500, date_txt);
function updateTime(my_txt:TextField) {
  my_txt.text = new Date().toString();
}
```

La méthode `MovieClip.getNextHighestDepth()` utilisée dans cet exemple requiert Flash Player 7 ou version ultérieure. Si votre fichier SWF comporte un composant de la version 2, utilisez la classe DepthManager des composants de la version 2 au lieu de la méthode `MovieClip.getNextHighestDepth()`.

# setNewTextFormat (méthode TextField.setNewTextFormat)

```
public setNewTextFormat(tf:TextFormat) : Void
```

Définit le format par défaut du nouveau texte dans un champ texte. Le format par défaut du nouveau texte correspond au format appliqué au nouveau texte inséré, tel que le texte inséré avec la méthode `replaceSel()` ou le texte entré par un utilisateur. Lorsque du texte est inséré, le nouveau format texte par défaut est attribué au nouveau texte inséré.

Le nouveau format texte par défaut est spécifié par `textFormat`, qui est un objet TextFormat.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

## Paramètres

*`tf`*`:TextFormat` - Objet TextFormat.

## Exemple

Dans l'exemple suivant, un champ texte (appelé `my_txt`) est créé lors de l'exécution et plusieurs propriétés sont définies. Le format du texte le plus récent s'applique.

```
var my_fmt:TextFormat = new TextFormat();
my_fmt.bold = true;
my_fmt.font = "Arial";
my_fmt.color = 0xFF9900;

this.createTextField("my_txt", 999, 0, 0, 400, 300);
my_txt.wordWrap = true;
my_txt.multiline = true;
my_txt.border = true;
my_txt.type = "input";
my_txt.setNewTextFormat(my_fmt);
my_txt.text = "Oranges are a good source of vitamin C";
```

## Voir également

`getNewTextFormat (méthode TextField.getNewTextFormat)`, `getTextFormat (méthode TextField.getTextFormat)`, `setTextFormat (méthode TextField.setTextFormat)`

# setTextFormat (méthode TextField.setTextFormat)

```
public setTextFormat([beginIndex:Number], [endIndex:Number],
  textFormat:TextFormat) : Void
```

Applique la mise en format de texte spécifié par le paramètre`textFormat` à tout ou partie du texte du champ. `textFormat` doit être un TextFormat qui spécifie les modifications de formatage voulues. Seules les propriétés non null de textFormat sont appliquées au champ texte. Toute propriété de `textFormat` qui est définie sur null ne sera pas appliquée. Par défaut, toutes les propriétés d'un nouvel objet TextFormat créé sont définies sur `null`.

Il existe deux types de mise en forme des informations dans un objet TextFormat : mise en forme au niveau des caractères et au niveau du paragraphe. Chaque caractère dans un champ texte peut avoir ses propres paramètres de mise en forme de caractère, tels que le nom de la police, la taille de la police, gras et italique.

Pour les paragraphes, le premier caractère du paragraphe est analysé pour les paramètres de mise en forme du paragraphe entier. La marge gauche, la marge droite et le retrait sont des exemples de paramètres de mise en forme de paragraphe.

La méthode `setTextFormat()` modifie la mise en forme de texte appliquée à chaque caractère, à une plage de caractères ou à l'ensemble du corps de texte d'un champ texte. Ces utilisations sont indiquées dans le tableau suivant.

| Utilisation | Description |
|---|---|
| `my_textField.setTextFormat(textFormat:TextFormat)` | Applique les propriétés de `textFormat` à l'ensemble du texte dans le champ texte. |
| `my_textField.setTextFormat(beginIndex:Number, textFormat:TextFormat)` | Applique les propriétés de `textFormat` au caractère sur la position `beginIndex`. |
| `my_textField.setTextFormat(beginIndex:Number, endIndex:Number, textFormat:TextFormat)` | Applique les propriétés du paramètre `textFormat` à la plage de texte de la position `beginIndex` à la position `endIndex`. |

Notez que tout texte inséré manuellement par l'utilisateur ou remplacé à l'aide de `TextField.replaceSel()`, reçoit la mise en forme par défaut du champ texte pour un nouveau texte, et non la mise en forme spécifiée pour le point d'insertion du texte. Pour définir la mise en forme par défaut d'un champ texte pour un nouveau texte, utilisez `TextField.setNewTextFormat()`.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

## Paramètres

*beginIndex*:Number [facultatif] - Entier qui spécifie le premier caractère de la plage de texte souhaitée. Si vous ne spécifiez pas beginIndex et endIndex, l'objet TextFormat est appliqué à l'ensemble du TextField.

*endIndex*:Number [facultatif] - Entier qui spécifie le premier caractère après la plage de texte souhaitée. Si vous spécifiez beginIndex mais ne spécifiez pas endIndex, le TextFormat est appliqué au seul caractère spécifié par beginIndex.

*textFormat*:TextFormat - Objet TextFormat contenant des informations sur la mise en forme des caractères et des paragraphes.

## Exemple

L'exemple suivant définit le format de texte de deux chaînes distinctes. La méthode setTextFormat() est appelée et appliquée au champ texte my_txt.

```
var format1_fmt:TextFormat = new TextFormat();
format1_fmt.font = "Arial";
var format2_fmt:TextFormat = new TextFormat();
format2_fmt.font = "Courier";

var string1:String = "Sample string number one."+newline;
var string2:String = "Sample string number two."+newline;

this.createTextField("my_txt", this.getNextHighestDepth(), 0, 0, 300, 200);
my_txt.multiline = true;
my_txt.wordWrap = true;
my_txt.text = string1;
var firstIndex:Number = my_txt.length;
my_txt.text += string2;
var secondIndex:Number = my_txt.length;

my_txt.setTextFormat(0, firstIndex, format1_fmt);
my_txt.setTextFormat(firstIndex, secondIndex, format2_fmt);
```

La méthode MovieClip.getNextHighestDepth() utilisée dans cet exemple requiert Flash Player 7 ou version ultérieure. Si votre fichier SWF comporte un composant de la version 2, utilisez la classe DepthManager des composants de la version 2 au lieu de la méthode MovieClip.getNextHighestDepth().

## Voir également

TextFormat, setNewTextFormat (méthode TextField.setNewTextFormat)

# sharpness (propriété TextField.sharpness)

```
public sharpness : Number
```

La netteté des bords du glyphe dans cette occurrence TextField. Cette propriété ne s'applique que si la propriété `antiAliasType` du champ texte est définie sur `"advanced"`. La plage pour `sharpness` est un nombre compris entre -400 et 400. Si vous tentez de définir `sharpness` sur une valeur non comprise dans cette plage, Flash définit la propriété sur la valeur la plus proche dans la plage (-400 ou 400).

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

## Exemple

Cet exemple crée trois champs texte avec `sharpness` définis sur `400`, `0` et `-400`. Il suppose que vous avez une police intégrée à la librairie avec l'identifiant de liaison défini sur `"Times-12"`. Pour intégrer la police, procédez comme suit :

- Ouvrez votre Bibliothèque
- Cliquez sur le menu d'options de la Bibliothèque dans le coin supérieur droit de la Bibliothèque
- Sélectionnez « Nouvelle Police » dans la liste déroulante
- Nommez la police « Times-12 »
- Sélectionnez « Times New Roman » dans le menu déroulant
- Appuyez sur le bouton « OK »
- Cliquez du bouton droit sur la nouvelle police créée et sélectionnez « Liaison... »
- Cochez la case « Exporter pour ActionScript »
- Acceptez l'identifiant par défaut « Times-12 » en appuyant sur le bouton « OK »

```
var my_format:TextFormat = new TextFormat();
my_format.font = "Times-12";

var my_text1:TextField = this.createTextField("my_text1",
  this.getNextHighestDepth(), 10, 10, 400, 100);
my_text1.text = "This text has sharpness set to 400."
my_text1.embedFonts = true;
my_text1.antiAliasType = "advanced";
my_text1.gridFitType = "pixel";
my_text1.sharpness = 400;
my_text1.setTextFormat(my_format);

var my_text2:TextField = this.createTextField("my_text2",
  this.getNextHighestDepth(), 10, 40, 400, 100);
my_text2.text = "This text has sharpness set to 0."
my_text2.embedFonts = true;
my_text2.antiAliasType = "advanced";
```

```
my_text2.gridFitType = "pixel";
my_text2.sharpness = 0;
my_text2.setTextFormat(my_format);

var my_text3:TextField = this.createTextField("my_text3",
  this.getNextHighestDepth(), 10, 70, 400, 100);
my_text3.text = "This text has sharpness set to -400."
my_text3.embedFonts = true;
my_text3.antiAliasType = "advanced";
my_text3.gridFitType = "pixel";
my_text3.sharpness = -400;
my_text3.setTextFormat(my_format);
```

Si votre fichier SWF comporte un composant de la version 2, utilisez la classe DepthManager des composants de la version 2 au lieu de la méthode `MovieClip.getNextHighestDepth()`, utilisée dans cet exemple.

### Voir également

`gridFitType (propriété TextField.gridFitType)`, `antiAliasType (propriété TextField.antiAliasType)`

## _soundbuftime (propriété TextField._soundbuftime)

```
public _soundbuftime : Number
```

Le nombre de secondes pendant lequel les sons sont chargés en mémoire tampon avant d'être diffusés en continu.

*Remarque* : Bien que vous puissiez spécifier cette propriété pour un objet TextField object, il s'agit en fait d'une propriété globale qui s'applique à l'ensemble des sons qui ont été chargés. Il vous suffit donc de définir sa valeur sur`_soundbuftime`. La définition de cette propriété pour un objet TextField définit en fait la propriété globale.

Pour plus d'informations et un exemple, consultez la propriété globale `_soundbuftime`.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

### Voir également

`_soundbuftime, propriété`

## styleSheet (propriété TextField.styleSheet)

```
public styleSheet : StyleSheet
```

Associe une feuille de style au champ texte. Pour plus d'informations sur la création des feuilles de style, consultez l'entrée relative à la classe TextField.StyleSheet.

La feuille de style associée à un champ texte peut être modifiée à tout moment. Si la feuille de style en cours d'utilisation est modifiée, le champ texte est redessiné avec la nouvelle feuille de style. La feuille de style peut être définie sur `null` ou `undefined` pour la supprimer. Si la feuille de style en cours d'utilisation est supprimée, le champ texte est redessiné sans feuille de style. La mise en forme effectuée par une feuille de style n'est pas conservée si cette dernière est supprimée.

**Disponibilité :** ActionScript 1.0 ; Flash Player 7

## Exemple

L'exemple suivant crée un champ texte lors de l'exécution, appelé `news_txt`. Trois boutons placés sur la Scène, `css1_btn`, `css2_btn` et `clearCss_btn`, permettent de changer la feuille de style qui s'applique à `news_txt`, ou à supprimer la feuille de style du champ texte. Ajoutez le code ActionScript suivant à votre fichier FLA ou AS :

```
this.createTextField("news_txt", this.getNextHighestDepth(), 0, 0, 300,
  200);
news_txt.wordWrap = true;
news_txt.multiline = true;
news_txt.html = true;
var newsText:String = "<p class='headline'>Description</p> Method; "
  + "starts loading the CSS file into styleSheet. The load operation is
  asynchronous; "
  + "use the <span class='bold'>TextField.StyleSheet.onLoad</span> "
  + "callback handler to determine when the file has finished loading. "
  + "<span class='important'>The CSS file must reside in exactly the same "
  + "domain as the SWF file that is loading it.</span> For more information
  about "
  + "restrictions on loading data across domains, see Flash Player security
  features.";

news_txt.htmlText = newsText;

css1_btn.onRelease = function() {
  var styleObj:TextField.StyleSheet = new TextField.StyleSheet();
  styleObj.onLoad = function(success:Boolean) {
  if (success) {
    news_txt.styleSheet = styleObj;
    news_txt.htmlText = newsText;
  }
  };
  styleObj.load("styles.css");
};

css2_btn.onRelease = function() {
  var styleObj:TextField.StyleSheet = new TextField.StyleSheet();
  styleObj.onLoad = function(success:Boolean) {
  if (success) {
```

```
    news_txt.styleSheet = styleObj;
    news_txt.htmlText = newsText;
  }
  };
  styleObj.load("styles2.css");
};

clearCss_btn.onRelease = function() {
  news_txt.styleSheet = undefined;
  news_txt.htmlText = newsText;
};
```

Les styles suivants sont appliqués au champ texte. Enregistrez les deux fichiers CSS dans le même répertoire que le fichier FLA ou AS que vous avez créé auparavant :

```
// in styles.css
.important {
  color: #FF0000;
}
.bold {
  font-weight: bold;
}
.headline {
  color: #000000;
  font-family: Arial,Helvetica,sans-serif;
  font-size: 18px;
  font-weight: bold;
  display: block;
}

// in styles2.css
.important {
  color: #FF00FF;
}
.bold {
  font-weight: bold;
}
.headline {
  color: #00FF00;
  font-family: Arial,Helvetica,sans-serif;
  font-size: 18px;
  font-weight: bold;
  display: block;
}
```

Si votre fichier SWF comporte un composant de la version 2, utilisez la classe DepthManager des composants de la version 2 au lieu de la méthode `MovieClip.getNextHighestDepth()`, utilisée dans cet exemple.

### Voir également

StyleSheet (TextField.StyleSheet)

## tabEnabled (propriété TextField.tabEnabled)

```
public tabEnabled : Boolean
```

Spécifie si le champ texte est inclus dans l'ordre de tabulation automatique. La valeur par défaut est `undefined`.

Si la propriété `tabEnabled` est définie sur `undefined` ou `true`, l'objet est inclus dans l'ordre de tabulation automatique. Si la propriété `tabIndex` est également définie sur une valeur, l'objet est également inclus dans l'ordre de tabulation personnalisé. Si `tabEnabled` est `false`, l'objet n'est pas inclus dans l'ordre de tabulation automatique ou personnalisé, même si la propriété `tabIndex` est définie.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

### Exemple

L'exemple suivant crée deux champs texte pendant l'exécution, appelés `one_txt`, `two_txt`, `three_txt` et `four_txt`. Le champ texte `three_txt` a sa propriété `tabEnabled` définie sur `false`, ce qui entraîne son exclusion de la séquence de tabulation.

```
this.createTextField("one_txt", this.getNextHighestDepth(), 10, 10, 100,
  22);
one_txt.border = true;
one_txt.type = "input";
this.createTextField("two_txt", this.getNextHighestDepth(), 10, 40, 100,
  22);
two_txt.border = true;
two_txt.type = "input";
this.createTextField("three_txt", this.getNextHighestDepth(), 10, 70, 100,
  22);
three_txt.border = true;
three_txt.type = "input";
this.createTextField("four_txt", this.getNextHighestDepth(), 10, 100, 100,
  22);
four_txt.border = true;
four_txt.type = "input";

three_txt.tabEnabled = false;
three_txt.text = "tabEnabled = false;";
```

La méthode `MovieClip.getNextHighestDepth()` utilisée dans cet exemple requiert Flash Player 7 ou version ultérieure. Si votre fichier SWF comporte un composant de la version 2, utilisez la classe DepthManager des composants de la version 2 au lieu de la méthode `MovieClip.getNextHighestDepth()`.

### Voir également

tabEnabled (propriété Button.tabEnabled), tabEnabled (propriété MovieClip.tabEnabled)

## tabIndex (propriété TextField.tabIndex)

```
public tabIndex : Number
```

Permet de personnaliser l'ordre de tabulation des objets dans un fichier SWF. Vous pouvez définir la propriété `tabIndex` sur un bouton, clip ou sur une occurrence de champ texte ; sa valeur par défaut est `undefined`.

Si un objet actuellement affiché dans le fichier SWF contient une propriété `tabIndex` l'ordre de tabulation automatique est désactivé : l'ordre de tabulation est alors calculé à partir des propriétés `tabIndex` des objets contenus dans le fichier SWF. L'ordre de tabulation personnalisé inclut uniquement des objets dotés de propriétés `tabIndex`.

La propriété `tabIndex` doit être un entier positif.. Les objets sont triés selon leurs propriétés `tabIndex` par ordre croissant. Un objet d'une valeur `tabIndex` de `1` précède un objet de valeur `tabIndex` de 2. Si deux objets ont la même valeur `tabIndex`, celui qui précède l'autre dans l'ordre de tabulation est `undefined`.

L'ordre de tabulation personnalisé défini par la propriété `tabIndex` est *plat.* Cela signifie qu'on ne prête aucune attention aux relations hiérarchiques des objets contenus dans le fichier SWF. Tous les objets du fichier SWF dotés de propriétés `tabIndex` sont placés dans l'ordre de tabulation, qui est déterminé par l'ordre des valeurs `tabIndex`. Si deux objets ont la même valeur `tabIndex`, celui qui apparaît en premier est `undefined`. Il est recommandé de ne pas affecter la même valeur `tabIndex` à plusieurs objets.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

### Exemple

L'exemple de code ActionScript suivant crée de façon dynamique quatre champs texte et les associe à un ordre de tabulation personnalisé. Ajoutez le code ActionScript suivant à votre fichier FLA ou AS :

```
this.createTextField("one_txt", this.getNextHighestDepth(), 10, 10, 100,
  22);
one_txt.border = true;
one_txt.type = "input";
this.createTextField("two_txt", this.getNextHighestDepth(), 10, 40, 100,
  22);
two_txt.border = true;
two_txt.type = "input";
```

```
this.createTextField("three_txt", this.getNextHighestDepth(), 10, 70, 100,
  22);
three_txt.border = true;
three_txt.type = "input";
this.createTextField("four_txt", this.getNextHighestDepth(), 10, 100, 100,
  22);
four_txt.border = true;
four_txt.type = "input";

one_txt.tabIndex = 3;
two_txt.tabIndex = 1;
three_txt.tabIndex = 2;
four_txt.tabIndex = 4;
```

La méthode `MovieClip.getNextHighestDepth()` utilisée dans cet exemple requiert Flash Player 7 ou version ultérieure. Si votre fichier SWF comporte un composant de la version 2, utilisez la classe DepthManager des composants de la version 2 au lieu de la méthode `MovieClip.getNextHighestDepth()`.

### Voir également

`tabIndex (propriété Button.tabIndex)`, `tabIndex (propriété MovieClip.tabIndex)`

## _target (propriété TextField._target)

```
public _target : String [lecture seule]
```

Le chemin cible de l'occurrence du champ texte. La cible `_self` spécifie l'image actuelle dans la fenêtre actuelle, `_blank` spécifie une nouvelle fenêtre, `_parent` spécifie le parent de l'image actuelle et `_top` spécifie l'image de plus haut niveau dans la fenêtre active.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

### Exemple

Le code ActionScript suivant crée un champ texte appelé `my_txt` et renvoie le chemin cible du nouveau champ en notation à barre oblique aussi bien que point.

```
this.createTextField("my_txt", this.getNextHighestDepth(), 10, 10, 100,
  22);
trace(my_txt._target); // output: /my_txt
trace(eval(my_txt._target)); // output: _level0.my_txt
```

La méthode `MovieClip.getNextHighestDepth()` utilisée dans cet exemple requiert Flash Player 7 ou version ultérieure. Si votre fichier SWF comporte un composant de la version 2, utilisez la classe DepthManager des composants de la version 2 au lieu de la méthode `MovieClip.getNextHighestDepth()`.

# text (propriété TextField.text)

```
public text : String
```

Indique le texte actuel dans le champ texte. Les lignes sont séparées par le caractère de retour chariot ("\r", ASCII 13). Cette propriété contient le texte normal, non mis en forme dans le champ texte, sans balises HTML, même si le champ texte est HTML.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

## Exemple

L'exemple suivant crée un champ texte HTML appelé `my_txt`, et lui associe une chaîne de texte au format HTML. Lorsque vous appliquez une instruction trace à la propriété `htmlText` le panneau de sortie affiche la chaîne au format HTML. Lorsque vous appliquez une instruction trace à la propriété `text` la chaîne non formatée avec des balises HTML s'affiche dans le panneau de sortie.

```
this.createTextField("my_txt", this.getNextHighestDepth(), 10, 10, 400,
  22);
my_txt.html = true;
my_txt.htmlText = "<B>Lorem ipsum dolor sit amet.</B>";

trace("htmlText: "+my_txt.htmlText);
trace("text: "+my_txt.text);
```

Ceci génère la sortie suivante :

```
htmlText: <P ALIGN="LEFT"><FONT FACE="Times New Roman" SIZE="12"
  COLOR="#000000" KERNING="0">
<B>Lorem ipsum dolor sit amet.</B></FONT></P>
text: Lorem ipsum dolor sit amet.
```

La méthode `MovieClip.getNextHighestDepth()` utilisée dans cet exemple requiert Flash Player 7 ou version ultérieure. Si votre fichier SWF comporte un composant de la version 2, utilisez la classe DepthManager des composants de la version 2 au lieu de la méthode `MovieClip.getNextHighestDepth()`.

## Voir également

`htmlText (propriété TextField.htmlText)`

# textColor (propriété TextField.textColor)

```
public textColor : Number
```

Indique la couleur du texte dans un champ texte. Le système de couleur hexadécimal utilise les six chiffres pour représenter les valeurs de couleur. Chaque chiffre comporte seize valeurs ou caractères possibles. Les caractères 0 à 9 et A à F sont utilisés. Le noir est représenté par (`#000000`) et le blanc, à l'opposé du système de couleurs, est (`#FFFFFF`).

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

## Exemple

Le code ActionScript suivant crée un champ texte et applique une propriété de couleur rouge.

```
this.createTextField("my_txt", 99, 10, 10, 100, 300);
my_txt.text = "this will be red text";
my_txt.textColor = 0xFF0000;
```

La méthode `MovieClip.getNextHighestDepth()` utilisée dans cet exemple requiert Flash Player 7 ou version ultérieure. Si votre fichier SWF comporte un composant de la version 2, utilisez la classe DepthManager des composants de la version 2 au lieu de la méthode `MovieClip.getNextHighestDepth()`.

# textHeight (propriété TextField.textHeight)

```
public textHeight : Number
```

Indique la hauteur du texte.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

## Exemple

L'exemple suivant crée un champ texte et lui associe une chaîne de texte. Une instruction trace permet d'afficher la hauteur et la largeur du texte dans le panneau de sortie. La propriété `autoSize` est ensuite utilisée pour redimensionner le champ texte et la nouvelle hauteur et largeur s'affichent également dans le panneau de sortie.

```
this.createTextField("my_txt", 99, 10, 10, 100, 300);
my_txt.text = "Sample text";
trace("textHeight: "+my_txt.textHeight+", textWidth: "+my_txt.textWidth);
trace("_height: "+my_txt._height+", _width: "+my_txt._width+"\n");
my_txt.autoSize = true;
trace("after my_txt.autoSize = true;");
trace("_height: "+my_txt._height+", _width: "+my_txt._width);
```

Ce qui permet de renvoyer les informations suivantes :

```
textHeight: 15, textWidth: 56
_height: 300, _width: 100
```

```
after my_txt.autoSize = true;
_height: 19, _width: 60
```

### Voir également

textWidth (propriété TextField.textWidth)

## textWidth (propriété TextField.textWidth)

```
public textWidth : Number
```

Indique la largeur du texte.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

### Exemple

Consultez l'exemple relatif à `TextField.textHeight`.

### Voir également

textHeight (propriété TextField.textHeight)

## thickness (propriété TextField.thickness)

```
public thickness : Number
```

L'épaisseur des bords du glyphe dans cette occurrence TextField. Cette propriété ne s'applique que si `antiAliasType()` est définie sur `"advanced"`.

La plage pour `thickness` est un nombre compris entre -200 et 200. Si vous tentez de définir `thickness` sur une valeur non comprise dans cette plage, la propriété est définie sur la valeur la plus proche dans la plage (-200 ou 200).

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

### Exemple

Cet exemple crée deux champs texte et applique une épaisseur `thickness` de -200 à l'un et de 200 à l'autre. Il suppose que vous avez une police intégrée à la librairie avec l'identifiant de liaison défini sur `"Times-12"`. Pour intégrer la police, procédez comme suit :

- Ouvrez votre Bibliothèque
- Cliquez sur le menu d'options de la Bibliothèque dans le coin supérieur droit de la Bibliothèque
- Sélectionnez « Nouvelle Police » dans la liste déroulante
- Nommez la police « Times-12 »

- Sélectionnez « Times New Roman » dans le menu déroulant
- Appuyez sur le bouton « OK »
- Cliquez du bouton droit sur la nouvelle police créée et sélectionnez « Liaison... »
- Cochez la case « Exporter pour ActionScript »
- Acceptez l'identifiant par défaut « Times-12 » en appuyant sur le bouton « OK »

```
var my_format:TextFormat = new TextFormat();
my_format.font = "Times-12";

var my_text1:TextField = this.createTextField("my_text1",
  this.getNextHighestDepth(), 10, 10, 300, 30);
my_text1.text = "thickness = 200";
my_text1.antiAliasType = "advanced";
my_text1.border = true;
my_text1.thickness = 200;
my_text1.embedFonts = true;
my_text1.setTextFormat(my_format);

var my_text2:TextField = this.createTextField("my_text2",
  this.getNextHighestDepth(), 10, 50, 300, 30);
my_text2.text = "thickness = -200."
my_text2.antiAliasType = "advanced";
my_text2.thickness = -200;
my_text2.border = true;
my_text2.embedFonts = true;
my_text2.setTextFormat(my_format);
```

Si votre fichier SWF comporte un composant de la version 2, utilisez la classe DepthManager des composants de la version 2 au lieu de la méthode `MovieClip.getNextHighestDepth()`, utilisée dans cet exemple.

### Voir également

`antiAliasType (propriété TextField.antiAliasType)`

## type (propriété TextField.type)

```
public type : String
```

Spécifie le type de champ texte. Il existe deux valeurs : `"dynamic"`, qui spécifie un champ texte dynamique qui ne peut pas être modifié par l'utilisateur et `"input"`, qui spécifie un champ texte de saisie.

**Disponibilité :** ActionScript 1.0 ; ActionScript 1.0 ; Flash Player 6

### Exemple

L'exemple suivant crée deux champs texte : `username_txt` et `password_txt`. Du texte est placé dans les deux champs texte ; toutefois, `password_txt` a la propriété `password` définie sur `true`. Par conséquent, les caractères s'affichent sous forme d'astérisques dans le champ `password_txt`.

```
this.createTextField("username_txt", this.getNextHighestDepth(), 10, 10,
  100, 22);
username_txt.border = true;
username_txt.type = "input";
username_txt.maxChars = 16;
username_txt.text = "hello";

this.createTextField("password_txt", this.getNextHighestDepth(), 10, 40,
  100, 22);
password_txt.border = true;
password_txt.type = "input";
password_txt.maxChars = 16;
password_txt.password = true;
password_txt.text = "world";
```

La méthode `MovieClip.getNextHighestDepth()` utilisée dans cet exemple requiert Flash Player 7 ou version ultérieure. Si votre fichier SWF comporte un composant de la version 2, utilisez la classe DepthManager des composants de la version 2 au lieu de la méthode `MovieClip.getNextHighestDepth()`.

## _url (propriété TextField._url)

```
public _url : String [lecture seule]
```

Récupère l'URL du fichier SWF qui a créé le champ texte.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

### Exemple

L'exemple suivant extrait l'URL du fichier SWF qui a créé le champ texte et un fichier SWF qui se charge dans ce dernier.

```
this.createTextField("my_txt", 1, 10, 10, 100, 22);
trace(my_txt._url);

var mclListener:Object = new Object();
mclListener.onLoadInit = function(target_mc:MovieClip) {
  trace(target_mc._url);
};
var holder_mcl:MovieClipLoader = new MovieClipLoader();
holder_mcl.addListener(mclListener);
```

```
holder_mcl.loadClip("best_flash_ever.swf",
  this.createEmptyMovieClip("holder_mc", 2));
```

Lorsque vous testez cet exemple, l'URL du fichier SWF que vous testez et le fichier appelé best_flash_ever.swf s'affichent dans le panneau de sortie.

La méthode MovieClipLoader utilisée dans cet exemple requiert Flash Player 7 ou version ultérieure.

# variable (propriété TextField.variable)

```
public variable : String
```

Nom de la variable à laquelle le champ texte est associé. Le type de cette propriété est String.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

## Exemple

L'exemple suivant crée un champ texte appelé `my_txt` et associe la variable `today_date` au champ texte. Lorsque vous modifiez la variable `today_date`, le texte s'affiche dans les mises à jour de `my_txt`.

```
this.createTextField("my_txt", 1, 10, 10, 200, 22);
my_txt.variable = "today_date";
var today_date:Date = new Date();

var date_interval:Number = setInterval(updateDate, 500);
function updateDate():Void {
  today_date = new Date();
}
```

# _visible (propriété TextField._visible)

```
public _visible : Boolean
```

Valeur booléenne indiquant si le champ texte *`my_txt`* est visible. Les champs texte qui ne sont pas visibles (`_visible` définie sur `false`) sont désactivés.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

## Exemple

L'exemple suivant crée un champ texte appelé `my_txt`. Un bouton appelé `visible_btn` permet de faire basculer la visibilité de `my_txt`.

```
this.createTextField("my_txt", 1, 10, 10, 200, 22);
my_txt.background = true;
my_txt.backgroundColor = 0xDFDFDF;
my_txt.border = true;
my_txt.type = "input";
```

```
visible_btn.onRelease = function() {
  my_txt._visible = !my_txt._visible;
};
```

### Voir également

_visible (propriété Button._visible), _visible (propriété MovieClip._visible)

## _width (propriété TextField._width)

```
public _width : Number
```

Largeur du champ texte, en pixels.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

### Exemple

L'exemple suivant crée deux fichiers texte que vous pouvez utiliser pour modifier la largeur et la hauteur d'un troisième champ texte sur la Scène. Ajoutez le code ActionScript suivant à un fichier FLA ou AS.

```
this.createTextField("my_txt", this.getNextHighestDepth(), 10, 40, 160,
  120);
my_txt.background = true;
my_txt.backgroundColor = 0xFF0000;
my_txt.border = true;
my_txt.multiline = true;
my_txt.type = "input";
my_txt.wordWrap = true;

this.createTextField("width_txt", this.getNextHighestDepth(), 10, 10, 30,
  20);
width_txt.border = true;
width_txt.maxChars = 3;
width_txt.restrict = "0-9";
width_txt.type = "input";
width_txt.text = my_txt._width;
width_txt.onChanged = function() {
  my_txt._width = this.text;
}

this.createTextField("height_txt", this.getNextHighestDepth(), 70, 10, 30,
  20);
height_txt.border = true;
height_txt.maxChars = 3;
height_txt.restrict = "0-9";
height_txt.type = "input";
```

```
height_txt.text = my_txt._height;
height_txt.onChanged = function() {
  my_txt._height = this.text;
}
```

Lorsque vous testez cet exemple, essayez de saisir de nouvelles valeurs dans `width_txt` et `height_txt` pour modifier les dimensions de `my_txt`.

La méthode `MovieClip.getNextHighestDepth()` utilisée dans cet exemple requiert Flash Player 7 ou version ultérieure. Si votre fichier SWF comporte un composant de la version 2, utilisez la classe DepthManager des composants de la version 2 au lieu de la méthode `MovieClip.getNextHighestDepth()`.

### Voir également

`_height (propriété TextField._height)`

## wordWrap (propriété TextField.wordWrap)

```
public wordWrap : Boolean
```

Valeur booléenne indiquant si le champ texte comporte un retour à la ligne. Si la valeur de `wordWrap` est `true`, le champ texte comporte un retour à la ligne ; si la valeur est `false`, le champ texte ne comporte pas de retour à la ligne.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

### Exemple

L'exemple suivant démontre comment `wordWrap` affecte du texte long à un champ texte qui est créé pendant l'exécution.

```
this.createTextField("my_txt", 99, 10, 10, 100, 200);
my_txt.text = "This is very long text that will certainly extend beyond the
  width of this text field";
my_txt.border = true;
```

Testez le fichier SWF dans Flash Player en choisissant Contrôle > Tester l'animation. Revenez ensuite au code ActionScript et ajoutez la ligne suivante au code et testez de nouveau le fichier SWF :

```
my_txt.wordWrap = true;
```

# _x (propriété TextField._x)

```
public _x : Number
```

Entier qui définit la coordonnée x d'un champ texte par rapport aux coordonnées locales du clip parent. Si un champ texte se trouve sur le scénario principal, son système de coordonnées se réfère alors au coin supérieur gauche de la Scène : (0, 0). Si le champ texte est imbriqué dans un clip subissant des transformations, le champ texte se trouve dans le système de coordonnées locales du clip qui l'encadre. Ainsi, dans le cas d'un clip qui a effectué une rotation à 90 degrés en sens anti-horaire, le champ texte imbriqué hérite d'un système de coordonnées ayant effectué une rotation à 90 degrés en sens anti-horaire. Les coordonnées du champ texte renvoient à la position du point d'alignement.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

## Exemple

L'exemple suivant crée un champ texte lorsque vous cliquez sur la souris. Lorsqu'il crée un champ texte, ce champ affiche les coordonnées *x* et *y* actuelles du champ texte.

```
this.createTextField("coords_txt", this.getNextHighestDepth(), 0, 0, 60,
  22);
coords_txt.autoSize = true;
coords_txt.selectable = false;
coords_txt.border = true;

var mouseListener:Object = new Object();
mouseListener.onMouseDown = function() {
  coords_txt.text = "X:"+Math.round(_xmouse)+", Y:"+Math.round(_ymouse);
  coords_txt._x = _xmouse;
  coords_txt._y = _ymouse;
};
Mouse.addListener(mouseListener);
```

La méthode `MovieClip.getNextHighestDepth()` utilisée dans cet exemple requiert Flash Player 7 ou version ultérieure. Si votre fichier SWF comporte un composant de la version 2, utilisez la classe DepthManager des composants de la version 2 au lieu de la méthode `MovieClip.getNextHighestDepth()`.

## Voir également

_xscale (propriété TextField._xscale), _y (propriété TextField._y), _yscale (propriété TextField._yscale)

# _xmouse (propriété TextField._xmouse)

`public _xmouse : Number [lecture seule]`

Renvoie la coordonnée x de la position de la souris par rapport au champ texte.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

## Exemple

L'exemple suivant crée trois champs texte sur la Scène. L'occurrence `mouse_txt` affiche la position actuelle de la souris par rapport à la Scène. L'occurrence `textfield_txt` affiche la position actuelle du pointeur de la souris par rapport à l'occurrence `my_txt`. Ajoutez le code ActionScript suivant à un fichier FLA ou AS :

```
this.createTextField("mouse_txt", this.getNextHighestDepth(), 10, 10, 200,
  22);
mouse_txt.border = true;
this.createTextField("textfield_txt", this.getNextHighestDepth(), 220, 10,
  200, 22);
textfield_txt.border = true;
this.createTextField("my_txt", this.getNextHighestDepth(), 100, 100, 160,
  120);
my_txt.border = true;

var mouseListener:Object = new Object();
mouseListener.onMouseMove = function() {
  mouse_txt.text = "MOUSE ... X:" + Math.round(_xmouse) + ",\tY:" +
  Math.round(_ymouse);
  textfield_txt.text = "TEXTFIELD ... X:" + Math.round(my_txt._xmouse) +
  ",\tY:" +
  Math.round(my_txt._ymouse);
}

Mouse.addListener(mouseListener);
```

La méthode `MovieClip.getNextHighestDepth()` utilisée dans cet exemple requiert Flash Player 7 ou version ultérieure. Si votre fichier SWF comporte un composant de la version 2, utilisez la classe DepthManager des composants de la version 2 au lieu de la méthode `MovieClip.getNextHighestDepth()`.

## Voir également

`_ymouse (propriété TextField._ymouse)`

# _xscale (propriété TextField._xscale)

```
public _xscale : Number
```

Détermine le redimensionnement horizontal du champ texte tel qu'il est appliqué à partir du point d'alignement du champ texte, exprimé en pourcentage. Le point d'alignement par défaut est (0,0).

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

## Exemple

L'exemple suivant redimensionne l'occurrence `my_txt` lorsque vous cliquez sur les occurrences `scaleUp_btn` et `scaleDown_btn`.

```
this.createTextField("my_txt", 99, 10, 40, 100, 22);
my_txt.autoSize = true;
my_txt.border = true;
my_txt.selectable = false;
my_txt.text = "Sample text goes here.";

scaleUp_btn.onRelease = function() {
  my_txt._xscale = 2;
  my_txt._yscale = 2;
}
scaleDown_btn.onRelease = function() {
  my_txt._xscale /= 2;
  my_txt._yscale /= 2;
}
```

## Voir également

_x (propriété TextField._x), _y (propriété TextField._y), _yscale (propriété TextField._yscale)

# _y (propriété TextField._y)

```
public _y : Number
```

Coordonnée y d'un champ texte par rapport aux coordonnées locales du clip parent. Si un champ texte se trouve dans le scénario principal, son système de coordonnées se réfère alors au coin supérieur gauche de la Scène : (0, 0). Si le champ texte est imbriqué dans un autre clip subissant des transformations, le champ texte se trouve dans le système de coordonnées locales du clip qui l'encadre. Ainsi, dans le cas d'un clip qui a effectué une rotation à 90 degrés en sens anti-horaire, le champ texte imbriqué hérite d'un système de coordonnées ayant effectué une rotation à 90 degrés en sens anti-horaire. Les coordonnées du champ texte renvoient à la position du point d'alignement.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

### Exemple

Consultez l'exemple relatif à `TextField._x`.

### Voir également

`_x (propriété TextField._x)`, `_xscale (propriété TextField._xscale)`, `_yscale (propriété TextField._yscale)`

## _ymouse (propriété TextField._ymouse)

```
public _ymouse : Number [lecture seule]
```

Indique la coordonnée y de la position de la souris par rapport au champ texte.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

### Exemple

Consultez l'exemple relatif à `TextField._xmouse`.

### Voir également

`_xmouse (propriété TextField._xmouse)`

## _yscale (propriété TextField._yscale)

```
public _yscale : Number
```

Redimensionnement vertical du champ texte tel qu'il est appliqué à partir du point d'alignement du champ texte, exprimé en pourcentage. Le point d'alignement par défaut est (0,0).

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

### Exemple

Consultez l'exemple relatif à `TextField._xscale`.

### Voir également

`_x (propriété TextField._x)`, `_xscale (propriété TextField._xscale)`, `_y (propriété TextField._y)`

# TextFormat

```
Object
  |
  +-TextFormat
```

```
public class TextFormat
extends Object
```

La classe TextFormat représente les informations de mise en forme de caractères. Utilisez la classe TextFormat pour personnaliser la mise en forme des champs texte. Vous pouvez formater le texte des champs statiques et dynamiques. Certaines propriétés de la classe TextFormat ne sont pas disponibles à la fois pour les polices incorporées et de périphérique.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

## Voir également

`setTextFormat (méthode TextField.setTextFormat)`, `getTextFormat (méthode TextField.getTextFormat)`

## Résumé des propriétés

| Modificateurs | Propriété | Description |
|---|---|---|
| | `align:String` | Une chaîne indiquant l'alignement du paragraphe. |
| | `blockIndent:Number` | Nombre indiquant l'indentation d'un bloc en points. |
| | `bold:Boolean` | Valeur booléenne indiquant si le texte est en gras. |
| | `bullet:Boolean` | Valeur booléenne indiquant que le texte fait partie d'une liste à puces. |
| | `color:Number` | Indique la couleur du texte. |
| | `font:String` | Nom de la police pour du texte dans ce format de texte, sous forme de chaîne. |
| | `indent:Number` | Entier indiquant l'indentation à appliquer de la marge gauche au premier caractère du paragraphe. |
| | `italic:Boolean` | Valeur booléenne indiquant si le texte dans ce format de texte est en italique. |
| | `kerning:Boolean` | Valeur booléenne indiquant si le crénage est activé ou désactivé. |
| | `leading:Number` | Entier représentant le montant d'espace vertical en pixels (appelé *interlignage*) entre les lignes. |
| | `leftMargin:Number` | Marge gauche du paragraphe, en points. |

| Modificateurs | Propriété | Description |
|---|---|---|
| | `letterSpacing:Number` | La quantité d'espace répartie uniformément entre les caractères. |
| | `rightMargin:Number` | Marge droite du paragraphe, en points. |
| | `size:Number` | Taille en points du texte dans ce format de texte. |
| | `tabStops:Array` | Spécifie des taquets de tabulation personnalisés, sous forme d'un tableau d'entiers non négatifs. |
| | `target:String` | Indique la fenêtre cible dans laquelle s'affiche l'hyperlien. |
| | `underline:Boolean` | Valeur booléenne qui indique si le texte qui utilise ce format texte est souligné (`true`) ou non (`false`). |
| | `url:String` | Indique l'URL à laquelle relie le texte de ce format texte. |

*Propriétés héritées de la classe Object*

`constructeur (propriété Object.constructor), __proto__ (Object.__proto__ property), prototype (Object.prototype, propriété), __resolve (Object.__resolve, propriété)`

## Résumé des constructeurs

| Signature | Description |
|---|---|
| `TextFormat([font:String], [size:Number], [color:Number], [bold:Boolean], [italic:Boolean], [underline:Boolean], [url:String], [target:String], [align:String], [leftMargin:Number], [rightMargin:Number], [indent:Number], [leading:Number])` | Crée un objet TextFormat avec les propriétés spécifiées. |

### Résumé de la méthode

| Modificateurs | Signature | Description |
|---|---|---|
| | `getTextExtent(text:String, [width:Number]) : Object` | *Déconseillé* à partir de Flash Player 8. Il n'existe pas de remplacement.<br>Renvoie les informations de mesure de texte pour la chaîne de texte `text` dans le format spécifié par `my_fmt`. |

*Méthodes héritées de la classe Object*

`addProperty (méthode Object.addProperty), hasOwnProperty (méthode Object.hasOwnProperty), isPropertyEnumerable (méthode Object.isPropertyEnumerable), isPrototypeOf (méthode Object.isPrototypeOf), registerClass (méthode Object.registerClass), toString (méthode Object.toString), unwatch (méthode Object.unwatch), valueOf (méthode Object.valueOf), watch (méthode Object.watch)`

## align (propriété TextFormat.align)

`public align : String`

Une chaîne indiquant l'alignement du paragraphe. Vous pouvez appliquer cette propriété tant aux champs texte statiques qu'aux champs texte dynamiques. Le tableau ci-dessous donne la liste des valeurs possibles pour cette propriété.

- `"left"`-Le paragraphe est aligné à gauche.
- `"center"`-Le paragraphe est centré.
- `"right"`-Le paragraphe est aligné à droite.
- `"justify"`-Le paragraphe est justifié. (Cette valeur a été ajoutée dans Flash Player 8.)

La valeur par défaut est `null`, ce qui indique que la propriété n'est pas définie.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

### Exemple

L'exemple suivant illustre la propriété `align` définie pour la justification, ce qui entraîne un espacement des caractères sur chaque ligne qui donne au texte un espacement horizontal d'un aspect plus régulier.

```
var format:TextFormat = new TextFormat();
format.align = "justify";

var txtField:TextField = this.createTextField("txtField",
  this.getNextHighestDepth(), 100, 100, 300, 100);
txtField.multiline = true;
```

```
txtField.wordWrap = true;
txtField.border = true;
txtField.text = "When this text is justified, it will be "
  + "spread out to more cleanly fill the horizontal "
  + "space for each line. This can be considered an "
  + "improvement over regular left-aligned text that "
  + "will simply wrap and do no more.";
txtField.setTextFormat(format);
```

La méthode `MovieClip.getNextHighestDepth()` utilisée dans cet exemple requiert Flash Player 7 ou version ultérieure. Si votre fichier SWF comporte un composant de la version 2, utilisez la classe DepthManager des composants de la version 2 au lieu de la méthode `MovieClip.getNextHighestDepth()`.

# blockIndent (propriété TextFormat.blockIndent)

```
public blockIndent : Number
```

Nombre indiquant l'indentation d'un bloc en points. L'indentation d'un bloc est appliquée à l'ensemble d'un bloc de texte ; c'est-à-dire à toutes les lignes du texte. Par contraste, l'indentation normale (`TextFormat.indent`) affecte seulement la première ligne de chaque paragraphe. Si cette propriété est `null`, l'objet TextFormat ne spécifie pas l'indentation d'un bloc.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

## Exemple

Cet exemple crée un champ texte avec une bordure et définit blockIndent sur 20.

```
this.createTextField("mytext",1,100,100,100,100);
mytext.multiline = true;
mytext.wordWrap = true;
mytext.border = true;

var myformat:TextFormat = new TextFormat();
myformat.blockIndent = 20;

mytext.text = "This is my first test field object text";
mytext.setTextFormat(myformat);
```

# bold (propriété TextFormat.bold)

```
public bold : Boolean
```

Valeur booléenne indiquant si le texte est en gras. La valeur par défaut est `null`, ce qui indique que la propriété n'est pas définie. Si la valeur est `true`, le texte est en gras.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

### Exemple

L'exemple suivant crée un champ texte qui inclut des caractères en gras.

```
var my_fmt:TextFormat = new TextFormat();
my_fmt.bold = true;

this.createTextField("my_txt", 1, 100, 100, 300, 100);
my_txt.multiline = true;
my_txt.wordWrap = true;
my_txt.border = true;
my_txt.text = "This is my test field object text";
my_txt.setTextFormat(my_fmt);
```

## bullet (propriété TextFormat.bullet)

```
public bullet : Boolean
```

Valeur booléenne indiquant que le texte fait partie d'une liste à puces. Dans une liste à puces, chaque paragraphe du texte apparaît en retrait. A gauche de la première ligne de chaque paragraphe, le symbole d'une puce s'affiche. La valeur par défaut est `null`.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

### Exemple

L'exemple suivant crée un champ texte lors de l'exécution et entre une chaîne avec un saut de ligne dans le champ. La classe TextFormat permet de formater les caractères en ajoutant des puces aux différentes lignes du champ texte. Ceci est illustré par l'exemple ActionScript suivant :

```
var my_fmt:TextFormat = new TextFormat();
my_fmt.bullet = true;

this.createTextField("my_txt", 1, 100, 100, 300, 100);
my_txt.multiline = true;
my_txt.wordWrap = true;
my_txt.border = true;
my_txt.text = "this is my text"+newline;
my_txt.text += "this is more text"+newline;
my_txt.setTextFormat(my_fmt);
```

## color (propriété TextFormat.color)

```
public color : Number
```

Indique la couleur du texte. Un nombre contenant trois composants RVB 8 bits ; par exemple, 0xFF0000 est rouge, et 0x00FF00 est vert.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

### Exemple

L'exemple suivant crée un champ texte et applique une couleur rouge au texte.

```
var my_fmt:TextFormat = new TextFormat();
my_fmt.blockIndent = 20;
my_fmt.color = 0xFF0000; // hex value for red

this.createTextField("my_txt", 1, 100, 100, 300, 100);
my_txt.multiline = true;
my_txt.wordWrap = true;
my_txt.border = true;
my_txt.text = "this is my first test field object text";
my_txt.setTextFormat(my_fmt);
```

## font (propriété TextFormat.font)

```
public font : String
```

Nom de la police pour du texte dans ce format de texte, sous forme de chaîne. La valeur par défaut est `null`, ce qui indique que la propriété n'est pas définie.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

### Exemple

L'exemple suivant crée un champ texte et applique la police Courier.

```
this.createTextField("mytext",1,100,100,100,100);
mytext.multiline = true;
mytext.wordWrap = true;
mytext.border = true;

var myformat:TextFormat = new TextFormat();
myformat.font = "Courier";

mytext.text = "this is my first test field object text";
mytext.setTextFormat(myformat);
```

## getTextExtent (méthode TextFormat.getTextExtent)

```
public getTextExtent(text:String, [width:Number]) : Object
```

*Déconseillé* à partir de Flash Player 8. Il n'existe pas de remplacement.

Renvoie les informations de mesure de texte pour la chaîne de texte `text` dans le format spécifié par `my_fmt`. La chaîne de texte est traitée comme du texte brut (non HTML).

La méthode renvoie un objet avec six propriétés : `ascent`, `descent`, `width`, `height`, `textFieldHeight` et `textFieldWidth`. Toutes les mesures sont en pixels.

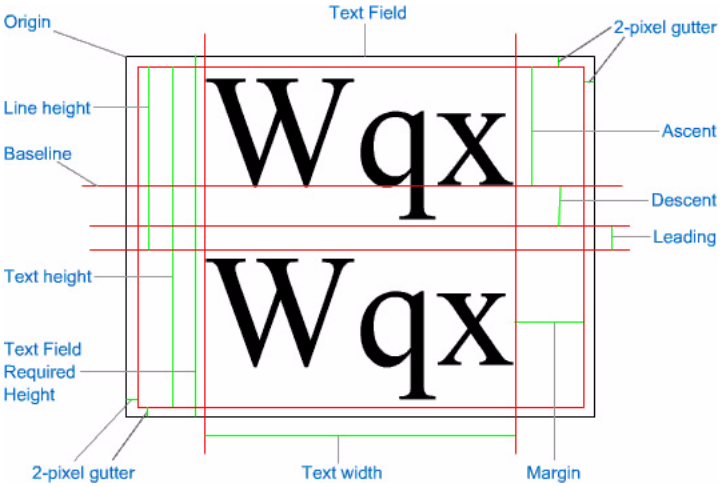
Si un paramètre `width` (largeur) est spécifié, un retour à la ligne automatique est appliqué au texte spécifié. Ceci vous permet de déterminer la hauteur à laquelle un champ affiche l'ensemble du texte spécifié.

Les mesures `ascent` et `descent` indiquent, respectivement, la distance au-dessus et en dessous de la ligne de base pour une ligne de texte. La ligne de base pour la première ligne de texte est positionnée au début du champ de texte plus sa mesure `ascent`.

Les mesures `width` et `height` indiquent la largeur et la hauteur de la chaîne de texte. Les mesures `textFieldHeight` et `textFieldWidth` indiquent la hauteur et la largeur requises pour un objet de champ de texte pour afficher l'ensemble de la chaîne de texte. Les champs de texte ont une marge de reliure de 2 pixels autour d'eux, de sorte que la valeur de `textFieldHeight` est égale à la valeur de `height` + 4; de même, la valeur de `textFieldWidth` est toujours égale à la valeur de `width` + 4.

Si vous créez un champ de texte basé sur les propriétés de texte, utilisez `textFieldHeight` plutôt que `height` et `textFieldWidth` plutôt que `width`.

La figure suivante illustre ces mesures.




Lorsque vous configurez un objet TextFormat, définissez exactement tous les attributs tels qu'ils seront définis pour la création du champ de texte, y compris le nom de police, la taille de police et l'interligne. La valeur par défaut pour l'interligne est 2.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

## Paramètres

*`text`*`:String` - Chaîne.

*width*:Number [facultatif] - Nombre indiquant la largeur, en pixels, à laquelle le texte spécifié doit effectuer un retour à la ligne automatique.

## Valeur renvoyée

Object - Objet avec les propriétés width, height, ascent, descent, textFieldHeight, textFieldWidth.

## Exemple

Cet exemple crée un champ texte d'une seule ligne de taille tout juste suffisante pour afficher une chaîne de texte avec la mise en format spécifiée.

```
var my_str:String = "Small string";

// Create a TextFormat object,
// and apply its properties.
var my_fmt:TextFormat = new TextFormat();
with (my_fmt) {
  font = "Arial";
  bold = true;
}

// Obtain metrics information for the text string
// with the specified formatting.
var metrics:Object = my_fmt.getTextExtent(my_str);

// Create a text field just large enough to display the text.
this.createTextField("my_txt", this.getNextHighestDepth(), 100, 100,
  metrics.textFieldWidth,
metrics.textFieldHeight);
my_txt.border = true;
my_txt.wordWrap = true;
// Assign the same text string and TextFormat object to the my_txt object.
my_txt.text = my_str;
my_txt.setTextFormat(my_fmt);
```

L'exemple suivant crée un champ texte multiligne d'une largeur de 100 pixels et d'une hauteur suffisante pour afficher une chaîne avec la mise en format spécifiée.

```
// Create a TextFormat object.
var my_fmt:TextFormat = new TextFormat();
// Specify formatting properties for the TextFormat object:
my_fmt.font = "Arial";
my_fmt.bold = true;
my_fmt.leading = 4;

// The string of text to be displayed
var textToDisplay:String = "Macromedia Flash Player 7, now with improved
  text metrics.";
```

```
// Obtain text measurement information for the string,
// wrapped at 100 pixels.
var metrics:Object = my_fmt.getTextExtent(textToDisplay, 100);

// Create a new TextField object using the metric
// information just obtained.
this.createTextField("my_txt", this.getNextHighestDepth(), 50, 50-
  metrics.ascent, 100,
metrics.textFieldHeight);
my_txt.wordWrap = true;
my_txt.border = true;
// Assign the text and the TextFormat object to the TextObject:
my_txt.text = textToDisplay;
my_txt.setTextFormat(my_fmt);
```

# indent (propriété TextFormat.indent)

```
public indent : Number
```

Entier indiquant l'indentation à appliquer de la marge gauche au premier caractère du paragraphe. Une valeur positive indique une indentation normale. Vous pouvez utiliser une valeur négative, mais l'indentation négative s'applique uniquement si la marge de gauche est supérieure à 0. Pour définir une marge supérieure à 0, utilisez la propriété `indent` ou la propriété `blockIndent` de l'objet TextFormat. La valeur par défaut est `null`, ce qui indique que la propriété n'est pas définie.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

## Exemple

L'exemple suivant crée un champ texte et définit l'indentation sur 10 :

```
this.createTextField("mytext",1,100,100,100,100);
mytext.multiline = true;
mytext.wordWrap = true;
mytext.border = true;

var myformat:TextFormat = new TextFormat();
myformat.indent = 10;

mytext.text = "this is my first test field object text";
mytext.setTextFormat(myformat);
```

## Voir également

`blockIndent (propriété TextFormat.blockIndent)`

# italic (propriété TextFormat.italic)

```
public italic : Boolean
```

Valeur booléenne indiquant si le texte dans ce format de texte est en italique. La valeur par défaut est `null`, ce qui indique que la propriété n'est pas définie.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

## Exemple

L'exemple suivant crée un champ texte et met le texte en italique.

```
this.createTextField("mytext",1,100,100,100,100);
mytext.multiline = true;
mytext.wordWrap = true;
mytext.border = true;

var myformat:TextFormat = new TextFormat();
myformat.italic = true;

mytext.text = "This is my first text field object text";
mytext.setTextFormat(myformat);
```

# kerning (propriété TextFormat.kerning)

```
public kerning : Boolean
```

Valeur booléenne indiquant si le crénage est activé ou désactivé. Le crénage consiste à régler l'espace séparant certaines paires de caractères pour en améliorer la lisibilité. La valeur par défaut est false `false`, ce qui indique que le crénage est désactivé.

Le crénage n'est pris en charge que pour les polices incorporées. Certaines polices, telles que Courier New, ne prennent pas en charge le crénage.

La propriété de crénage `kerning` n'est prise en charge que dans les fichiers SWF créés dans Windows, pas dans les fichiers SWF créés sur Macintosh. Cependant, les fichiers SWF Windows *peuvent* être lus dans des versions de Flash Player autres que Windows et le crénage s'applique encore.

N'utilisez cette fonction qu'en cas de nécessité, dans les titres avec des polices de grande taille par exemple.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

### Exemple

L'exemple suivant présente deux champs texte : Le format du premier utilise du texte rouge avec `kerning` défini sur `false` et le format du deuxième utilise du texte bleu avec `kerning` défini sur `true`. Pour utiliser cet exemple, ajoutez un symbole de police à la Bibliothèque et sélectionnez ensuite Arial comme police. Dans la boîte de dialogue Propriétés de Liaison pour la police, définissez le nom de l'Identifiant sur `"Font 1"`, sélectionnez Export pour ActionScript puis sélectionnez Export dans First Frame.

```
var fmt1:TextFormat = new TextFormat();
fmt1.font = "Font 1";
fmt1.size = 50;
fmt1.color = 0xFF0000;
fmt1.kerning = false;

var fmt2:TextFormat = new TextFormat();
fmt2.font = "Font 1";
fmt2.size = 50;
fmt2.color = 0x0000FF;
fmt2.kerning = true;

this.createTextField("tf1", this.getNextHighestDepth(), 10, 10, 400, 100);
tf1.embedFonts = true;
tf1.text = "Text 7AVA-7AVA";
tf1.setTextFormat(fmt1);

this.createTextField("tf2", this.getNextHighestDepth(), 10, 40, 400, 100);
tf2.embedFonts = true;
tf2.text = tf1.text;
tf2.setTextFormat(fmt2);
```

Si votre fichier SWF comporte un composant de la version 2, utilisez la classe DepthManager des composants de la version 2 au lieu de la méthode `MovieClip.getNextHighestDepth()`, utilisée dans cet exemple.

## leading (propriété TextFormat.leading)

```
public leading : Number
```

Entier représentant le montant d'espace vertical en pixels (appelé *interlignage*) entre les lignes. La valeur par défaut est `null`, ce qui indique que la propriété n'est pas définie.

Flash Player 8 prend en change l'*interlignage négatif*, ce qui signifie que l'espacement des lignes est inférieur à la hauteur du texte. L'interlignage négatif peut être utile lorsque vous devez rapprocher des lignes de texte, dans des titres par exemple. Pour prévenir les chevauchements, appliquez l'interlignage négatif aux lignes qui ne contiennent pas de descendantes, notamment les lignes contenant uniquement des majuscules.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

### Exemple

L'exemple suivant crée un champ texte et définit l'interlignage sur 10.

```
var my_fmt:TextFormat = new TextFormat();
my_fmt.leading = 10;

this.createTextField("my_txt", 1, 100, 100, 100, 100);
my_txt.multiline = true;
my_txt.wordWrap = true;
my_txt.border = true;
my_txt.text = "This is my first text field object text";
my_txt.setTextFormat(my_fmt);
```

## leftMargin (propriété TextFormat.leftMargin)

```
public leftMargin : Number
```

Marge gauche du paragraphe, en points. La valeur par défaut est `null`, ce qui indique que la propriété n'est pas définie.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

### Exemple

L'exemple suivant crée un champ texte et définit la marge gauche sur 20 points.

```
this.createTextField("mytext",1,100,100,100,100);
mytext.multiline = true;
mytext.wordWrap = true;
mytext.border = true;

var myformat:TextFormat = new TextFormat();
myformat.leftMargin = 20;

mytext.text = "this is my first test field object text";
mytext.setTextFormat(myformat);
```

## letterSpacing (propriété TextFormat.letterSpacing)

```
public letterSpacing : Number
```

La quantité d'espace répartie uniformément entre les caractères. Le nombre spécifie le nombre de pixels ajoutés à l'espacement après chaque caractère. Une valeur négative comprime l'espace entre les caractères.

Les polices du système ne prennent en charge que les valeurs entières ; toutefois, pour les polices intégrées, vous pouvez spécifier des valeurs (non entières) en virgule flottante (par ex. 2.6).

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

### Exemple

L'exemple de code suivant utilise deux objets TextFormat pour appliquer des valeurs positives et négatives de l'espacement `letterSpacing` à différentes plages de texte dans un champ texte.

```
this.createTextField("mytext", this.getNextHighestDepth(), 10, 10, 200,
  100);
mytext.multiline = true;
mytext.wordWrap = true;
mytext.border = true;

var format1:TextFormat = new TextFormat();
format1.letterSpacing = -1;

var format2:TextFormat = new TextFormat();
format2.letterSpacing = 10;

mytext.text = "Eat at \nJOE'S.";
mytext.setTextFormat(0, 7, format1);
mytext.setTextFormat(8, 12, format2);
```

Si votre fichier SWF comporte un composant de la version 2, utilisez la classe DepthManager des composants de la version 2 au lieu de la méthode `MovieClip.getNextHighestDepth()`, utilisée dans cet exemple.

## rightMargin (propriété TextFormat.rightMargin)

```
public rightMargin : Number
```

Marge droite du paragraphe, en points. La valeur par défaut est `null`, ce qui indique que la propriété n'est pas définie.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

### Exemple

L'exemple suivant crée un champ texte et définit la marge droite sur 20 points.

```
this.createTextField("mytext",1,100,100,100,100);
mytext.multiline = true;
mytext.wordWrap = true;
mytext.border = true;
```

```
var myformat:TextFormat = new TextFormat();
myformat.rightMargin = 20;

mytext.text = "this is my first test field object text";
mytext.setTextFormat(myformat);
```

## size (propriété TextFormat.size)

```
public size : Number
```

Taille en points du texte dans ce format de texte. La valeur par défaut est `null`, ce qui indique que la propriété n'est pas définie.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

### Exemple

L'exemple suivant crée un champ texte et définit la taille du texte sur 20 points.

```
this.createTextField("mytext",1,100,100,100,100);
mytext.multiline = true;
mytext.wordWrap = true;
mytext.border = true;

var myformat:TextFormat = new TextFormat();
myformat.size = 20;

mytext.text = "This is my first text field object text";
mytext.setTextFormat(myformat);
```

## tabStops (propriété TextFormat.tabStops)

```
public tabStops : Array
```

Spécifie des taquets de tabulation personnalisés, sous forme d'un tableau d'entiers non négatifs. Chaque taquet de tabulation est spécifié en pixels. Si des taquets de tabulation personnalisés ne sont pas spécifiés (`null`), le taquet de tabulation par défaut est 4 (largeur moyenne de caractère).

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

### Exemple

L'exemple suivant crée deux champs texte, dont l'un contient des taquets de tabulation tous les 40 pixels et l'autre tous les 75 pixels.

```
this.createTextField("mytext",1,100,100,400,100);
mytext.border = true;
var myformat:TextFormat = new TextFormat();
myformat.tabStops = [40,80,120,160];
```

```
mytext.text = "A\tB\tC\tD"; // \t is the tab stop character
mytext.setTextFormat(myformat);

this.createTextField("mytext2",2,100,220,400,100);
mytext2.border = true;
var myformat2:TextFormat = new TextFormat();
myformat2.tabStops = [75,150,225,300];
mytext2.text ="A\tB\tC\tD";
mytext2.setTextFormat(myformat2);
```

# target (propriété TextFormat.target)

```
public target : String
```

Indique la fenêtre cible dans laquelle s'affiche l'hyperlien. Si la fenêtre cible est une chaîne vide, le texte s'affiche dans la fenêtre cible par défaut `_self`. Vous pouvez choisir un nom personnalisé ou l'un des quatre noms suivants : `_self` spécifie l'image actuelle dans la fenêtre actuelle, `_blank` spécifie une nouvelle fenêtre, `_parent` spécifie le parent de l'image actuelle et `_top` spécifie l'image de plus haut niveau dans la fenêtre active. Si la propriété `TextFormat.url` est une chaîne vide ou `null`, vous pouvez obtenir ou définir cette propriété, mais la propriété n'aura aucun effet.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

## Exemple

L'exemple suivant crée un champ texte incluant un hyperlien pointant vers le site Web de Macromedia. Cet exemple utilise `TextFormat.target` pour afficher le site Web de Macromedia dans une nouvelle fenêtre de navigateur.

```
var myformat:TextFormat = new TextFormat();
myformat.url = "http://www.macromedia.com";
myformat.target = "_blank";

this.createTextField("mytext",1,100,100,200,100);
mytext.multiline = true;
mytext.wordWrap = true;
mytext.border = true;
mytext.html = true;
mytext.text = "Go to Macromedia.com";
mytext.setTextFormat(myformat);
```

## Voir également

`url (propriété TextFormat.url)`

# TextFormat, constructeur

```
public TextFormat([font:String], [size:Number], [color:Number],
  [bold:Boolean], [italic:Boolean], [underline:Boolean], [url:String],
  [target:String], [align:String], [leftMargin:Number],
  [rightMargin:Number], [indent:Number], [leading:Number])
```

Crée un objet TextFormat avec les propriétés spécifiées. Vous pouvez modifier les propriétés de l'objet TextFormat pour modifier le format des champs de texte.

Tout paramètre peut être défini sur `null` pour indiquer qu'il n'est pas défini. Tous les paramètres sont facultatifs ; tous les paramètres omis sont traités comme `null`.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

## Paramètres

*font*:`String` [facultatif] - Nom d'une police de texte sous forme de chaîne.

*size*:`Number` [facultatif] - Entier indiquant la taille en point.

*color*:`Number` [facultatif] - Couleur du texte qui utilise ce format de texte. Nombre contenant trois composants RVB 8 bits ; par exemple, 0xFF0000 correspond au rouge et 0x00FF00 au vert.

*bold*:`Boolean` [facultatif] - Valeur booléenne qui indique si le texte est en gras.

*italic*:`Boolean` [facultatif] - Valeur booléenne qui indique si le texte est en italique.

*underline*:`Boolean` [facultatif] - Valeur booléenne qui indique si le texte est souligné.

*url*:`String` [facultatif] - URL correspondant à l'hyperlien du texte de ce format de texte. Si l'`url` est une chaîne vide, le texte ne comporte pas d'hyperlien.

*target*:`String` [facultatif] - Fenêtre cible dans laquelle s'affiche l'hyperlien. Si la fenêtre cible est une chaîne vide, le texte s'affiche dans la fenêtre cible par défaut `_self`. Si le paramètre `url` est défini sur une chaîne vide ou sur la valeur `null`, vous pouvez obtenir ou définir cette propriété, mais la propriété n'aura aucun effet.

*align*:`String` [facultatif] - Alignement du paragraphe, représenté sous forme de chaîne. Si `"left"`, le paragraphe est aligné à gauche. Si `"center"`, le paragraphe est centré. Si `"right"`, le paragraphe est aligné à droite.

*leftMargin*:`Number` [facultatif] - Indique la marge gauche du paragraphe, en points.

*rightMargin*:`Number` [facultatif] - Indique la marge droite du paragraphe, en points.

*indent*:`Number` [facultatif] - Entier indiquant l'indentation à appliquer de la marge gauche au premier caractère du paragraphe.

*leading*:`Number` [facultatif] - Nombre qui indique le montant d'interlignage vertical entre les lignes.

### Exemple

L'exemple suivant crée un objet TextFormat, formate le champ texte `stats_txt` et crée un nouveau champ pour afficher le texte :

```
// Define a TextFormat which is used to format the stats_txt text field.
var my_fmt:TextFormat = new TextFormat();
my_fmt.bold = true;
my_fmt.font = "Arial";
my_fmt.size = 12;
my_fmt.color = 0xFF0000;
// Create a text field to display the player's statistics.
this.createTextField("stats_txt", 5000, 10, 0, 530, 22);
// Apply the TextFormat to the text field.
stats_txt.setNewTextFormat(my_fmt);
stats_txt.selectable = false;
stats_txt.text = "Lorem ipsum dolor sit amet...";
```

Pour afficher un autre exemple, consultez le fichier animations.fla file dans le dossier d'exemples ActionScript. La liste suivante fournit des chemins types vers le dossier d'exemples ActionScript :

- Windows : *lecteur d'amorçage*\Program Files\Macromedia\Flash 8\Samples and Tutorials\Samples\ActionScript
- Macintosh : *disque dur Macintosh*/Applications/Macromedia Flash 8/Samples and Tutorials/Samples/ActionScript

## underline (propriété TextFormat.underline)

```
public underline : Boolean
```

Valeur booléenne qui indique si le texte qui utilise ce format texte est souligné (`true`) ou non (`false`). Ce soulignement est similaire à celui créé par la balise `<U>` mais ce dernier n'est pas un vrai soulignement, étant donné qu'il ne saute pas correctement les lettres à queue. La valeur par défaut est `null`, ce qui indique que la propriété n'est pas définie.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

### Exemple

L'exemple suivant crée un champ texte et souligne le texte.

```
this.createTextField("mytext",1,100,100,200,100);
mytext.multiline = true;
mytext.wordWrap = true;
mytext.border = true;

var myformat:TextFormat = new TextFormat();
myformat.underline = true;
mytext.text = "This is my first text field object text";
```

```
mytext.setTextFormat(myformat);
```

## url (propriété TextFormat.url)

```
public url : String
```

Indique l'URL à laquelle relie le texte de ce format texte. Si la propriété `url` est une chaîne vide, le texte ne comporte pas d'hyperlien. La valeur par défaut est `null`, ce qui indique que la propriété n'est pas définie.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

### Exemple

Cet exemple crée un champ texte constituant un hyperlien qui pointe vers le site Web de Macromedia.

```
var myformat:TextFormat = new TextFormat();
myformat.url = "http://www.macromedia.com";

this.createTextField("mytext",1,100,100,200,100);
mytext.multiline = true;
mytext.wordWrap = true;
mytext.border = true;
mytext.html = true;
mytext.text = "Go to Macromedia.com";
mytext.setTextFormat(myformat);
```

# TextRenderer (flash.text.TextRenderer)

```
Object
  |
  +-flash.text.TextRenderer
```

```
public class TextRenderer
extends Object
```

La classe TextRenderer permet d'exploiter la fonction avancée d'anti-aliasing des polices incorporées. L'anti-aliasing avancé permet d'obtenir un rendu très précis des polices de petite taille. Utilisez l'anti-aliasing avancé avec les applications comportant beaucoup de texte de petite taille. Macromedia ne recommande pas l'utilisation de l'anti-aliasing avancé pour les polices de très grande taille (supérieures à 48 points). L'anti-aliasing avancé est uniquement disponible sous Flash Player 8.

Pour définir l'anti-aliasing avancé sur un champ texte, définissez la propriété `antiAliasType` de l'occurrence TextField. Les exemples suivants nécessitent une police partagée dans la bibliothèque avec un identifiant de liaison appelé `"CustomFont"`.

```
var txtFormat:TextFormat = new TextFormat();
txtFormat.font = "CustomFont";

var label:TextField = this.createTextField("label",
  this.getNextHighestDepth(), 10, 10, 200, 20);
label.setNewTextFormat(txtFormat);
label.text = "Hello World";
label.embedFonts = true;
label.antiAliasType = "advanced";
```

L'anti-aliasing avancé offre une modulation continue du trait (CSM) qui s'applique à la fois à l'épaisseur du trait et à la netteté des bords. En tant que fonctionnalité avancée, vous pouvez utiliser la méthode `setAdvancedAntialiasingTable()` pour définir des paramètres pour des tailles de caractères et de polices spécifiques.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

## Voir également

`antiAliasType (propriété TextField.antiAliasType)`

## Résumé des propriétés

| Modificateurs | Propriété | Description |
|---|---|---|
| `static` | `maxLevel:Number` | Le niveau de qualité des champs de distance échantillonnés de façon adaptative (ADF) à appliquer pour l'anti-aliasing avancé. |

*Propriétés héritées de la classe Object*

`constructeur (propriété Object.constructor), __proto__ (Object.__proto__ property), prototype (Object.prototype, propriété), __resolve (Object.__resolve, propriété)`

### Résumé de la méthode

| Modificateurs | Signature | Description |
|---|---|---|
| static | setAdvancedAntialiasingTable(fontName:String, fontStyle:String, colorType:String, advancedAntialiasingTable:Array) : Void | Définit un tableau de recherche personnalisé de la modulation continue du trait (CSM) pour une police. |

*Méthodes héritées de la classe Object*

addProperty (méthode Object.addProperty), hasOwnProperty (méthode Object.hasOwnProperty), isPropertyEnumerable (méthode Object.isPropertyEnumerable), isPrototypeOf (méthode Object.isPrototypeOf), registerClass (méthode Object.registerClass), toString (méthode Object.toString), unwatch (méthode Object.unwatch), valueOf (méthode Object.valueOf), watch (méthode Object.watch)

## maxLevel (propriété TextRenderer.maxLevel)

```
public static maxLevel : Number
```

Le niveau de qualité des champs de distance échantillonnés de façon adaptative (ADF) à appliquer pour l'anti-aliasing avancé. Les seules valeurs acceptables sont 3, 4 et 7.

L'anti-aliasing avancé a recours aux ADF pour représenter les contours qui déterminent un glyphe. Plus la qualité est élevée, plus les structures ADF consomment de la mémoire cache. Une valeur de `3` occupe le moins d'espace mémoire possible, mais débouche sur la qualité moindre. Les polices de plus grande taille nécessitent davantage d'espace en mémoire cache ; pour les polices dont la taille est de 64 pixels, le niveau de qualité passe de `3` à `4` ou de `4` à `7`, à moins que ce niveau ne soit déjà défini sur `7`.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

### Exemple

L'exemple suivant spécifie la valeur `maxLevel` de l'ensemble du fichier SWF et affiche ensuite un champ texte avec la valeur définie. Pour que le texte de cet exemple soit affiché correctement, un symbole de police doit être disponible avec un identifiant de liaison de `"CustomFont"`.

```
import flash.text.TextRenderer;
TextRenderer.maxLevel = 3;
```

```
var txtFormat:TextFormat = new TextFormat();
txtFormat.font = "CustomFont";
txtFormat.size = 64;

var label:TextField = this.createTextField("label",
  this.getNextHighestDepth(), 10, 10, 500, 100);
label.setNewTextFormat(txtFormat);
label.text = "Hello World";
label.embedFonts = true;
trace("TextRenderer.maxLevel: " + TextRenderer.maxLevel);
```

# setAdvancedAntialiasingTable (méthode TextRenderer.setAdvancedAntialiasingTable)

```
public static setAdvancedAntialiasingTable(fontName:String,
  fontStyle:String, colorType:String, advancedAntialiasingTable:Array) :
  Void
```

Définit un tableau de recherche personnalisé de la modulation continue du trait (CSM) pour une police. Il s'agit d'une méthode avancée.

Flash Player inclut uniquement les paramètres d'anti-aliasing avancés pour 10 polices de base et pour ces polices, les paramètres ne s'appliquent qu'aux tailles 6 à 20. Toutes les tailles inférieures à 6 reprennent les paramètres de la taille 6. De même, toutes les tailles supérieures à 20 reprennent les paramètres de la taille 20. Les autres polices correspondent aux données de police fournies. La méthode `setAdvancedAntialiasingTable()` permet de définir des données d'anti-aliasing personnalisées pour les autres polices et tailles de police, ou de remplacer les paramètres par défaut des polices fournies.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

## Paramètres

*`fontName`*`:String` - Nom de la police pour laquelle vous appliquez des paramètres.

*`fontStyle`*`:String` - Le style de police peut être `"bold"`, `"bolditalic"`, `"italic"` et `"none"`.

*`colorType`*`:String` - Cette valeur peut être `"dark"` ou `"light"`.

*`advancedAntialiasingTable`*`:Array` - Tableau de paramètres CSM pour la police spécifiée. Chaque paramètre est un objet disposant des propriétés suivantes :

- `fontSize`
- `insideCutOff`
- `outsideCutOff`

Le tableau `advancedAntialiasingTable` peut comporter plusieurs entrées qui spécifient les paramètres CSM pour différentes tailles de police. (Voir l'exemple)

La `fontSize` est la taille, en pixels, pour laquelle les paramètres s'appliquent.

L'anti-aliasing avancé a recours aux champs de distance échantillonnés de façon adaptative (ADF) pour représenter les contours qui déterminent un glyphe. Flash applique une valeur butoir externe (`outsideCutOff`), en-deçà de laquelle les densités sont définies sur zéro, et une valeur butoir interne (`insideCutOff`), au-delà de laquelle les densités sont limitées à une valeur donnée (telle que 255). Entre ces deux valeurs, la fonction de mappage est une courbe linéaire allant de zéro, pour la valeur butoir externe, jusqu'à la valeur de densité maximum définie pour la valeur butoir interne.

Le réglage des valeurs butoir externe et interne affecte l'épaisseur du trait et la netteté des bords. L'espacement séparant ces deux paramètres est comparable au double du rayon du filtre des méthodes classiques d'anti-aliasing ; un espacement étroit fournit un bord plus net, tandis qu'un espacement plus large donne un bord plus doux, plus filtré. Lorsque l'espacement est nul, l'image de densité résultante est un bitmap à deux niveaux. Lorsque l'espacement est très large, le bord de l'image de densité résultante est assez semblable à celui d'une aquarelle.

De manière générale, les utilisateurs préfèrent les bords nets à fort contraste pour les petites tailles et des bords plus doux pour le texte animé et les polices de grande taille.

La valeur butoir externe correspond généralement à une valeur négative, tandis que la valeur interne a une valeur positive. Leur point intermédiaire est proche de zéro. Le réglage de ces paramètres pour décaler le point intermédiaire vers l'infini négatif augmente l'épaisseur du trait, tandis que son décalage vers l'infini positif la réduit. Assurez-vous que la valeur butoir externe soit toujours inférieure ou égale à la valeur interne.

Dans la plupart des circonstances, un exposant gamma égal à 1 est adéquat. Cependant, lors du rendu de sous-pixels [mode écran à cristaux liquides(LCD)], l'exposant gamma est utilisé pour atténuer les effets de franges de couleur observés lors du rendu des polices à trait fin (par exemple, Times Roman) et de petites tailles en point. Vous pouvez également utiliser l'exposant gamma pour améliorer un contraste dans les deux modes tube à rayon cathodique (CRT) et LCD.

## Exemple

L'exemple suivant crée deux entrées anti-alias et deux champs texte pour les illustrer. Pour que cet exemple fonctionne, le fichier SWF doit avoir une police partagée intégrée avec un identifiant de liaison de `"myArial"`. Pour intégrer la police, procédez comme suit :

- Ouvrez votre Bibliothèque.
- Cliquez sur le menu d'options de la Bibliothèque dans le coin supérieur droit de la Bibliothèque.

- Sélectionnez Nouvelle Police dans le menu contextuel.
- Nommez la police *myArial*.
- Sélectionnez Arial dans le menu contextuel de police.
- Cliquez sur OK.
- Cliquez du bouton droit sur la nouvelle police créée et sélectionnez Liaison.
- Activez la case à cocher Exporter pour ActionScript.
- Cliquez sur OK pour accepter l'identifiant par défaut, myArial.

```
import flash.text.TextRenderer;

var antiAliasEntry_1 = {fontSize:24, insideCutoff:1.61, outsideCutoff:-
  3.43};
var antiAliasEntry_2 = {fontSize:48, insideCutoff:0.8, outsideCutoff:-0.8};
var arialTable:Array = new Array(antiAliasEntry_1, antiAliasEntry_2);

var lbl_1:TextField = createLabel(0, 0, 300, 100, 24);
var lbl_2:TextField = createLabel(0, 100, 300, 100, 48);

TextRenderer.setAdvancedAntialiasingTable("Arial", "none", "dark",
  arialTable);

function createLabel(x:Number, y:Number, width:Number, height:Number,
  fontSize:Number):TextField {
  var depth:Number = this.getNextHighestDepth();

  var tmpTxt = this.createTextField("txt_" + depth, depth, x, y, width,
  height);
  tmpTxt.antiAliasType = "advanced";
  tmpTxt.gridFitType = "pixel";
  tmpTxt.border = true;
  tmpTxt.text = "Hello World";
  tmpTxt.embedFonts = true;
  tmpTxt.setTextFormat(getTextFormat(fontSize));
  return tmpTxt;
}

function getTextFormat(fontSize:Number):TextFormat {
  var tf:TextFormat = new TextFormat();
  tf.align = "center";
  tf.size = fontSize;
  tf.font = "myArial";
  return tf;
}
```

# TextSnapshot

```
Object
  |
  +-TextSnapshot
```

public class *TextSnapshot*
extends Object

Les objets TextSnapshot permettent de travailler avec du texte statique dans un clip. Vous pouvez les utiliser, par exemple, pour mettre en forme du texte avec une plus grande précision que celle permise par un texte dynamique, mais le texte est toujours en lecture seule.

Vous n'utilisez pas de constructeur pour créer un objet TextSnapshot ; il est renvoyé par la méthode `MovieClip.getTextSnapshot()`.

**Disponibilité :** ActionScript 1.0 ; Flash Player 7

## Voir également

`getTextSnapshot (méthode MovieClip.getTextSnapshot)`

## Résumé des propriétés

*Propriétés héritées de la classe Object*

`constructeur (propriété Object.constructor), __proto__ (Object.__proto__ property), prototype (Object.prototype, propriété), __resolve (Object.__resolve, propriété)`

## Résumé de la méthode

| Modificateurs | Signature | Description |
|---|---|---|
| | `findText(startIndex:Number, textToFind:String, caseSensitive:Boolean) : Number` | Recherche l'objet TextSnapshot spécifié et renvoie la position de la première occurrence de `textToFind` figurant au niveau de ou après `startIndex`. |
| | `getCount() : Number` | Renvoie le nombre de caractères dans un objet TextSnapshot. |
| | `getSelected(start:Number, [end:Number]) : Boolean` | Renvoie une valeur booléenne qui spécifie si un objet TextSnapshot contient du texte sélectionné dans la plage spécifiée. |

| Modificateurs | Signature | Description |
|---|---|---|
| | `getSelectedText([includeLineEndings:Boolean]) : String` | Renvoie une chaîne qui contient tous les caractères spécifiés par la méthode `TextSnapshot.setSelected()` correspondante. |
| | `getText(start:Number, end:Number, [includeLineEndings:Boolean]) : String` | Renvoie une chaîne qui contient tous les caractères spécifiés par les paramètres `start` et `end`. |
| | `getTextRunInfo(beginIndex:Number, endIndex:Number) : Array` | Renvoie un tableau d'objets contenant des informations sur un segment de texte. |
| | `hitTestTextNearPos(x:Number, y:Number, [closeDist:Number]) : Number` | Permet de déterminer lequel des caractères contenus dans un objet TextSnapshot se trouve sur ou est proche des coordonnées spécifiées *x*, *y* du clip contenant le texte dans l'objet TextSnapshot. |
| | `setSelectColor(color:Number) : Void` | Spécifie la couleur à utiliser lors de la mise en surbrillance des caractères qui ont été sélectionnés avec la méthode `TextSnapshot.setSelected()`. |
| | `setSelected(start:Number, end:Number, select:Boolean) : Void` | Spécifie une plage de caractères dans un objet TextSnapshot à sélectionner ou désélectionner. |

*Méthodes héritées de la classe Object*

`addProperty (méthode Object.addProperty)`, `hasOwnProperty (méthode Object.hasOwnProperty)`, `isPropertyEnumerable (méthode Object.isPropertyEnumerable)`, `isPrototypeOf (méthode Object.isPrototypeOf)`, `registerClass (méthode Object.registerClass)`, `toString (méthode Object.toString)`, `unwatch (méthode Object.unwatch)`, `valueOf (méthode Object.valueOf)`, `watch (méthode Object.watch)`

# findText (méthode TextSnapshot.findText)

```
public findText(startIndex:Number, textToFind:String,
  caseSensitive:Boolean) : Number
```

Recherche l'objet TextSnapshot spécifié et renvoie la position de la première occurrence de `textToFind` figurant au niveau de ou après `startIndex`. Si `textToFind` n'est pas détecté, la méthode renvoie `-1`.

**Disponibilité :** ActionScript 1.0 ; Flash Player 7

### Paramètres

*startIndex*:Number - Spécifie le point d'index de départ dans le texte TextSnapshot où le texte spécifié doit être recherché.

*textToFind*:String - Texte à rechercher. Spécifiez soit un littéral de chaîne (mis entre guillemets), soit une variable.

*caseSensitive*:Boolean - Valeur booléenne spécifiant si le texte détecté doit correspondre à la casse de la chaîne dans textToFind (true); sinon, false.

### Valeur renvoyée

Number - Position d'index basée sur zéro de la première occurrence du texte spécifié, ou -1 si aucun texte correspondant n'a été trouvé.

### Exemple

L'exemple suivant illustre comment utiliser cette méthode. Pour utiliser ce code, placez un champ de texte statique contenant le texte « TextSnapshot Example » sur la Scène.

```
var my_mc:MovieClip = this;
var my_snap:TextSnapshot = my_mc.getTextSnapshot();
var index1:Number = my_snap.findText(0, "Snap", true);
var index2:Number = my_snap.findText(0, "snap", true);
var index3:Number = my_snap.findText(0, "snap", false);
trace(index1); // 4
trace(index2); // -1
trace(index3); // 4
```

### Voir également

getText (méthode TextSnapshot.getText)

## getCount (méthode TextSnapshot.getCount)

```
public getCount() : Number
```

Renvoie le nombre de caractères dans un objet TextSnapshot.

**Disponibilité :** ActionScript 1.0 ; Flash Player 7

### Valeur renvoyée

Number - Renvoie le nombre de caractères dans l'objet TextSnapshot.

### Exemple

L'exemple suivant illustre comment renvoyer le nombre de caractères dans un objet TextSnapshot. Pour utiliser ce code, placez un champ de texte statique contenant le texte « TextSnapshot Example » (et uniquement ce texte) sur la Scène.

```
var my_mc:MovieClip = this;
var my_snap:TextSnapshot = my_mc.getTextSnapshot();
var count:Number = my_snap.getCount();
var theText:String = my_snap.getText(0, count, false);
trace(count); // 20
trace(theText); // TextSnapshot Example
```

### Voir également

`getText (méthode TextSnapshot.getText)`

## getSelected (méthode TextSnapshot.getSelected)

```
public getSelected(start:Number, [end:Number]) : Boolean
```

Renvoie une valeur booléenne qui spécifie si un objet TextSnapshot contient du texte sélectionné dans la plage spécifiée.

Pour rechercher tous les caractères, transmettez une valeur `0` pour `start` et `TextSnapshot.getCount()` (ou un grand nombre quelconque) pour `end`. Pour rechercher un seul caractère, transmettez au paramètre `end` une valeur égale au paramètre `start` plus un.

**Disponibilité :** ActionScript 1.0 ; Flash Player 7

### Paramètres

*`start`*`:Number` - Position d'index du premier caractère à examiner. Les valeurs valides pour `start` vont de `0` à `TextSnapshot.getCount() - 1`. Si `start` est une valeur négative, `0` est utilisé.

*`end`*`:Number` [facultatif] - Position d'index qui correspond à 1+ le dernier caractère à examiner. Les valeurs valides pour `end` vont de `0` à `TextSnapshot.getCount()`. Le caractère indexé par le paramètre `end` n'est pas inclus dans la chaîne extraite. Si vous omettez ce paramètre, `TextSnapshot.getCount()` est utilisé. Si la valeur de `end` est inférieure ou égale à la valeur `start`, `start + 1` est utilisé.

### Valeur renvoyée

`Boolean` - Valeur booléenne indiquant si, dans la plage donnée, au moins un caractère a été sélectionné par la méthode `TextSnapshot.setSelected()` correspondante (`true`); sinon, `false`.

### Exemple

L'exemple suivant illustre comment utiliser cette méthode. Pour utiliser ce code, placez un champ de texte statique contenant le texte « TextSnapshot Example » sur la Scène. Incluez dans la bibliothèque la police utilisée par le champ texte statique et dans options de Liaison pour la police, sélectionnez Exporter pour ActionScript. Ajoutez le code ActionScript suivant à l'image 1 du scénario :

```
var my_snap:TextSnapshot = this.getTextSnapshot();
var count:Number = my_snap.getCount();
my_snap.setSelected(0, 4, true);
my_snap.setSelected(1, 2, false);

var firstCharIsSelected:Boolean = my_snap.getSelected(0, 1);
var secondCharIsSelected:Boolean = my_snap.getSelected(1, 2);
trace(firstCharIsSelected); // true
trace(secondCharIsSelected); // false
```

### Voir également

`getCount (méthode TextSnapshot.getCount)`, `getText (méthode TextSnapshot.getText)`, `getSelectedText (méthode TextSnapshot.getSelectedText)`, `setSelected (méthode TextSnapshot.setSelected)`

## getSelectedText (méthode TextSnapshot.getSelectedText)

```
public getSelectedText([includeLineEndings:Boolean]) : String
```

Renvoie une chaîne qui contient tous les caractères spécifiés par la méthode `TextSnapshot.setSelected()` correspondante. Si aucun caractère n'est spécifié (par la méthode `TextSnapshot.setSelected()`), une chaîne vide est renvoyée.

Si vous transmettez `true` pour `includeLineEndings`, des caractères de nouvelle ligne sont insérés dans la chaîne de renvoi et la chaîne de renvoi peut être plus longue que la plage d'entrée. Si `includeLineEndings` est `false` ou omis, la méthode renvoie le texte sélectionné sans ajouter de caractères.

**Disponibilité :** ActionScript 1.0 ; Flash Player 7

### Paramètres

*`includeLineEndings`*`:Boolean` [facultatif] - Valeur booléenne qui spécifie si les caractères de nouvelle ligne sont insérés (`true`) ou s'ils ne sont pas insérés (`false`) dans la chaîne renvoyée. La valeur par défaut est `false`.

### Valeur renvoyée

`String` - Chaîne qui contient tous les caractères spécifiés par la méthode `TextSnapshot.setSelected()` correspondante.

### Exemple

L'exemple suivant illustre comment utiliser cette méthode. Pour utiliser ce code, placez un champ de texte statique contenant le texte « TextSnapshot Example » sur la Scène. Incluez ensuite dans la bibliothèque la police utilisée par le champ texte statique et dans options de Liaison pour la police, sélectionnez Exporter pour ActionScript. Ajoutez le code ActionScript suivant à l'image 1 du scénario :

```
var my_snap:TextSnapshot = this.getTextSnapshot();
var count:Number = my_snap.getCount();
my_snap.setSelected(0, 4, true);
my_snap.setSelected(1, 2, false);

var theText:String = my_snap.getSelectedText(false);
trace(theText); // Text
```

Lorsque vous testez le fichier SWF, un rectangle de couleur entoure les caractères spécifiés.

### Voir également

`getSelected (méthode TextSnapshot.getSelected)`, `setSelected (méthode TextSnapshot.setSelected)`

## getText (méthode TextSnapshot.getText)

```
public getText(start:Number, end:Number, [includeLineEndings:Boolean]) :
  String
```

Renvoie une chaîne qui contient tous les caractères spécifiés par les paramètres `start` et `end`. Si aucun caractère n'est spécifié, la méthode renvoie une chaîne vide.

Pour renvoyer tous les caractères, transmettez une valeur de `0` pour `start` et `TextSnapshot.getCount()` ou un grand nombre quelconque) pour `end`. Pour renvoyer un seul caractère, transmettez une valeur de `start +1` pour `end`.

Si vous transmettez `true` pour `includeLineEndings`, des caractères de nouvelle ligne sont insérés dans la chaîne de renvoi lorsque cela semble approprié et la chaîne de renvoi peut être plus longue que la plage d'entrée. Si `includeLineEndings` est `false` ou omis, la méthode renvoie le texte sélectionné sans ajouter de caractères.

**Disponibilité :** ActionScript 1.0 ; Flash Player 7

### Paramètres

*start*:Number - Entier qui indique la position du premier caractère à inclure dans la chaîne renvoyée. Les valeurs valides pour start vont de 0 à TextSnapshot.getCount() - 1. Si start est une valeur négative, 0 est utilisé.

*end*:Number - Entier qui correspond à 1+ l'index du dernier caractère à examiner dans l'objet TextSnapshot. Les valeurs valides pour end vont de 0 à TextSnapshot.getCount(). Le caractère indexé par le paramètre end n'est pas inclus dans la chaîne extraite. Si vous omettez ce paramètre, TextSnapshot.getCount() est utilisé. Si la valeur de end est inférieure ou égale à la valeur start, start + 1 est utilisé.

*includeLineEndings*:Boolean [facultatif] - Valeur booléenne qui spécifie si les caractères de nouvelle ligne sont insérés (true) ou s'ils ne sont pas insérés (false) dans la chaîne renvoyée. La valeur par défaut est false.

### Valeur renvoyée

String - Chaîne contenant les caractères dans la plage spécifiée, ou une chaîne vide si aucun caractère n'est détecté dans la plage spécifiée.

### Exemple

L'exemple suivant illustre comment renvoyer le nombre de caractères dans un objet TextSnapshot spécifié. Pour utiliser ce code, placez un champ de texte statique contenant le texte « TextSnapshot Example » sur la Scène.

```
var my_mc:MovieClip = this;
var my_snap:TextSnapshot = my_mc.getTextSnapshot();
var count:Number = my_snap.getCount();
var theText:String = my_snap.getText(0, count, false);
trace(count); // 20
trace(theText); // TextSnapshot Example
```

### Voir également

getCount (méthode TextSnapshot.getCount), getSelectedText (méthode TextSnapshot.getSelectedText)

## getTextRunInfo (méthode TextSnapshot.getTextRunInfo)

```
public getTextRunInfo(beginIndex:Number, endIndex:Number) : Array
```

Renvoie un tableau d'objets contenant des informations sur un segment de texte. Chaque objet correspond à un caractère dans la plage de caractères spécifiée par les deux paramètres de méthode.

*Remarque :* L'utilisation de la méthode `getTextRunInfo()` pour une plage de texte volumineuse peut générer le renvoi d'un objet volumineux. Macromedia recommande de limiter la plage de texte définie par les paramètres `beginIndex` et `endIndex`.

**Disponibilité :** ActionScript 1.0 ; Flash Player 7

## Paramètres

*`beginIndex`*`:Number` - Valeur d'index du premier caractère dans la plage de caractères.

*`endIndex`*`:Number` - Valeur d'index du dernier caractère dans la plage de caractères.

## Valeur renvoyée

`Array` - Tableau d'objets dans lequel chaque objet contient des informations sur un caractère spécifique dans la plage spécifique. Chaque objet contient les propriétés suivantes :

- `indexInRun` index d'entiers à base zéro du caractère (par rapport à l'ensemble de la chaîne plutôt qu'au segment de texte sélectionné).
- `selected` Valeur booléenne indiquant si le caractère est sélectionné comme `true`; `false` dans le cas contraire .
- `font` Le nom de la police de caractère.
- `color` La valeur combinée alpha et couleur du caractère. Les deux premiers nombres hexadécimaux représentent la valeur alpha, les autres nombres représentent la valeur couleur. (L'exemple comprend une méthode de conversion des valeurs décimales en valeurs hexadécimales.)
- `height` La hauteur des caractères, en pixels.
- `matrix_a`, `matrix_b`, `matrix_c`, `matrix_d`, `matrix_tx`, et `matrix_ty` Les valeurs d'une matrice qui définissent la transformation géométrique sur le caractère. La matrice d'un texte normal, droit est toujours de la forme `[1 0 0 1 x y]`, où `x` et `y` sont les positions du caractère dans le clip parent, indépendamment de la hauteur du texte. La matrice se trouve dans le système de coordonnées du clip parent et ne contient aucune transformations susceptibles de se trouver sur ce clip même (ou son parent).
- `corner0x`, `corner0y`, `corner1x`, `corner1y`, `corner2x`, `corner2y`, `corner3x`, et `corner3y` Les coins du cadre de délimitation du caractère, basé sur le système de coordonnées du clip parent. Ces valeurs ne sont disponibles que si la police utilisée par le caractère est intégrée dans le fichier SWF.

## Exemple

L'exemple suivant illustre comment utiliser cette méthode. Pour utiliser ce code, sur la Scène, créez un champ texte statique contenant le texte « AB ». Faites pivoter le champ texte de 45 degrés et définissez le deuxième caractère dans le style exposant avec une couleur de 0xFFFFFF, avec un alpha de 50%, comme dans l'illustration suivante :

Le script suivant répertorie les propriétés `getTextRunInfo()` de chaque caractère dans le champ texte :

```
var myTS:TextSnapshot = this.getTextSnapshot();
var myArray:Array = myTS["getTextRunInfo"](0, myTS.getCount());
for (var i = 0; i < myTS.getCount(); i++) {
  trace("indexInRun: " + myArray[i].indexInRun);
  trace("selected: " + myArray[i].selected);
  trace("font: " + myArray[i].font);
  trace("color: " + decToHex(myArray[i].color));
  trace("height: " + myArray[i].height);
  trace("matrix_a: " + myArray[i].matrix_a);
  trace("matrix_b: " + myArray[i].matrix_b);
  trace("matrix_c: " + myArray[i].matrix_c);
  trace("matrix_d: " + myArray[i].matrix_d);
  trace("matrix_ty: " + myArray[i].matrix_tx);
  trace("matrix_tx: " + myArray[i].matrix_ty);
  trace(" ");
}

function decToHex(dec:Number) {
  var hexString:String = "";
  if (dec > 15) {
    hexString = decToHex(Math.floor(dec / 16));
  }
  var hexDigit = dec - 16 * (Math.floor(dec / 16));
    if (hexDigit > 9) {
      hexDigit = String.fromCharCode(hexDigit + 55);
    }
    hexString = hexString + hexDigit;
    return hexString;
}
```

Ceci crée la sortie suivante :

```
indexInRun: 0
selected: false
font: Times New Roman
color: FF000000
height: 28.6
matrix_a: 0.0316612236983293
matrix_b: 0.0385940558426864
```

```
matrix_c: -0.0385940558426864
matrix_d: 0.0316612236983293
matrix_ty: 22.75
matrix_tx: 40.35

indexInRun: 0
selected: false
font: Times New Roman
color: 80000000
height: 28.6
matrix_a: 0.0316612236983293
matrix_b: 0.0385940558426864
matrix_c: -0.0385940558426864
matrix_d: 0.0316612236983293
matrix_ty: 49
matrix_tx: 45.5
```

Cet exemple utilise une méthode `decToHex()` pour convertir la valeur décimale de la propriété `color` en une valeur hexadécimale.

### Voir également

`Matrix (flash.geom.Matrix)`

## hitTestTextNearPos (méthode TextSnapshot.hitTestTextNearPos)

```
public hitTestTextNearPos(x:Number, y:Number, [closeDist:Number]) : Number
```

Permet de déterminer lequel des caractères contenus dans un objet TextSnapshot se trouve sur ou est proche des coordonnées spécifiées *x*, *y* du clip contenant le texte dans l'objet TextSnapshot.

Si vous omettez ou transmettez une valeur `0` pour `closeDist`, l'emplacement spécifié par les coordonnées *x*, *y* doit se trouver à l'intérieur du cadre de délimitation de l'objet TextSnapshot.

Cette méthode fonctionne correctement uniquement pour les polices qui incluent des informations métriques de caractères ; par défaut cependant, Macromedia Flash n'inclut pas ces informations pour les champs de texte statique. Par conséquent, il se peut que la méthode renvoie `-1` au lieu d'une valeur d'indexation. Pour garantir qu'une valeur d'indexation est renvoyée, vous pouvez forcer l'outil de programmation Flash à inclure les informations métriques de caractères pour une police. Pour ce faire, ajoutez un champ de texte dynamique qui utilise cette police, sélectionnez les options des caractères pour ce champ de texte dynamique et spécifiez que les polices vectorielles doivent être intégrées pour au moins un caractère. (Le choix des caractères spécifiés et la présence ou non des caractères utilisés dans les champs de texte statique en question importent peu.)

**Disponibilité :** ActionScript 1.0 ; Flash Player 7

## Paramètres

*x*:Number - Coordonnée *x* du clip qui contient le texte dans l'objet TextSnapshot.

*y*:Number - Coordonnée *y* du clip qui contient le texte dans l'objet TextSnapshot.

*closeDist*:Number [facultatif] - Distance maximale de *x*, *y* dans laquelle une recherche de texte peut être lancée. Cette distance est mesurée à partir du point central de chaque caractère. La valeur par défaut est 0.

## Valeur renvoyée

Number - Valeur d'index du caractère dans l'objet TextSnapshot le plus proche de la coordonnée *x*, *y* spécifiée. La méthode renvoie -1 si aucun caractère n'est détecté ou si la police ne contient pas d'informations métriques de caractères.

## Exemple

L'exemple suivant illustre comment utiliser cette méthode. Pour utiliser ce code, placez un champ de texte statique contenant le texte « TextSnapshot Example » sur la Scène. Incluez dans la bibliothèque la police utilisée par le champ texte statique et dans options de Liaison pour la police, sélectionnez Exporter pour ActionScript. Pour tester le code, exécutez le fichier SWF et placez le pointeur de la souris sur le texte à l'écran.

```
var my_ts:TextSnapshot = getTextSnapshot();
this.onMouseMove = function() {
  var hitIndex:Number = my_ts.hitTestTextNearPos(_xmouse, _ymouse, 0);
  my_ts.setSelected(0, my_ts.getCount(), false);
  if (hitIndex >= 0) {
    my_ts.setSelected(hitIndex, hitIndex + 1, true);
  }
};
```

## Voir également

getTextSnapshot (méthode MovieClip.getTextSnapshot), _x (propriété MovieClip._x), _y (propriété MovieClip._y)

# setSelectColor (méthode TextSnapshot.setSelectColor)

```
public setSelectColor(color:Number) : Void
```

Spécifie la couleur à utiliser lors de la mise en surbrillance des caractères qui ont été sélectionnés avec la méthode `TextSnapshot.setSelected()`. La couleur est toujours opaque ; vous ne pouvez pas spécifier de valeur de transparence.

Cette méthode fonctionne correctement uniquement pour les polices qui incluent des informations métriques de caractères ; par défaut cependant, Macromedia Flash n'inclut pas ces informations pour les champs de texte statique. Par conséquent, il se peut que la méthode renvoie `-1` au lieu d'une valeur d'indexation. Pour garantir qu'une valeur d'indexation est renvoyée, vous pouvez forcer l'outil de programmation Flash à inclure les informations métriques de caractères pour une police. Pour ce faire, ajoutez un champ de texte dynamique qui utilise cette police, sélectionnez les options des caractères pour ce champ de texte dynamique et spécifiez que les polices vectorielles doivent être intégrées pour au moins un caractère. (Le choix des caractères spécifiés et la présence ou non des caractères utilisés dans les champs de texte statique en question importent peu.)

**Disponibilité :** ActionScript 1.0 ; Flash Player 7

## Paramètres

*`color`*`:Number` - Couleur du cadre entourant les caractères sélectionnés par la méthode `TextSnapshot.setSelected()` exprimée au format 0x*RRGGBB*.

## Exemple

L'exemple suivant illustre comment utiliser cette méthode. Pour utiliser ce code, placez un champ de texte statique contenant le texte « TextSnapshot Example » sur la Scène. Incluez dans la bibliothèque la police utilisée par le champ texte statique et dans options de Liaison pour la police, sélectionnez Exporter pour ActionScript. Ajoutez le code ActionScript suivant à l'image 1 du scénario :

```
var my_snap:TextSnapshot = this.getTextSnapshot();
var count:Number = my_snap.getCount();
my_snap.setSelectColor(0xFF0000);
my_snap.setSelected(0, 4, true);
my_snap.setSelected(1, 2, false);

var theText:String = my_snap.getSelectedText(false); // get the selected
  text
trace(theText); // Text
```

Lorsque vous testez le fichier SWF, un rectangle de couleur entoure les caractères spécifiés.

### Voir également

`setSelected (méthode TextSnapshot.setSelected)`

## setSelected (méthode TextSnapshot.setSelected)

```
public setSelected(start:Number, end:Number, select:Boolean) : Void
```

Spécifie une plage de caractères dans un objet TextSnapshot à sélectionner ou désélectionner. Les caractères qui sont sélectionnés sont dessinés sur fond de rectangle coloré, correspondant au cadre de délimitation du caractère. La couleur du cadre de délimitation est définie par `TextSnapshot.setSelectColor()`.

Pour sélectionner ou désélectionner tous les caractères, transmettez une valeur de `0` pour `start` et `TextSnapshot.getCount()` (ou un grand nombre quelconque) pour `end`. Pour spécifier un seul caractère, transmettez une valeur de `start + 1` pour `end`.

Les caractères étant individuellement marqués comme sélectionnés, vous pouvez appeler cette méthode plusieurs fois pour sélectionner plusieurs caractères ; c'est-à-dire, l'utilisation de cette méthode ne désélectionne pas les autres caractères qui ont été définis par cette méthode.

Cette méthode fonctionne correctement uniquement avec les polices qui incluent des informations métriques de caractères ; par défaut, Flash n'inclut pas ces informations pour les champs de texte statique. Par conséquent, il se peut que du texte sélectionné n'apparaisse pas pour être sélectionné à l'écran. Pour garantir que l'ensemble du texte sélectionné s'affiche comme tel, vous pouvez forcer l'outil de programmation Flash à inclure les informations métriques de caractères pour une police. A cet effet, incluez dans la bibliothèque la police utilisée par le champ texte statique et dans options de Liaison pour la police, sélectionnez Exporter pour ActionScript.

**Disponibilité :** ActionScript 1.0 ; Flash Player 7

### Paramètres

*`start`*`:Number` - Position du premier caractère à sélectionner. Les valeurs valides pour `start` vont de `0` à `TextSnapshot.getCount() - 1`. Si `start` est une valeur négative, `0` est utilisé.

*`end`*`:Number` - Entier qui correspond à 1+ l'index du dernier caractère à examiner. Les valeurs valides pour `end` vont de `0` à `TextSnapshot.getCount()`. Le caractère indexé par le paramètre `end` n'est pas inclus dans la chaîne extraite. Si vous omettez ce paramètre, `TextSnapshot.getCount()` est utilisé. Si la valeur de `end` est inférieure ou égale à la valeur `start`, `start + 1` est utilisé.

*`select`*`:Boolean` - Valeur booléenne indiquant si le texte doit être sélectionné (`true`) ou non (`false`).

### Exemple

L'exemple suivant illustre comment utiliser cette méthode. Pour utiliser ce code, placez un champ de texte statique contenant le texte « TextSnapshot Example » sur la Scène. Incluez dans la bibliothèque la police utilisée par le champ texte statique et dans options de Liaison pour la police, sélectionnez Exporter pour ActionScript. Ajoutez le code ActionScript suivant à l'image 1 du scénario :

```
var my_snap:TextSnapshot = this.getTextSnapshot();
var count:Number = my_snap.getCount();
my_snap.setSelected(0, 4, true);
my_snap.setSelected(1, 2, false);

var theText:String = my_snap.getSelectedText(false);
trace(theText); // Text
```

### Voir également

`getCount (méthode TextSnapshot.getCount)`

# Transform (flash.geom.Transform)

```
Object
  |
  +-flash.geom.Transform
```

`public class` ***Transform***
`extends Object`

La classe Transform rassemble des données sur les transformations de couleurs et les manipulations de coordonnées qui s'appliquent à un objet MovieClip.

Un objet Transform s'obtient normalement en lisant la valeur de la propriété `transform` dans un objet MovieClip.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

### Voir également

`transform (propriété MovieClip.transform)`, `ColorTransform (flash.geom.ColorTransform)`, `Matrix (flash.geom.Matrix)`

## Résumé des propriétés

| Modificateurs | Propriété | Description |
|---|---|---|
| | `colorTransform:ColorTransform` | Un objet ColorTransform contenant des valeurs qui règlent de façon universelle les couleurs dans le clip. |
| | `concatenatedColorTransform:ColorTransform` [lecture seule] | Objet ColorTransform représentant les transformations de couleur combinées qui s'appliquent à cet objet et à l'ensemble de ses objets parents, jusqu'à la racine. |
| | `concatenatedMatrix:Matrix` [lecture seule] | Objet Matrix représentant les matrices de transformation combinées qui s'appliquent à cet objet et à l'ensemble de ses objets parents, jusqu'à la racine. |
| | `matrix:Matrix` | Objet Matrix de transformation contenant des valeurs qui influent sur la mise à l'échelle, la rotation et la translation du clip. |
| | `pixelBounds:Rectangle` | Objet rectangle qui définit le cadre de délimitation de l'objet MovieClip sur la Scène. |

*Propriétés héritées de la classe Object*

`constructeur (propriété Object.constructor), __proto__ (Object.__proto__ property), prototype (Object.prototype, propriété), __resolve (Object.__resolve, propriété)`

## Résumé des constructeurs

| Signature | Description |
|---|---|
| `Transform(mc:MovieClip)` | Crée un nouvel objet Transform attaché à l'objet MovieClip donné. |

## Résumé de la méthode

*Méthodes héritées de la classe Object*

`addProperty (méthode Object.addProperty), hasOwnProperty (méthode Object.hasOwnProperty), isPropertyEnumerable (méthode Object.isPropertyEnumerable), isPrototypeOf (méthode Object.isPrototypeOf), registerClass (méthode Object.registerClass), toString (méthode Object.toString), unwatch (méthode Object.unwatch), valueOf (méthode Object.valueOf), watch (méthode Object.watch)`

# colorTransform (propriété Transform.colorTransform)

```
public colorTransform : ColorTransform
```

Objet ColorTransform contenant des valeurs qui règlent de façon universelle les couleurs dans le clip.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

## Exemple

L'exemple suivant applique l'objet ColorTransform `blueColorTransform` à l'objet Transform `trans`. Ce ColorTransform convertit la couleur du MovieClip `rect` de rouge en bleu.

```
import flash.geom.Transform;
import flash.geom.ColorTransform;

var rect:MovieClip = createRectangle(20, 80, 0xFF0000);

var trans:Transform = new Transform(rect);
trace(trans.colorTransform);
// (redMultiplier=1, greenMultiplier=1, blueMultiplier=1,
  alphaMultiplier=1, redOffset=0, greenOffset=0, blueOffset=0,
  alphaOffset=0)

var blueColorTransform:ColorTransform = new ColorTransform(0, 1, 1, 1, 0,
  0, 255, 0);

rect.onPress = function() {
  trans.colorTransform = blueColorTransform;
  trace(trans.colorTransform);
  // (redMultiplier=0, greenMultiplier=1, blueMultiplier=1,
  alphaMultiplier=1, redOffset=0, greenOffset=0, blueOffset=255,
  alphaOffset=0)
}

function createRectangle(width:Number, height:Number, color:Number,
  scope:MovieClip):MovieClip {
  scope = (scope == undefined) ? this : scope;
  var depth:Number = scope.getNextHighestDepth();
  var mc:MovieClip = scope.createEmptyMovieClip("mc_" + depth, depth);
    mc.beginFill(color);
  mc.lineTo(0, height);
  mc.lineTo(width, height);
  mc.lineTo(width, 0);
  mc.lineTo(0, 0);
  return mc;
}
```

### Voir également

ColorTransform (flash.geom.ColorTransform)

## concatenatedColorTransform (propriété Transform.concatenatedColorTransform)

```
public concatenatedColorTransform : ColorTransform [read-only]
```

Objet ColorTransform représentant les transformations de couleur combinées qui s'appliquent à cet objet et à l'ensemble de ses objets parents, jusqu'à la racine. Si les différentes transformations de couleur s'appliquent à différents niveaux, ces dernières transformations seront concaténées dans un objet ColorTransform pour cette propriété.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

### Exemple

L'exemple suivant applique deux objets Transform à la fois à un objet MovieClip parent et à un objet MovieClip enfant. Une variable `blueColorTransform` est ensuite appliquée à l'objet Transform `parentTrans` qui règle la couleur des deux objets MovieClip parent et enfant vers le bleu. Vous voyez que `child.concatenatedColorTransform` est la combinaison de `parentTrans` et `childTrans`.

```
import flash.geom.Transform;
import flash.geom.ColorTransform;

var parentRect:MovieClip = createRectangle(20, 80, 0xFF0000);
var childRect:MovieClip = createRectangle(10, 40, 0x00FF00, parentRect);

var parentTrans:Transform = new Transform(parentRect);
var childTrans:Transform = new Transform(childRect);

var blueColorTransform:ColorTransform = new ColorTransform(0, 1, 1, 1, 0,
  0, 255, 0);

parentTrans.colorTransform = blueColorTransform;

trace(childTrans.concatenatedColorTransform);
// (redMultiplier=0, greenMultiplier=1, blueMultiplier=1,
  alphaMultiplier=1, redOffset=0, greenOffset=0, blueOffset=255,
  alphaOffset=0)
trace(childTrans.colorTransform);
// (redMultiplier=1, greenMultiplier=1, blueMultiplier=1,
  alphaMultiplier=1, redOffset=0, greenOffset=0, blueOffset=0,
  alphaOffset=0)
trace(parentTrans.concatenatedColorTransform);
```

```
// (redMultiplier=0, greenMultiplier=1, blueMultiplier=1,
  alphaMultiplier=1, redOffset=0, greenOffset=0, blueOffset=255,
  alphaOffset=0)

function createRectangle(width:Number, height:Number, color:Number,
  scope:MovieClip):MovieClip {
  scope = (scope == undefined) ? this : scope;
  var depth:Number = scope.getNextHighestDepth();
  var mc:MovieClip = scope.createEmptyMovieClip("mc_" + depth, depth);
  mc.beginFill(color);
  mc.lineTo(0, height);
  mc.lineTo(width, height);
  mc.lineTo(width, 0);
  mc.lineTo(0, 0);
  return mc;
}
```

### Voir également

ColorTransform (flash.geom.ColorTransform)

## concatenatedMatrix (propriété Transform.concatenatedMatrix)

```
public concatenatedMatrix : Matrix [read-only]
```

Objet Matrix représentant les matrices de transformation combinées qui s'appliquent à cet objet et à l'ensemble de ses objets parents, jusqu'à la racine. Si les différentes matrices de transformation s'appliquent à différents niveaux, ces dernières seront concaténées en une seule matrice pour cette propriété.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

### Exemple

L'exemple suivant applique deux objets Transform, à la fois à un objet MovieClip parent et à un objet MovieClip enfant. Une `scaleMatrix` est ensuite appliquée à l'objet Transform `parentTrans`, qui mettent à l'échelle les objets MovieClip, parent et enfant. Vous voyez que `child.concatenatedMatrix` est la combinaison de `parentTrans` et `childTrans`.

```
import flash.geom.Transform;
import flash.geom.Matrix;

var parentRect:MovieClip = createRectangle(20, 80, 0xFF0000);
var childRect:MovieClip = createRectangle(10, 40, 0x00FF00, parentRect);

var parentTrans:Transform = new Transform(parentRect);
var childTrans:Transform = new Transform(childRect);
```

```
var scaleMatrix:Matrix = new Matrix();
scaleMatrix.scale(2, 2);

parentTrans.matrix = scaleMatrix;

trace(childTrans.concatenatedMatrix); // (a=2, b=0, c=0, d=2, tx=0, ty=0)
trace(childTrans.matrix); // (a=1, b=0, c=0, d=1, tx=0, ty=0)
trace(parentTrans.concatenatedMatrix); // (a=2, b=0, c=0, d=2, tx=0, ty=0)

function createRectangle(width:Number, height:Number, color:Number,
  scope:MovieClip):MovieClip {
  scope = (scope == undefined) ? this : scope;
  var depth:Number = scope.getNextHighestDepth();
  var mc:MovieClip = scope.createEmptyMovieClip("mc_" + depth, depth);
  mc.beginFill(color);
  mc.lineTo(0, height);
  mc.lineTo(width, height);
  mc.lineTo(width, 0);
  mc.lineTo(0, 0);
  return mc;
}
```

# matrix (propriété Transform.matrix)

```
public matrix : Matrix
```

Objet Matrix de transformation contenant des valeurs qui influent sur la mise à l'échelle, la rotation et la translation du clip.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

## Exemple

L'exemple suivant applique l'objet Matrix `scaleMatrix` à l'objet Transform `trans`. Cette Matrix met à l'échelle le MovieClip `rect` en le multipliant par deux.

```
import flash.geom.Transform;
import flash.geom.Matrix;

var rect:MovieClip = createRectangle(20, 80, 0xFF0000);

var trans:Transform = new Transform(rect);
trace(trans.matrix); // (a=1, b=0, c=0, d=1, tx=0, ty=0)

var scaleMatrix:Matrix = new Matrix();
scaleMatrix.scale(2, 2);

rect.onPress() = function() {
  trans.matrix = scaleMatrix;
```

```
  trace(trans.matrix); // (a=2, b=0, c=0, d=2, tx=0, ty=0)
}

function createRectangle(width:Number, height:Number, color:Number,
  scope:MovieClip):MovieClip {
  scope = (scope == undefined) ? this : scope;
  var depth:Number = scope.getNextHighestDepth();
  var mc:MovieClip = scope.createEmptyMovieClip("mc_" + depth, depth);
  mc.beginFill(color);
  mc.lineTo(0, height);
  mc.lineTo(width, height);
  mc.lineTo(width, 0);
  mc.lineTo(0, 0);
  return mc;
}
```

### Voir également

`Matrix (flash.geom.Matrix)`

## pixelBounds (propriété Transform.pixelBounds)

```
public pixelBounds : Rectangle
```

Objet rectangle qui définit le cadre de délimitation de l'objet MovieClip sur la Scène.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

### Exemple

L'exemple suivant crée un objet Transform `trans` et suit ses `pixelBounds` propriétés. Notez que `pixelBounds` renvoie un cadre de délimitation avec des valeurs égales aux méthodes `getBounds()` et `getRect()` de l'objet MovieClip.

```
import flash.geom.Transform;

var rect:MovieClip = createRectangle(20, 80, 0xFF0000);
var trans:Transform = new Transform(rect);
trace(trans.pixelBounds); // (x=0, y=0, w=20, h=80)

var boundsObj:Object = rect.getBounds();
trace(boundsObj.xMin); // 0
trace(boundsObj.yMin); // 0
trace(boundsObj.xMax); // 20
trace(boundsObj.yMax); // 80

var rectObj:Object = rect.getRect();
trace(rectObj.xMin); // 0
trace(rectObj.yMin); // 0
trace(rectObj.xMax); // 20
```

```
trace(rectObj.yMax); // 80

function createRectangle(width:Number, height:Number, color:Number,
  scope:MovieClip):MovieClip {
  scope = (scope == undefined) ? this : scope;
  var depth:Number = scope.getNextHighestDepth();
  var mc:MovieClip = scope.createEmptyMovieClip("mc_" + depth, depth);
  mc.beginFill(color);
  mc.lineTo(0, height);
  mc.lineTo(width, height);
  mc.lineTo(width, 0);
  mc.lineTo(0, 0);
  return mc;
}
```

# Transform, constructeur

```
public Transform(mc:MovieClip)
```

Crée un nouvel objet Transform attaché à l'objet MovieClip donné.

Une fois créé, le nouvel objet Transform peut être extrait en lisant la propriété `transform` de l'objet MovieClip donné.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

## Paramètres

*mc*`:MovieClip` - Objet MovieClip auquel le nouvel objet Transform s'applique.

## Exemple

L'exemple suivant crée le Transform `trans` et l'applique au rect. MovieClip. Vous constatez que les `trans` et `rect.transform` de l'objet Transform ne sont pas évalués comme égaux bien qu'ils contiennent les mêmes valeurs.

```
import flash.geom.Transform;

var rect:MovieClip = createRectangle(20, 80, 0xFF0000);

var trans:Transform = new Transform(rect);

trace(rect.transform == trans); // false

for(var i in trans) {
  trace(">> " + i + ": " + trans[i]);
  // >> pixelBounds: (x=0, y=0, w=20, h=80)
  // >> concatenatedColorTransform: (redMultiplier=1, greenMultiplier=1,
  blueMultiplier=1, alphaMultiplier=1, redOffset=0, greenOffset=0,
  blueOffset=0, alphaOffset=0)
```

```
  // >> colorTransform: (redMultiplier=1, greenMultiplier=1,
  blueMultiplier=1, alphaMultiplier=1, redOffset=0, greenOffset=0,
  blueOffset=0, alphaOffset=0)
  // >> concatenatedMatrix: (a=1, b=0, c=0, d=1, tx=0, ty=0)
  // >> matrix: (a=1, b=0, c=0, d=1, tx=0, ty=0)
}

for(var i in rect.transform) {
  trace(">> " + i + ": " + rect.transform[i]);
  // >> pixelBounds: (x=0, y=0, w=20, h=80)
  // >> concatenatedColorTransform: (redMultiplier=1, greenMultiplier=1,
  blueMultiplier=1, alphaMultiplier=1, redOffset=0, greenOffset=0,
  blueOffset=0, alphaOffset=0)
  // >> colorTransform: (redMultiplier=1, greenMultiplier=1,
  blueMultiplier=1, alphaMultiplier=1, redOffset=0, greenOffset=0,
  blueOffset=0, alphaOffset=0)
  // >> concatenatedMatrix: (a=1, b=0, c=0, d=1, tx=0, ty=0)
  // >> matrix: (a=1, b=0, c=0, d=1, tx=0, ty=0)
}

function createRectangle(width:Number, height:Number, color:Number,
  scope:MovieClip):MovieClip {
  scope = (scope == undefined) ? this : scope;
  var depth:Number = scope.getNextHighestDepth();
  var mc:MovieClip = scope.createEmptyMovieClip("mc_" + depth, depth);
  mc.beginFill(color);
  mc.lineTo(0, height);
  mc.lineTo(width, height);
  mc.lineTo(width, 0);
  mc.lineTo(0, 0);
  return mc;
}
```

# Video

```
Object
  |
  +-Video
```

```
public class Video
extends Object
```

La classe Video vous permet d'afficher un contenu vidéo Internet en direct sur la Scène sans l'intégrer dans votre fichier SWF. Vous capturez la vidéo à l'aide de `Camera.get()`. Dans les fichiers publiés pour Flash Player 7 et les versions ultérieures, vous pouvez également utiliser la classe Video pour lire les fichiers Flash Video (FLV) sur HTTP ou sur le système de fichiers local. Pour plus d'informations, consultez la section relative à la classe NetConnection et les entrées de classe NetStream.

Flash Player 7 prend en charge les fichiers Flash vidéo (FLV) codés avec le codec vidéo Sorenson Spark. Flash Player 8 prend en charge les fichiers Flash vidéo (FLV) codés avec le codec Sorenson ou On2 VP6 et prend également en charge un canal alpha. Le codec vidéo On2 VP6 utilise moins de largeur de bande que les technologies plus anciennes et offre des filtres supplémentaires de dégroupage et deringing.

Si votre contenu Flash charge de façon dynamique des fichiers vidéo (à l'aide du téléchargeage progressif ou du serveur de communication Flash), vous pouvez utiliser les vidéos On2 VP6 sans avoir à publier de nouveau votre fichier SWF pour Flash Player 8, tant que les utilisateurs exécutent Flash Player 8 pour afficher votre contenu. En diffusant en flux continu ou en téléchargeant la vidéo On2 VP6 en Flash SWF version 6 ou 7 et en lisant le contenu dans Flash Player 8, vous évitez de devoir à recréer vos fichiers SWF pour les utiliser dans Flash Player 8.

| Codec | Contenu (SWF)Version(version publiée) | Flash PlayerVersion (version requise pour la lecture) |
|---|---|---|
| Sorenson Spark | 6 | 6, 7, 8 |
| | 7 | 7, 8 |
| On2 VP6 | 6 | 8* |
| | 7 | 8 |
| | 8 | 8 |

* Si votre contenu Flash charge de façon dynamique des fichiers vidéo Flash (FLV), vous pouvez utiliser les vidéos On2 VP6 sans avoir à publier de nouveau votre fichier SWF pour Flash Player 8, tant que les utilisateurs exécutent Flash Player 8 pour afficher votre contenu. Seul Flash Player 8 prend en charge la publication et la lecture des vidéos On2 VP6.

Un objet Video peut être utilisé comme un clip. Comme les autres objets que vous placez sur la Scène, vous pouvez commander différentes propriétés des objets vidéo. Par exemple, vous pouvez déplacer l'objet Video autour de la Scène à l'aide de ses propriétés `_x` et `_y`, vous pouvez modifier sa taille à l'aide de ses propriétés `_height` et `_width`, etc...

Pour afficher le flux vidéo, remplacez d'abord un objet Video sur la Scène. Choisissez ensuite `Video.attachVideo()` pour joindre le flux vidéo à l'objet Video.

- Si le panneau Bibliothèque n'est pas visible, sélectionnez Bibliothèque > Fenêtre pour l'afficher.
- Ajoutez un objet Video intégré à la bibliothèque en cliquant sur le menu Options à droite de la barre de titre du panneau bibliothèque et en sélectionnant Nouvelle Video.

- Faites glisser l'objet Video sur la Scène et utilisez l'inspecteur des propriétés pour lui donner un nom d'occurrence unique, par ex. `my_video`. (Ne l'appelez pas Video.)

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

## Voir également

`NetConnection`, `NetStream`

## Résumé des propriétés

| Modificateurs | Propriété | Description |
|---|---|---|
| | `_alpha:Number` | Indique la valeur de transparence alpha de l'objet Video spécifié. |
| | `deblocking:Number` | Indique le type de filtre de dégroupage appliqué aux vidéos décodées en tant que partie du post-traitement. |
| | `_height:Number` | Indique la hauteur de l'objet Video, en pixels. |
| | `height:Number` [lecture seule] | Nombre entier spécifiant la hauteur en pixels du flux vidéo. |
| | `_name:String` | Indique le nom d'occurrence de l'objet Video spécifié. |
| | `_parent:MovieClip` | Indique le clip ou l'objet qui contient l'objet Video actuel. |
| | `_rotation:Number` | Indique la rotation de l'objet Video, en degrés, à partir de son orientation d'origine. |
| | `smoothing:Boolean` | Indique si la vidéo doit être lissée (interpolée) lors de son redimensionnement. |
| | `_visible:Boolean` | Indique si l'objet Video spécifié par *`my_video`* est visible. |
| | `_width:Number` | Indique la largeur de l'objet Video, en pixels. |
| | `width:Number` [lecture seule] | Nombre entier spécifiant la largeur en pixels du flux vidéo. |
| | `_x:Number` | Indique la coordonnée x d'un objet Video par rapport aux coordonnées locales du clip parent. |
| | `_xmouse:Number` [lecture seule] | Indique la coordonnée *x* de la position de la souris. |
| | `_xscale:Number` | Indique le redimensionnement horizontal (*`percentage`*) de l'objet Video tel qu'il est appliqué à partir de son point d'alignement. |

| Modificateurs | Propriété | Description |
|---|---|---|
| | `_y:Number` | Indique la coordonnée *y* d'un objet Video par rapport aux coordonnées locales du clip parent. |
| | `_ymouse:Number` [lecture seule] | Renvoie la coordonnée *y* de la position de la souris. |
| | `_yscale:Number` | Indique le redimensionnement vertical (*`percentage`*) de l'objet Video tel qu'il est appliqué à partir de son point d'alignement. |

*Propriétés héritées de la classe Object*

`constructeur (propriété Object.constructor), __proto__ (Object.__proto__ property), prototype (Object.prototype, propriété), __resolve (Object.__resolve, propriété)`

### Résumé de la méthode

| Modificateurs | Signature | Description |
|---|---|---|
| | `attachVideo(source:Object) : Void` | Spécifie un flux vidéo (*`source`*) à afficher dans le cadre de l'objet Video figurant sur la Scène. |
| | `clear() : Void` | Efface l'image actuellement affichée dans l'objet Video. |

*Méthodes héritées de la classe Object*

`addProperty (méthode Object.addProperty), hasOwnProperty (méthode Object.hasOwnProperty), isPropertyEnumerable (méthode Object.isPropertyEnumerable), isPrototypeOf (méthode Object.isPrototypeOf), registerClass (méthode Object.registerClass), toString (méthode Object.toString), unwatch (méthode Object.unwatch), valueOf (méthode Object.valueOf), watch (méthode Object.watch)`

## _alpha (propriété Video._alpha)

`public _alpha : Number`

Indique la valeur de transparence alpha de l'objet Video spécifié. Les valeurs valides sont comprises entre 0 (entièrement transparent) et 100 (entièrement opaque). La valeur par défaut est 100. Les objets d'un clip dont la propriété `_alpha` est définie sur 0 sont actifs, même s'ils sont invisibles.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

### Voir également

`_visible (propriété Video._visible)`

## attachVideo (méthode Video.attachVideo)

```
public attachVideo(source:Object) : Void
```

Spécifie un flux vidéo (*source*) à afficher dans le cadre de l'objet Video figurant sur la Scène. Le flux vidéo est un fichier FLV affiché au moyen de la commande `NetStream.play()` un objet Camera, ou `null`. Si la *source* est `null`, la vidéo n'est plus lue dans l'objet Video.

Vous n'êtes pas obligé d'utiliser cette méthode si le fichier FLV contient uniquement des données audio ; la partie audio des fichiers FLV est automatiquement lue lorsque la commande `NetStream.play()` est transmise.

Si vous souhaitez contrôler la partie audio associée à un fichier FLV, vous pouvez utiliser `MovieClip.attachAudio()` pour ajouter le son à un clip ; vous pouvez ensuite créer un objet Sound pour contrôler certains aspects du son. Pour plus d'informations, consultez `MovieClip.attachAudio()`.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

### Paramètres

*source*`:Object` - Objet Camera qui capture des données vidéo ou un objet NetStream. Pour annuler la connexion à l'objet Video, transmettez `null` pour *source*.

### Exemple

L'exemple suivant lit la vidéo en direct et en local :

```
var my_video:Video; //my_video is a Video object on the Stage
var active_cam:Camera = Camera.get();
my_video.attachVideo(active_cam);
```

L'exemple suivant lit un fichier enregistré précédemment et appelé myVideo.flv, qui est stocké dans le même répertoire que le fichier SWF.

```
var my_video:Video; // my_video is a Video object on the Stage
var my_nc:NetConnection = new NetConnection();
my_nc.connect(null);
var my_ns:NetStream = new NetStream(my_nc);
my_video.attachVideo(my_ns);
my_ns.play("video1.flv");
```

### Voir également

`Camera`, `NetStream`

# clear (méthode Video.clear)

```
public clear() : Void
```

Efface l'image actuellement affichée dans l'objet Video. Cette fonction est par exemple utile lorsque vous souhaitez afficher des informations en attente sans devoir cacher l'objet Video.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

## Exemple

L'exemple suivant interrompt et efface le fichier video1.flv qui est lu par un objet Video (appelé `my_video`) lorsque l'utilisateur clique sur l'occurrence `pause_btn`.

```
var pause_btn:Button;
var my_video:Video; // my_video is a Video object on the Stage
var my_nc:NetConnection = new NetConnection();
my_nc.connect(null);
var my_ns:NetStream = new NetStream(my_nc);
my_video.attachVideo(my_ns);
my_ns.play("video1.flv");
pause_btn.onRelease = function() {
  my_ns.pause();
  my_video.clear();
};
```

## Voir également

`attachVideo (méthode Video.attachVideo)`

# deblocking (propriété Video.deblocking)

```
public deblocking : Number
```

Indique le type de filtre de dégroupage appliqué aux vidéos décodées en tant que partie du post-traitement. Deux filtres de dégroupage sont disponibles : l'un dans le codec Sorenson et l'autre dans le codec On2 VP6. Les valeurs suivantes sont valides :

- 0 (par défaut)-Laissez le compresseur vidéo appliquer le filtre de dégroupage si nécessaire.
- 1-N'utilisez pas de filtre de dégroupage.
- 2-Utilisez le filtre de dégroupage Sorenson.
- 3-Utilisez le filtre de dégroupage On2 et pas de filtre de deringing.
- 4-Utilisez le filtre de dégroupage On2 et le filtre On2 rapide de deringing.
- 5-Utilisez le filtre de dégroupage On2 et le meilleur filtre On2 de deringing.
- 6-Identique à 5.
- 7-Identique à 5.

Si un mode supérieur à 2 est sélectionné pour la vidéo avec le codec Sorenson, le décodeur Sorenson applique par défaut le mode 2 en interne.

Le filtre de dégroupage a un effet sur les performances globales de lecture et il n'est généralement pas nécessaire pour la vidéo à large bande passante. Si votre système n'est pas assez puissant, il vous sera peut-être difficile de lire la vidéo avec ce filtre activé.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

### Exemple

L'exemple suivant lit le fichier video1.flv dans l'objet vidéo `my_video` et permet à l'utilisateur de modifier le comportement du filtre de dégroupage pour video1.flv. Ajoutez à votre fichier un objet vidéo appelé `my_video` et l'occurrence ComboBox appelée `deblocking_cb`, puis ajoutez le code ActionScript à votre fichier FLA ou AS.

```
var deblocking_cb:mx.controls.ComboBox;
var my_video:Video; // my_video is a Video object on the Stage
var my_nc:NetConnection = new NetConnection();
my_nc.connect(null);
var my_ns:NetStream = new NetStream(my_nc);
my_video.attachVideo(my_ns);
my_ns.play("video1.flv");

deblocking_cb.addItem({data:0, label:'Auto'});
deblocking_cb.addItem({data:1, label:'No'});
deblocking_cb.addItem({data:2, label:'Yes'});

var cbListener:Object = new Object();
cbListener.change = function(evt:Object) {
  my_video.deblocking = evt.target.selectedItem.data;
};
deblocking_cb.addEventListener("change", cbListener);
```

Utilisez l'occurrence ComboBox pour modifier le comportement du filtre de dégroupage pour video1.flv.

## _height (propriété Video._height)

```
public _height : Number
```

Indique la hauteur de l'objet Video, en pixels.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

### Voir également

`_width (propriété Video._width)`

# height (propriété Video.height)

```
public height : Number [lecture seule]
```

Nombre entier spécifiant la hauteur en pixels du flux vidéo. Pour les flux vidéo en direct, cette valeur est la même que la propriété `Camera.height` de l'objet Camera qui capture le flux vidéo. Pour les fichiers FLV, cette valeur est la hauteur du fichier qui a été exporté en tant que FLV.

Vous pouvez utiliser cette propriété, par exemple, pour garantir que l'utilisateur regarde la vidéo au même format que celui auquel il a été capturé, quel que soit le format réel de l'objet Video sur la Scène.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

## Exemple

L'exemple suivant définit les propriétés `_height` et `_width` de l'occurrence Video Symbol comme étant égales aux propriétés `height` et `width` du fichier FLV chargé.

Pour utiliser cet exemple, commencez par créer un nouveau symbole Video avec un nom d'occurrence de « myVideo » et placez-le dans le même contexte que ce script. Les propriétés `height` et `width` seront égales à zéro lorsque le code de statut `NetStream.Play.Start` sera déclenché. En redimensionnant la vidéo une fois que le statut de `NetStream.Buffer.Full` a été reçu, assurez-vous que sa taille soit correcte avant que le premier cadre de vidéo ne soit présenté.

```
var netConn:NetConnection = new NetConnection();
netConn.connect(null);
var netStrm:NetStream = new NetStream(netConn);

myVideo.attachVideo(netStrm);
netStrm.play("Video.flv");

netStrm.onStatus = function(infoObject:Object) {
  switch (infoObject.code) {
    case 'NetStream.Play.Start' :
    case 'NetStream.Buffer.Full' :
      myVideo._width = myVideo.width;
      myVideo._height = myVideo.height;
      break;
  }
}
```

## Voir également

`_height (propriété MovieClip._height)`, `width (propriété Video.width)`

## _name (propriété Video._name)

```
public _name : String
```

Indique le nom d'occurrence de l'objet Video spécifié.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

## _parent (propriété Video._parent)

```
public _parent : MovieClip
```

Indique le clip ou l'objet qui contient l'objet Video actuel. L'objet actuel est l'objet qui contient le code ActionScript faisant référence à `_parent`. Utilisez la propriété `_parent` pour spécifier un chemin relatif vers les clips ou les objets situés au-dessus de l'objet actuel.

Vous pouvez utiliser `_parent` pour remonter de plusieurs niveaux dans l'arborescence de la liste d'affichage, comme dans l'exemple suivant :

```
this._parent._parent._alpha = 20;
```

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

### Voir également

`_root`, propriété, `_target (propriété MovieClip._target)`

## _rotation (propriété Video._rotation)

```
public _rotation : Number
```

Indique la rotation de l'objet Video, en degrés, à partir de son orientation d'origine. Les valeurs comprises entre 0 et 180 représentent la rotation en sens horaire ; les valeurs comprises entre 0 et -180 représentent la rotation en sens anti-horaire. Les valeurs hors de cette plage sont ajoutées ou soustraites de 360 pour obtenir une valeur comprise dans la plage. Par exemple les instructions `my_video._rotation = 450` et `my_video._rotation = 90` sont les mêmes.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

## smoothing (propriété Video.smoothing)

```
public smoothing : Boolean
```

Indique si la vidéo doit être lissée (interpolée) lors de son redimensionnement. Pour faciliter le lissage, le lecteur doit être en mode haute qualité. La valeur par défaut est `false` (pas de lissage).

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

### Exemple

L'exemple suivant utilise un bouton (appelé `smoothing_btn`) pour faire basculer la propriété qui s'applique à la vidéo `my_video` lorsque cette dernière est lue dans un fichier SWF. Crée un bouton appelé `smoothing_btn` et ajoute le code ActionScript suivant au fichier FLA ou AS :

```
this.createTextField("smoothing_txt", this.getNextHighestDepth(), 0, 0,
  100, 22);
smoothing_txt.autoSize = true;

var my_nc:NetConnection = new NetConnection();
my_nc.connect(null);
var my_ns:NetStream = new NetStream(my_nc);
my_video.attachVideo(my_ns);
my_ns.play("video1.flv");
my_ns.onStatus = function(infoObject:Object) {
  updateSmoothing();
};
smoothing_btn.onRelease = function() {
  my_video.smoothing = !my_video.smoothing;
  updateSmoothing();
};
function updateSmoothing():Void {
  smoothing_txt.text = "smoothing = "+my_video.smoothing;
}
```

La méthode `MovieClip.getNextHighestDepth()` utilisée dans cet exemple requiert Flash Player 7 ou version ultérieure. Si votre fichier SWF comporte un composant de la version 2, utilisez la classe DepthManager des composants de la version 2 au lieu de la méthode `MovieClip.getNextHighestDepth()`.

## _visible (propriété Video._visible)

```
public _visible : Boolean
```

Indique si l'objet Video spécifié par *`my_video`* est visible.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

## _width (propriété Video._width)

```
public _width : Number
```

Indique la largeur de l'objet Video, en pixels.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

### Voir également

`_height (propriété Video._height)`

## width (propriété Video.width)

```
public width : Number [lecture seule]
```

Nombre entier spécifiant la largeur en pixels du flux vidéo. Pour les flux vidéo en direct, cette valeur est la même que la propriété `Camera.width` de l'objet Camera qui capture le flux vidéo. Pour les fichiers FLV, cette valeur est la largeur du fichier qui a été exporté en tant que FLV.

Vous pouvez utiliser cette propriété, par exemple, pour garantir que l'utilisateur regarde la vidéo au même format que celui auquel il a été capturé, quel que soit le format réel de l'objet Video sur la Scène.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

### Exemple

Consultez les exemples relatifs à `Video.height`.

## _x (Video._x property)

```
public _x : Number
```

Indique la coordonnée x d'un objet Video par rapport aux coordonnées locales du clip parent. Si un objet Video se trouve dans le scénario principal, son système de coordonnées se réfère alors au coin supérieur gauche de la Scène : (0, 0). Si l'objet Video est imbriqué dans un clip subissant des transformations, cet objet se trouve dans le système de coordonnées local du clip qui l'encadre. Ainsi, dans le cas d'un clip qui a effectué une rotation à 90 ° en sens anti-horaire, les enfants du clip héritent d'un système de coordonnées ayant effectué une rotation à 90 ° en sens anti-horaire. Les coordonnées de l'objet Video renvoient à la position du point d'alignement.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

### Voir également

`_xscale (propriété Video._xscale)`, `_y (propriété Video._y)`, `_yscale (propriété Video._yscale)`

## _xmouse (propriété Video._xmouse)

```
public _xmouse : Number [lecture seule]
```

Indique la coordonnée *x* de la position de la souris.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

### Voir également

Souris, _ymouse (propriété Video._ymouse)

## _xscale (propriété Video._xscale)

```
public _xscale : Number
```

Indique le redimensionnement horizontal (*percentage*) de l'objet Video tel qu'il est appliqué à partir de son point d'alignement. Le point d'alignement par défaut est (0,0).

Le redimensionnement du système de coordonnées local affecte les paramètres des propriétés _x et _y définis en pixels.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

### Voir également

_x (Video._x property), _y (propriété Video._y), _yscale (propriété Video._yscale), _width (propriété Video._width)

## _y (propriété Video._y)

```
public _y : Number
```

Indique la coordonnée *y* d'un objet Video par rapport aux coordonnées locales du clip parent. Si un objet Video se trouve dans le scénario principal, son système de coordonnées se réfère alors au coin supérieur gauche de la Scène : (0, 0). Si l'objet Video est imbriqué dans un clip subissant des transformations, cet objet se trouve dans le système de coordonnées local du clip qui l'encadre. Ainsi, dans le cas d'un clip qui a effectué une rotation à 90 ° en sens anti-horaire, les enfants du clip héritent d'un système de coordonnées ayant effectué une rotation à 90 ° en sens anti-horaire. Les coordonnées de l'objet Video renvoient à la position du point d'alignement.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

### Voir également

_x (Video._x property), _xscale (propriété Video._xscale), _yscale (propriété Video._yscale)

## _ymouse (propriété Video._ymouse)

```
public _ymouse : Number [lecture seule]
```

Renvoie la coordonnée *y* de la position de la souris.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

### Voir également

Souris, _xmouse (propriété Video._xmouse)

## _yscale (propriété Video._yscale)

```
public _yscale : Number
```

Indique le redimensionnement vertical (*percentage*) de l'objet Video tel qu'il est appliqué à partir de son point d'alignement. Le point d'alignement par défaut est (0,0).

Le redimensionnement du système de coordonnées local affecte les paramètres des propriétés _x et _y définis en pixels.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

### Voir également

_x (Video._x property), _xscale (propriété Video._xscale), _y (propriété Video._y), _height (propriété Video._height)

# XML

```
Object
  |
  +-XMLNode
    |
    +-XML
```

```
public class XML
extends XMLNode
```

Utilisez les méthodes et propriétés de la classe XML pour charger, analyser, envoyer, créer et manipuler des arborescences de documents XML.

Vous devez utiliser le constructeur `new XML()` pour créer un objet XML avant d'appeler une méthode quelconque de la classe XML.

Un document XML est représenté dans Flash par la classe XML. Chaque élément du document hiérarchique est représenté par un objet XMLNode.

Pour plus d'informations sur les méthodes et propriétés suivantes, consultez la section relative à la classe XMLNode, plus précisément `appendChild()`, `attributes`, `childNodes`, `cloneNode()`, `firstChild`, `hasChildNodes()`, `insertBefore()`, `lastChild`, `nextSibling`, `nodeName`, `nodeType`, `nodeValue`, `parentNode`, `previousSibling`, `removeNode()` et `toString()`.

Dans les versions précédentes du Guide de référence du langage ActionScript, les méthodes et les propriétés antérieures étaient documentées dans la section relative à la classe XML. Elles figurent désormais dans la section portant sur la classe XMLNode.

*Remarque :* Les objets XML et XMLNode sont modélisés d'après la recommandation W3C DOM Level, que vous trouverez dans : http://www.w3.org/tr/1998/REC-DOM-Level-1-19981001/level-one-core.html. Cette recommandation spécifie une interface Node et une interface Document. L'interface Document hérite de l'interface Node et ajoute des méthodes telles que `createElement()` et `createTextNode()`. Dans ActionScript, les objets XML et XMLNode sont conçus pour diviser la fonctionnalité le long de lignes similaires.

**Disponibilité :** ActionScript 1.0 ; Flash Player 5

## Voir également

`appendChild (XMLNode.appendChild, méthode)`, `attributes (XMLNode.attributes, propriété)`, `childNodes (XMLNode.childNodes, propriété)`, `cloneNode (XMLNode.cloneNode, méthode)`, `firstChild (XMLNode.firstChild, propriété)`, `hasChildNodes (XMLNode.hasChildNodes, méthode)`, `insertBefore (XMLNode.insertBefore, méthode)`, `lastChild (XMLNode.lastChild, propriété)`, `nextSibling (XMLNode.nextSibling, propriété)`, `nodeName (XMLNode.nodeName, propriété)`, `nodeType (XMLNode.nodeType, propriété)`, `nodeValue (XMLNode.nodeValue, propriété)`, `parentNode (XMLNode.parentNode, propriété)`, `previousSibling (XMLNode.previousSibling, propriété)`, `removeNode (XMLNode.removeNode, méthode)`, `toString (XMLNode.toString, méthode)`

## Résumé des propriétés

| Modificateurs | Propriété | Description |
|---|---|---|
| | `contentType:String` | Type de contenu MIME envoyé au serveur lorsque vous appelez la méthode `XML.send()` ou `XML.sendAndLoad()`. |
| | `docTypeDecl:String` | Spécifie des informations à propos de la déclaration `DOCTYPE` du document XML. |
| | `idMap:Object` | Objet contenant les nœuds du fichier XML auxquels un attribut `id` a été attribué. |
| | `ignoreWhite:Boolean` | La valeur par défaut est `false`. |
| | `loaded:Boolean` | La propriété indiquant si le document a été chargé avec succès. |
| | `status:Number` | Définit automatiquement et renvoie une valeur numérique qui indique si un document XML a été correctement analysé dans un objet XML. |
| | `xmlDecl:String` | Chaîne qui spécifie des informations sur une déclaration XML du document. |

*Propriétés héritées de la classe XMLNode*

`attributes (XMLNode.attributes, propriété)`, `childNodes (XMLNode.childNodes, propriété)`, `firstChild (XMLNode.firstChild, propriété)`, `lastChild (XMLNode.lastChild, propriété)`, `localName (XMLNode.localName, propriété)`, `namespaceURI (XMLNode.namespaceURI, propriété)`, `nextSibling (XMLNode.nextSibling, propriété)`, `nodeName (XMLNode.nodeName, propriété)`, `nodeType (XMLNode.nodeType, propriété)`, `nodeValue (XMLNode.nodeValue, propriété)`, `parentNode (XMLNode.parentNode, propriété)`, `prefix (XMLNode.prefix, propriété)`, `previousSibling (XMLNode.previousSibling, propriété)`

*Propriétés héritées de la classe Object*

`constructeur (propriété Object.constructor)`, `__proto__ (Object.__proto__ property)`, `prototype (Object.prototype, propriété)`, `__resolve (Object.__resolve, propriété)`

## Résumé des événements

| Evénement | Description |
|---|---|
| `onData = function(src:String) {}` | Appelé lorsque le texte XML a été totalement téléchargé à partir du serveur, ou lorsqu'une erreur survient au cours du téléchargement du texte XML à partir d'un serveur. |
| `onHTTPStatus = function(httpStatus:Number) {}` | Appelé quand Flash Player reçoit un code d'état HTTP du serveur. |
| `onLoad = function(success:Boolean) {}` | Appelé par Flash Player lorsqu'un document XML est reçu en provenance du serveur. |

## Récapitulatif des constructeurs

| Signature | Description |
|---|---|
| `XML(text:String)` | Crée un nouvel objet XML. |

## Résumé de la méthode

| Modificateurs | Signature | Description |
|---|---|---|
| | `addRequestHeader(header:Object, headerValue:String) : Void` | Ajoute ou modifie les en-têtes de requête HTTP (tels que `Content-Type` ou `SOAPAction`) envoyés avec les actions `POST`. |
| | `createElement(name:String) : XMLNode` | Crée un nouvel élément XML avec le nom spécifié dans le paramètre. |
| | `createTextNode(value:String) : XMLNode` | Crée un nouveau nœud XML avec le texte spécifié. |
| | `getBytesLoaded() : Number` | Renvoie le nombre d'octets chargés (transmis en continu) pour le document XML. |
| | `getBytesTotal() : Number` | Renvoie la taille, en octets, du document XML. |
| | `load(url:String) : Boolean` | Charge un document XML à partir de l'URL spécifié et remplace le contenu de l'objet XML spécifié par les données XML téléchargées. |
| | `parseXML(value:String) : Void` | Analyse le texte XML spécifié dans le paramètre `value` et renseigne l'objet XML spécifié avec l'arborescence XML obtenue. |

| Modificateurs | Signature | Description |
|---|---|---|
| | `send(url:String, [target:String], [method:String]) : Boolean` | Code l'objet XML spécifié dans un document XML et l'envoie à l'URL `target` spécifié. |
| | `sendAndLoad(url:String, resultXML:XML) : Void` | Code l'objet XML spécifié en un document XML, l'envoie à l'URL spécifiée à l'aide de la méthode `POST`, télécharge la réponse du serveur et la charge dans le `resultXMLobject` spécifié dans les paramètres. |

*Méthodes héritées de la classe XMLNode*

`appendChild (XMLNode.appendChild, méthode)`, `cloneNode (XMLNode.cloneNode, méthode)`, `getNamespaceForPrefix (XMLNode.getNamespaceForPrefix, méthode)`, `getPrefixForNamespace (XMLNode.getPrefixForNamespace, méthode)`, `hasChildNodes (XMLNode.hasChildNodes, méthode)`, `insertBefore (XMLNode.insertBefore, méthode)`, `removeNode (XMLNode.removeNode, méthode)`, `toString (XMLNode.toString, méthode)`

*Méthodes héritées de la classe Object*

`addProperty (méthode Object.addProperty)`, `hasOwnProperty (méthode Object.hasOwnProperty)`, `isPropertyEnumerable (méthode Object.isPropertyEnumerable)`, `isPrototypeOf (méthode Object.isPrototypeOf)`, `registerClass (méthode Object.registerClass)`, `toString (méthode Object.toString)`, `unwatch (méthode Object.unwatch)`, `valueOf (méthode Object.valueOf)`, `watch (méthode Object.watch)`

# addRequestHeader (méthode XML.addRequestHeader)

`addRequestHeader public(header:Object, headerValue:String) : Void`

Ajoute ou modifie les en-têtes de requête HTTP (tels que `Content-Type` ou `SOAPAction`) envoyés avec les actions `POST`. Dans la première utilisation, vous transmettez deux chaînes à la méthode : `header` et `headerValue`. Au cours de la deuxième utilisation, vous transmettez un tableau de chaînes, en alternant les noms d'en-têtes et les valeurs d'en-têtes.

En cas d'appels multiples pour définir le même nom d'en-tête, chaque valeur successive remplace la valeur définie dans l'appel précédent.

Vous ne pouvez pas ajouter ou modifier les en-têtes HTTP standard suivants à l'aide de cette méthode : `Accept-Ranges`, `Age`, `Allow`, `Allowed`, `Connection`, `Content-Length`, `Content-Location`, `Content-Range` , `ETag`, `Host`, `Last-Modified`, `Locations`, `Max-Forwards`, `Proxy-Authenticate`, `-Authorization`, `Public`, `Range`, `Retry-After`, `Server`, `TE`, `Trailer`, `-Encoding`, `Upgrade`, `URI`, `Vary`, `Via`, `Warning`, et `WWW-Authenticate`.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

### Paramètres

*`header`*`:Object` - Chaîne qui représente un nom d'en-tête de requête HTTP.

*`headerValue`*`:String` - Chaîne qui représente la valeur associée à `header`.

### Exemple

L'exemple suivant ajoute un en-tête HTTP personnalisé, `SOAPAction` avec la valeur `Foo`, à un objet XML appelé `my_xml` :

```
my_xml.addRequestHeader("SOAPAction", "'Foo'");
```

L'exemple suivant crée un tableau appelé `headers` qui contient deux en-têtes HTTP interchangeables et leurs valeurs. Le tableau est transmis en tant que paramètre à la méthode `addRequestHeader()`.

```
var headers:Array = new Array("Content-Type", "text/plain", "X-
  ClientAppVersion", "2.0");
my_xml.addRequestHeader(headers);
```

### Voir également

`addRequestHeader (méthode LoadVars.addRequestHeader)`

## contentType (XML.contentType, propriété)

```
contentType public : String
```

Type de contenu MIME envoyé au serveur lorsque vous appelez la méthode `XML.send()` ou `XML.sendAndLoad()`. Le type par défaut est `application/x-www-form-urlencoded`, qui est le type de contenu MIME standard utilisé pour la plupart des formes HTML.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

### Exemple

L'exemple suivant crée un document XML et vérifie son type de contenu par défaut :

```
// create a new XML document
var doc:XML = new XML();
```

```
// trace the default content type
trace(doc.contentType); // output: application/x-www-form-urlencoded
```

L'exemple suivant définit un paquet XML et le type de contenu de l'objet XML. Les données sont alors envoyées à un serveur et les résultats s'affichent dans une fenêtre de navigateur.

```
var my_xml:XML = new XML("<highscore><name>Ernie</name><score>13045</
  score></highscore>");
my_xml.contentType = "text/xml";
my_xml.send("http://www.flash-mx.com/mm/highscore.cfm", "_blank");
```

Appuyez sur F12 pour tester cet exemple dans un navigateur.

### Voir également

`send (XML.send, méthode)`, `sendAndLoad (XML.sendAndLoad, méthode)`

## createElement (méthode nodeML.createElement)

```
public createElement(name:String) : XMLNode
```

Crée un nouvel élément XML avec le nom spécifié dans le paramètre. Le nouvel élément n'a initialement pas de parent, pas d'enfants et pas de frères. La méthode renvoie une référence au nouvel objet XML créé qui représente l'élément. La méthode `XML.createTextNode()` et celle-ci sont les méthodes du constructeur de création de nœuds pour un objet XML.

**Disponibilité :** ActionScript 1.0 ; Flash Player 5

### Paramètres

*`name`*`:String` - Nom de balise de l'élément XML en cours de création.

### Valeur renvoyée

`XMLNode` - Objet XMLNode ; un élément XML.

### Exemple

L'exemple suivant crée trois nœuds XML avec la méthode `createElement()` :

```
// create an XML document
var doc:XML = new XML();

// create three XML nodes using createElement()
var element1:XMLNode = doc.createElement("element1");
var element2:XMLNode = doc.createElement("element2");
var element3:XMLNode = doc.createElement("element3");

// place the new nodes into the XML tree
doc.appendChild(element1);
element1.appendChild(element2);
```

```
element1.appendChild(element3);

trace(doc);
// output: <element1><element2 /><element3 /></element1>
```

### Voir également

createTextNode (méthode XML.createTextNode)

## createTextNode (méthode XML.createTextNode)

```
public createTextNode(value:String) : XMLNode
```

Crée un nouveau nœud XML avec le texte spécifié. Le nouveau nœud n'a initialement pas de parent et les nœuds de texte ne peuvent pas avoir d'enfants ou de frères. Cette méthode renvoie une référence à l'objet XML qui représente le nouveau nœud de texte. La méthode `XML.createElement()` et celle-ci sont les méthodes du constructeur de création de nœuds pour un objet XML.

**Disponibilité :** ActionScript 1.0 ; Flash Player 5

### Paramètres

*`value`*`:String` - Chaîne ; le texte utilisé pour créer le nouveau nœud de texte.

### Valeur renvoyée

`XMLNode` - Objet XMLNode.

### Exemple

L'exemple suivant crée deux nœuds de texte XML avec la méthode `createTextNode()` et les place dans les nœuds XML existants :

```
// create an XML document
var doc:XML = new XML();

// create three XML nodes using createElement()
var element1:XMLNode = doc.createElement("element1");
var element2:XMLNode = doc.createElement("element2");
var element3:XMLNode = doc.createElement("element3");

// place the new nodes into the XML tree
doc.appendChild(element1);
element1.appendChild(element2);
element1.appendChild(element3);

// create two XML text nodes using createTextNode()
var textNode1:XMLNode = doc.createTextNode("textNode1 String value");
```

```
var textNode2:XMLNode = doc.createTextNode("textNode2 String value");

// place the new nodes into the XML tree
element2.appendChild(textNode1);
element3.appendChild(textNode2);

trace(doc);
// output (with line breaks added between tags):
// <element1>
// <element2>textNode1 String value</element2>
// <element3>textNode2 String value</element3>
// </element1>
```

### Voir également

createElement (méthode nodeML.createElement)

## docTypeDecl (XML.docTypeDecl, propriété)

```
public docTypeDecl : String
```

Spécifie des informations à propos de la déclaration `DOCTYPE` du document XML. Après analyse du texte XML dans un objet XML, la propriété `XML.docTypeDecl` de l'objet XML est défini sur la déclaration `DOCTYPE` du texte du document XML (par exemple `<!DOCTYPE greeting SYSTEM "hello.dtd">`). Cette propriété est définie à l'aide d'une représentation sous forme de chaîne de la déclaration `DOCTYPE`, pas d'un objet de nœud XML.

Le programme d'analyse ActionScript XML n'est pas un programme d'analyse de validation. La déclaration `DOCTYPE` est lue par le programme d'analyse et enregistrée dans la propriété `XML.docTypeDecl`, mais aucune validation Dtd n'est effectuée.

En cas d'absence de déclaration `DOCTYPE` au cours d'une opération d'analyse, la propriété `XML.docTypeDecl` est définie sur `undefined`. La méthode `XML.toString()` produit le contenu de `XML.docTypeDecl` immédiatement après la déclaration XML enregistrée dans `XML.xmlDecl`, et avant tout autre texte dans l'objet XML. Si `XML.docTypeDecl` n'est pas définie, aucune déclaration `DOCTYPE` n'est produite.

**Disponibilité :** ActionScript 1.0 ; Flash Player 5

### Exemple

L'exemple suivant utilise la propriété `XML.docTypeDecl` pour définir la déclaration `DOCTYPE` d'un objet XML :

```
my_xml.docTypeDecl = "<!DOCTYPE greeting SYSTEM \"hello.dtd\">";
```

### Voir également

`xmlDecl (XML.xmlDecl, propriété)`

## getBytesLoaded (XML.getBytesLoaded, méthode)

```
public getBytesLoaded() : Number
```

Renvoie le nombre d'octets chargés (transmis en continu) pour le document XML. Vous pouvez comparer la valeur de `getBytesLoaded()`) à la valeur de `getBytesTotal()` pour déterminer le pourcentage d'un document XML qui a été chargé.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

### Valeur renvoyée

`Number` - Entier indiquant le nombre d'octets chargés.

### Exemple

L'exemple suivant indique comment utiliser la méthode `XML.getBytesLoaded()` en conjonction avec la méthode `XML.getBytesTotal()` pour suivre la progression d'une commande `XML.load()`. Vous devez remplacer le paramètre URL de la commande `XML.load()` de façon à faire référence à un fichier XML valide utilisant le code HTTP. Si vous tentez d'utiliser cet exemple pour charger un fichier local résidant sur votre disque dur, il ne fonctionnera pas correctement car, en mode de test d'animation, Flash Player charge intégralement les fichiers locaux.

```
// create a new XML document
var doc:XML = new XML();

var checkProgress = function(xmlObj:XML) {
  var bytesLoaded:Number = xmlObj.getBytesLoaded();
  var bytesTotal:Number = xmlObj.getBytesTotal();
  var percentLoaded:Number = Math.floor((bytesLoaded / bytesTotal ) 100);
  trace ("milliseconds elapsed: " + getTimer());
  trace ("bytesLoaded: " + bytesLoaded);
  trace ("bytesTotal: " + bytesTotal);
  trace ("percent loaded: " + percentLoaded);
  trace ("---------------------------------");
}

doc.onLoad = function(success:Boolean) {
  clearInterval(intervalID);
  trace("intervalID: " + intervalID);
}
doc.load("[place a valid URL pointing to an XML file here]");
var intervalID:Number = setInterval(checkProgress, 100, doc);
```

### Voir également

`getBytesTotal (XML.getBytesTotal, méthode)`

## getBytesTotal (XML.getBytesTotal, méthode)

```
public getBytesTotal() : Number
```

Renvoie la taille, en octets, du document XML.

**Disponibilité :** ActionScript 1.0 ; Flash Player 6

### Valeur renvoyée

`Number` - Entier.

### Exemple

Consultez l'exemple relatif à `XML.getBytesLoaded()`.

### Voir également

`getBytesLoaded (XML.getBytesLoaded, méthode)`

## idMap (XML.idMap, propriété)

```
public idMap : Object
```

Objet contenant les nœuds du fichier XML auxquels un attribut `id` a été attribué. Les noms des propriétés de l'objet (contenant un nœud chacun) correspondent aux valeurs des attributs `id`.

Considérez l'objet XML suivant :

```
<employee id='41'>
<name>
John Doe
</name>
<address>
601 Townsend St.
</address>
</employee>

<employee id='42'>
<name>
Jane Q. Public
</name>
</employee>
<department id="IT">
Information Technology
```

```
</department>
```

Dans cet exemple, la propriété `idMap` de cet objet XML est un Objet avec trois propriétés : `41`, `42` et `IT`. Chacune de ces propriétés est un XMLNode avec la valeur `id` correspondante. Par exemple, la propriété `IT` de l'objet `idMap` est ce nœud :

```
<department id="IT">
Information Technology
</department>
```

Vous devez utiliser la méthode `parse()` sur l'objet XML pour la propriété `idMap` à être instancié.

Si plus d'un XMLNode a la même valeur `id`, la propriété correspondante de l'objet `idNode` est celle du dernier nœud analysé, de la manière suivante :

```
var x1:XML = new XML("<a id='1'><b id='2' /><c id='1' /></a>");
x2 = new XML();
x2.parseXML(x1);
trace (x2.idMap['1']);
```

L'exemple suivant donnera en sortie le nœud `<c>` :

```
<c id='1' />
```

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

## Exemple

Vous pouvez créer un texte fichier nommé `idMapTest.xml` contenant le texte suivant .

```
<?xml version="1.0"?>
<doc xml:base="http://example.org/today/" xmlns:xlink="http://www.w3.org/
 1999/xlink">
<head>
<title>Virtual Library</title>
</head>
<body>
<paragraph id="linkP1">See <link xlink:type="simple"
 xlink:href="new.xml">what's
new</link>!</paragraph>
<paragraph>Check out the hot picks of the day!</paragraph>
<olist xml:base="/hotpicks/">
<item>
<link id="foo" xlink:type="simple" xlink:href="pick1.xml">Hot Pick #1</
 link>
</item>
<item>
<link id="bar" xlink:type="simple" xlink:href="pick2.xml">Hot Pick #2</
 link>
</item>
```

```
<item>
<link xlink:type="simple" xlink:href="pick3.xml">Hot Pick #3</link>
</item>
</olist>
</body>
</doc>
```

Vous pouvez ensuite créer un fichier SWF dans le même répertoire que le fichier XML. Vous pouvez inclure le script suivant dans le SWF.

```
var readXML = new XML();
readXML.load("idMapTest.xml");
readXML.onLoad = function(success) {
  myXML = new XML();
  myXML.parseXML(readXML);
  for (var x in myXML.idMap){
    trace('idMap.' + x + " = " + newline + myXML.idMap[x]);
    trace('_____________' + newline);
  }
}
```

Lorsque vous testez le fichier SWF, la sortie suivante est générée.

```
idMap.bar =
<link id="bar" xlink:type="simple" xlink:href="pick2.xml">Hot Pick #2</
 link>
_____________

idMap.foo =
<link id="foo" xlink:type="simple" xlink:href="pick1.xml">Hot Pick #1</
 link>
_____________

idMap.linkP1 =
<paragraph id="linkP1">See <link xlink:type="simple"
 xlink:href="new.xml">what's

new</link>!</paragraph>
_____________
```

## ignoreWhite (XML.ignoreWhite, propriété)

```
public ignoreWhite : Boolean
```

La valeur par défaut est `false`. Lorsque le réglage est `true`, les nœuds de texte qui ne contiennent que des espaces vierges sont supprimés au cours de l'analyse. Les nœuds de texte qui contiennent un espace vierge avant ou après leur nom ne sont pas affectés.

Usage 1 : Vous pouvez définir la propriété `ignoreWhite` pour les objets XML individuels, comme indiqué par le code suivant :

```
my_xml.ignoreWhite = true;
```

Usage 2 : Vous pouvez définir la propriété `ignoreWhite` par défaut pour les objets XML, comme indiqué par le code suivant :

```
XML.prototype.ignoreWhite = true;
```

**Disponibilité :** ActionScript 1.0 ; Flash Player 5

## Exemple

L'exemple suivant charge un fichier XML avec un nœud texte qui contient uniquement un espace blanc ; la balise `foyer` contient quatorze caractères d'espacement. Pour exécuter cet exemple, créez un fichier texte appelé *flooring.xml* et copiez les balises suivantes dedans :

```
 <house>
 <kitchen> ceramic tile </kitchen>
 <bathroom>linoleum</bathroom>
 <foyer> </foyer>
 </house>
```

Créez un document Flash appelé *flooring.fla* et enregistrez-le dans le même répertoire que le fichier XML. Placez le code suivant sur le scénario principal :

```
// Create a new XML object.
var flooring:XML = new XML();

// Set the ignoreWhite property to true (default value is false)
flooring.ignoreWhite = true;

// After loading is complete, trace the XML object.
flooring.onLoad = function(success:Boolean) {
  trace(flooring);
}

// Load the XML into the flooring object.
flooring.load("flooring.xml");

// Output (line breaks added for clarity):
<house>
  <kitchen> ceramic tile </kitchen>
  <bathroom>linoleum</bathroom>
  <foyer />
</house>
```

Ensuite, si vous définissez les paramètres de `flooring.ignoreWhite` sur `false` ou retirez simplement cette ligne de code, les quatorze espaces de la balise foyer seront préservés :

```
...
// Set the ignoreWhite property to false (default value).
flooring.ignoreWhite = false;
...
// Output (line breaks added for clarity):
<house>
  <kitchen> ceramic tile </kitchen>
  <bathroom>linoleum</bathroom>
  <foyer> </foyer>
</house>
```

Les fichiers XML_blogTracker.fla et XML_languagePicker.fla du dossier des échantillons ActionScript contiennent également un exemple de code. Vous trouverez ci-dessous les chemins type de ce dossier :

- Windows : *lecteur d'amorçage*\Program Files\Macromedia\Flash 8\Samples and Tutorials\Samples\ActionScript
- Macintosh : *disque dur Macintosh*/Applications/Macromedia Flash 8/Samples and Tutorials/Samples/ActionScript
-

## load (XML.load, méhode)

```
public load(url:String) : Boolean
```

Charge un document XML à partir de l'URL spécifié et remplace le contenu de l'objet XML spécifié par les données XML téléchargées. L'URL est relative et appelée en utilisant HTTP. Le processus de chargement est asynchrone ; il ne finit pas immédiatement après l'exécution de la méthode `load()`.

Lorsque la méthode `load()` est exécutée, la propriété d'objet XML `loaded`est définie sur `false`. Lorsque le téléchargement des données XML se termine, la propriété `loaded` est définie sur `true` et le gestionnaire d'événements `onLoad` est appelé. Les données XML ne sont pas analysées avant la fin du téléchargement. Si l'objet XML contenait précédemment des arborescences XML, elles sont supprimées.

Vous pouvez définir une fonction personnalisée qui s'exécute lorsque le gestionnaire des événements `onLoad` de l'objet XML est appelé.

*Remarque :* Si un fichier en cours de chargement contient des caractères non-ASCII (comme s'en trouvent dans les langues autres que l'anglais), nous vous recommandons d'enregistrer le fichier avec codage UTF-8 ou UTF-16, plutôt que sous un format non-Unicode, ASCII par exemple.

Lorsque vous employez cette méthode, considérez le modèle de sécurité Flash Player :

Pour Flash Player 8 :

- Le chargement de données n'est pas autorisé si le fichier SWF appelant est dans le sandbox local-with-file-system et que la ressource cible provient d'un sandbox de réseau.
- Le chargement de données n'est pas autorisé si le fichier SWF appelant provient d'un sandbox de réseau et que la ressource cible est locale.

Pour plus d'informations, consultez :

- Chapitre 17, « Understanding Security » dans *Learning ActionScript 2.0 in Flash*
- Le livre blanc Flash Player 8 Security disponible à l'adresse : http://www.macromedia.com/go/fp8_security
- Le livre blanc Flash Player 8 Security-Related APIs disponible à l'adresse : http://www.macromedia.com/go/fp8_security_apis

Pour Flash Player 7 et postérieur, les sites Web permettent l'accès inter-domaines à des ressources via un fichier de régulation inter-domaines. Dans les fichiers SWF d'une version exécutée dans Flash Player 7 ou une version ultérieure, le paramètre `url` doit se trouver exactement dans le même domaine. Par exemple, un fichier SWF à l'adresse www.someDomain.com peut charger uniquement des données provenant de sources situées aussi à l'adresse store.someDomain.com.

Dans les fichiers SWF exécutés dans une version du lecteur antérieure à Flash Player 7, le paramètre `url` doit être dans le même superdomaine que le fichier SWF qui transmet cet appel. Un *superdomaine* est dérivé en supprimant le composant le plus à gauche de l'URL d'un fichier. Par exemple, un fichier SWF à l'adresse www.someDomain.com peut charger des données provenant de sources à l'adresse store.someDomain.com, étant donné que les deux fichiers sont dans le même superdomaine de someDomain.com.

**Disponibilité :** ActionScript 1.0 ; Flash Player 5

## Paramètres

*`url`*`:String` - Chaîne qui représente l'URL du document XML à charger. Si le fichier SWF effectuant cet appel s'exécute dans un navigateur Web, `url` doit appartenir au même domaine que le fichier SWF.

## Valeur renvoyée

`Boolean` - Valeur booléenne `false` si aucun paramètre (null) n'est transmis; `true` sinon. Employez le gestionnaire d'événement `onLoad()` pour vérifier le succès d'un document XML chargé.

### Exemple

L'exemple de code suivant emploie la méthode `XML.load()` :

```
// Create a new XML object.
var flooring:XML = new XML();

// Set the ignoreWhite property to true (default value is false).
flooring.ignoreWhite = true;

// After loading is complete, trace the XML object.
flooring.onLoad = function(success) {
  trace(flooring);
};

// Load the XML into the flooring object.
flooring.load("flooring.xml");
```

Pour plus de détails sur le contenu du fichier flooring.xml et les résultats de cet exemple, consultez l'exemple de la propriété `XML.ignoreWhite`.

### Voir également

`ignoreWhite (XML.ignoreWhite, propriété)`, `loaded (XML.loaded, propriété)`, `onLoad (XML.onLoad, gestionnaire)`, `useCodepage (propriété System.useCodepage)`

## loaded (XML.loaded, propriété)

```
public loaded : Boolean
```

La propriété indiquant si le document a été chargé avec succès. En l'absence de gestionnaire d'événements `onLoad()` personnalisé et défini pour l'objet XML, cette propriété est définie sur `true` par Flash Player lorsque le processus de chargement de documents initié par l'appel `XML.load()` s'est terminé avec succès ; sinon, elle est définie sur `false`. Cependant, si vous définissez un comportement personnalisé pour le gestionnaire d'événements `onLoad()` de l'objet XML, vous devez définir `onload` dans cette fonction.

**Disponibilité :** ActionScript 1.0 ; Flash Player 5

### Exemple

L'exemple suivant utilise la propriété `XML.loaded` dans un script simple.

```
var my_xml:XML = new XML();
my_xml.ignoreWhite = true;
my_xml.onLoad = function(success:Boolean) {
  trace("success: "+success);
  trace("loaded: "+my_xml.loaded);
  trace("status: "+my_xml.status);
```

```
};
my_xml.load("http://www.flash-mx.com/mm/problems/products.xml");
```

Des informations s'affichent dans le panneau Sortie lorsque le gestionnaire Flash appelle le gestionnaire `onLoad()`. Si cet appel se termine correctement, `true` s'affiche pour le statut `loaded` dans le panneau Sortie.

```
success: true
loaded: true
status: 0
```

### Voir également

`load (XML.load, méhode)`, `onLoad (XML.onLoad, gestionnaire)`

## onData (XML.onData, gestionnaire)

```
onData = function(src:String) {}
```

Appelé lorsque le texte XML a été totalement téléchargé à partir du serveur, ou lorsqu'une erreur survient au cours du téléchargement du texte XML à partir d'un serveur. Ce gestionnaire est appelé avant l'analyse du XML, et vous pouvez l'utiliser pour appeler une routine d'analyse personnalisée au lieu d'utiliser le programme d'analyse XML Flash. Le paramètre src est une chaîne qui contient du texte XML téléchargé à partir du serveur, sauf si une erreur survient au cours du téléchargement, dans ce cas le paramètre `src` est `undefined`.

Par défaut, le gestionnaire d'événements `XML.onData` appelle `XML.onLoad`. Vous pouvez supplanter le gestionnaire d'événements `XML.onData` par un comportement personnalisé, mais `XML.onLoad` n'est pas appelé sauf si vous l'appelez dans votre implémentation de `XML.onData`.

**Disponibilité :** ActionScript 1.0 ; Flash Player 5

### Paramètres

*`src`*`:String` - Chaîne ou `undefined`; les données brutes, en général en format XML, envoyés par le serveur.

### Exemple

L'exemple suivant permet de voir l'aspect par défaut du gestionnaire d'événements XML.onData :

```
XML.prototype.onData = function (src:String) {
  if (src == undefined) {
    this.onLoad(false);
  } else {
    this.parseXML(src);
```

```
    this.loaded = true;
    this.onLoad(true);
  }
}
```

Vous pouvez ignorer le gestionnaire d'événements XML.onData pour intercepter le texte XML sans l'analyser.

### Voir également

`onLoad (XML.onLoad, gestionnaire)`

## onHTTPStatus (XML.onHTTPStatus, gestionnaire)

```
onHTTPStatus = function(httpStatus:Number) {}
```

Appelé quand Flash Player reçoit un code d'état HTTP du serveur. Ce gestionnaire vous laisse capturer et manipuler les codes de statut HTTP.

Le gestionnaire `onHTTPStatus` est appelé avant `onData`, qui déclenche des appels à `onLoad` avec une valeur de `undefined` si le chargement échoue. Il est important de noter qu'après le déclenchement de `onHTTPStatus`, `onData` est toujours déclenché, que vous supplantiez `onHTTPStatus` ou non. Pour un emploi optimal du gestionnaire `onHTTPStatus`, écrivez une fonction appropriée pour intercepter le résultat de l'appel `onHTTPStatus` ; vous pouvez ensuite l'employer dans vos fonctions de gestion `onData` ou `onLoad`. Si `onHTTPStatus` n'est pas appelé, cela signifie que le joueur n'a pas essayé d'envoyer la requête URL. Ceci est possible car la requête viole les règles de sécurité du sandbox pour SWF.

Si Flash Player ne peut pas obtenir du serveur un code de statut ou ne peut pas communiquer avec le serveur, la valeur par défaut (0) est transmise à votre code ActionScript. La valeur 0 peut être générée par tous les joueurs, comme si un URL malformulé avait été requis, et est toujours générée par le module Flash Player lorsqu'il est exécuté dans les navigateurs suivants, lesquels ne transmettent pas les codes de statut HTTP au joueur : Netscape, Mozilla, Safari, Opera, ou Internet Explorer pour Macintosh.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

### Paramètres

*`httpStatus`*`:Number` - Code de statut HTTP renvoyé par le serveur. Par exemple, la valeur 404 indique que le serveur n'a rien trouvé qui corresponde à l'URI requis. Les valeurs de code HTTP sont répertoriées dans les sections 10.4 et 10.5 de la spécification HTTP à l'adresse ftp://ftp.isi.edu/in-notes/rfc2616.txt.

## Exemple

L'exemple suivant montre l'emploi de la méthode `onHTTPStatus` d'aide au débogage. L'exemple rassemble des codes de statut HTTP et affecte leur valeur et type à une instance de l'objet `XML` (notez que cet exemple crée les membres `this.httpStatus` de l'instance et `this.httpStatusType` au cours de l'exécution). La méthode `onData` les emploie pour retracer les informations sur la réponse HTTP qui pourrait être utile au débogage.

```
var myXml:XML = new XML();

myXml.onHTTPStatus = function(httpStatus:Number) {
  this.httpStatus = httpStatus;
  if(httpStatus < 100) {
    this.httpStatusType = "flashError";
  }
  else if(httpStatus < 200) {
    this.httpStatusType = "informational";
  }
  else if(httpStatus < 300) {
    this.httpStatusType = "successful";
  }
  else if(httpStatus < 400) {
    this.httpStatusType = "redirection";
  }
  else if(httpStatus < 500) {
    this.httpStatusType = "clientError";
  }
  else if(httpStatus < 600) {
    this.httpStatusType = "serverError";
  }
}

myXml.onData = function(src:String) {
  trace(">> " + this.httpStatusType + ": " + this.httpStatus);
  if(src != undefined) {
    this.parseXML(src);
    this.loaded = true;
    this.onLoad(true);
  }
  else {
    this.onLoad(false);
  }
}

myXml.onLoad = function(success:Boolean) {
}

myXml.load("http://weblogs.macromedia.com/mxna/xml/
  rss.cfm?query=byMostRecent&languages=1");
```

### Voir également

onHTTPStatus (gestionnaire LoadVars.onHTTPStatus), load (XML.load, méhode), sendAndLoad (XML.sendAndLoad, méthode)

## onLoad (XML.onLoad, gestionnaire)

```
onLoad = function(success:Boolean) {}
```

Appelé par Flash Player lorsqu'un document XML est reçu en provenance du serveur. Si le document XML est bien reçu, le paramètre `success` est `true`. Si le document n'a pas été reçu, ou si une erreur est survenue au cours de la réception de la réponse provenant du serveur, le paramètre `success` renvoie `false`. Par défaut, l'implémentation de cette méthode n'est pas active. Pour annuler l'implémentation par défaut, vous devez attribuer une fonction qui contient des actions personnalisées.

**Disponibilité :** ActionScript 1.0 ; Flash Player 5

### Paramètres

*`success`*`:Boolean` - Valeur booléenne renvoyant `true` si l'objet XML a bien été chargé avec une opération `XML.load()` ou `XML.sendAndLoad()`; sinon elle renvoie `false`.

### Exemple

L'exemple suivant inclut du code ActionScript pour une application simple d'e-commerce. La méthode `sendAndLoad()` transmet un élément XML qui contient le nom d'utilisateur et un mot de passe et recourt au gestionnaire `XML.onLoad` pour traiter la réponse du serveur.

```
var login_str:String = "<login username=\""+username_txt.text+"\"
  password=\""+password_txt.text+"\" />";
var my_xml:XML = new XML(login_str);
var myLoginReply_xml:XML = new XML();

myLoginReply_xml.ignoreWhite = true;

myLoginReply_xml.onLoad = function(success:Boolean){

  if (success) {

    if ((myLoginReply_xml.firstChild.nodeName == "packet") &&
      (myLoginReply_xml.firstChild.attributes.success == "true")) {
      gotoAndStop("loggedIn");
    } else {
      gotoAndStop("loginFailed");
    }

  } else {
```

```
    gotoAndStop("connectionFailed");
  }

};

my_xml.sendAndLoad("http://www.flash-mx.com/mm/login_xml.cfm",
  myLoginReply_xml);
```

### Voir également

load (XML.load, méhode), sendAndLoad (XML.sendAndLoad, méthode),

## parseXML (XML.parseXML, méthode)

```
public parseXML(value:String) : Void
```

Analyse le texte XML spécifié dans le paramètre `value` et renseigne l'objet XML spécifié avec l'arborescence XML obtenue. Toutes les arborescences existantes dans l'objet XML sont supprimées.

**Disponibilité :** ActionScript 1.0 ; Flash Player 5

### Paramètres

*value*:String - Chaîne qui représente le texte XML à analyser et transmettre à l'objet XML spécifié.

### Exemple

L'exemple suivant crée et analyse un paquet XML :

```
var xml_str:String = "<state name=\"California\">
<city>San Francisco</city></state>"

// defining the XML source within the XML constructor:
var my1_xml:XML = new XML(xml_str);
trace(my1_xml.firstChild.attributes.name); // output: California

// defining the XML source using the XML.parseXML method:
var my2_xml:XML = new XML();
my2_xml.parseXML(xml_str);
trace(my2_xml.firstChild.attributes.name); // output: California
```

# send (XML.send, méthode)

```
public send(url:String, [target:String], [method:String]) : Boolean
```

Code l'objet XML spécifié dans un document XML et l'envoie à l'URL `target` spécifié.

Lorsque vous employez cette méthode, considérez le modèle de sécurité Flash Player :

- Pour Flash Player 8, elle n'est pas autorisée si le fichier SWF appelant est un sandbox local qui n'est pas entièrement fiable.
- Pour Flash Player 7 et supérieur, elle n'est pas autorisée si le fichier SWF appelant est local.

Pour plus d'informations, consultez :

- Chapitre 17, « Understanding Security » dans *Learning ActionScript 2.0 in Flash*
- Le livre blanc Flash Player 8 Security disponible à l'adresse : http://www.macromedia.com/go/fp8_security
- Le livre blanc Flash Player 8 Security-Related APIs disponible à l'adresse : http://www.macromedia.com/go/fp8_security_apis

**Disponibilité :** ActionScript 1.0 ; Flash Player 5

## Paramètres

*`url`*`:String` - URL de destination de l'objet XML spécifié.

*`target`*`:String` [en option] - Fenêtre de navigateur devant afficher les données renvoyées par le serveur :

- `_self` spécifie le cadre actif de la fenêtre en cours d'utilisation.
- `_blank` crée une fenêtre.
- `_parent` appelle le parent du cadre actif.
- `_top` sélectionne le cadre de plus haut niveau de la fenêtre active.

Si vous ne spécifiez pas de paramètre `target`, le système applique `_self`.

*`method`*`:String` [en option] - Méthode du protocole HTTP employé : `"GET"` ou `"POST"`. Dans un navigateur, la valeur par défaut est `"POST"`. Dans un environnement Flash de test, la valeur par défaut est `"GET"`.

## Valeur renvoyée

`Boolean` - `false` si aucun paramètre n'est précisé, sinon `true`.

### Exemple

L'exemple suivant définit un paquet XML et le type de contenu de l'objet XML. Les données sont alors envoyées à un serveur et les résultats s'affichent dans une fenêtre de navigateur.

```
var my_xml:XML = new XML("<highscore><name>Ernie</name><score>13045</
  score></highscore>");
my_xml.contentType = "text/xml";
my_xml.send("http://www.flash-mx.com/mm/highscore.cfm", "_blank");
```

Appuyez sur F12 pour tester cet exemple dans un navigateur.

### Voir également

`sendAndLoad (XML.sendAndLoad, méthode)`

## sendAndLoad (XML.sendAndLoad, méthode)

```
public sendAndLoad(url:String, resultXML:XML) : Void
```

Code l'objet XML spécifié en un document XML, l'envoie à l'URL spécifiée à l'aide de la méthode `POST`, télécharge la réponse du serveur et la charge dans le `resultXMLobject` spécifié dans les paramètres. La réponse du serveur se charge de la même manière que celle utilisée dans la méthode `XML.load()`.

Lorsque la méthode `sendAndLoad()` est exécutée, la propriété d'objet XML `loaded`est définie sur `false`. Lorsque le téléchargement des données XML se termine, la propriété `loaded` est définie sur `true` si les données ont bien été chargées et le gestionnaire d'événements `onLoad` est appelé. Les données XML ne sont pas analysées avant la fin du téléchargement. Si l'objet XML contenait précédemment des arborescences XML, elles sont supprimées.

Lorsque vous employez cette méthode, considérez le modèle de sécurité Flash Player :

Pour Flash Player 8 :

- Le chargement de données n'est pas autorisé si le fichier SWF appelant est dans le sandbox local-with-file-system et que la ressource cible provient d'un sandbox de réseau.
- Le chargement de données n'est pas autorisé si le fichier SWF appelant provient d'un sandbox de réseau et que la ressource cible est locale.

Pour plus d'informations, consultez :

- Chapitre 17, « Understanding Security » dans *Learning ActionScript 2.0 in Flash*
- Le livre blanc Flash Player 8 Security disponible à l'adresse : http://www.macromedia.com/go/fp8_security
- Le livre blanc Flash Player 8 Security-Related APIs disponible à l'adresse http://www.macromedia.com/go/fp8_security_apis

Pour Flash Player 7 et postérieur, les sites Web permettent l'accès inter-domaines à des ressources via un fichier de régulation inter-domaines. Dans les fichiers SWF d'une version exécutée dans Flash Player 7 ou une version ultérieure, le paramètre `url` doit se trouver exactement dans le même domaine. Par exemple, un fichier SWF à l'adresse www.someDomain.com peut charger uniquement des données provenant de sources situées aussi à l'adresse store.someDomain.com.

Dans les fichiers SWF exécutés dans une version du lecteur antérieure à Flash Player 7, le paramètre `url` doit être dans le même superdomaine que le fichier SWF qui transmet cet appel. Un *superdomaine* est dérivé en supprimant le composant le plus à gauche de l'URL d'un fichier. Par exemple, un fichier SWF à l'adresse www.someDomain.com peut charger des données provenant de sources à l'adresse store.someDomain.com, étant donné que les deux fichiers sont dans le même superdomaine de someDomain.com.

**Disponibilité :** ActionScript 1.0 ; Flash Player 5

## Paramètres

*`url`*`:String` - Chaîne ; URL de destination de l'objet XML spécifié. Si le fichier SWF effectuant cet appel s'exécute dans un navigateur Web, `url` doit appartenir au même domaine que le fichier SWF. Pour plus de détails, reportez-vous à la section Description.

*`resultXML`*`:XML` - Objet XML cible créé avec la méthode constructeur XML qui reçoit les informations renvoyées par le serveur.

## Exemple

L'exemple suivant inclut du code ActionScript pour une application simple d'e-commerce. La méthode `XML.sendAndLoad()` transmet un élément XML qui contient le nom d'utilisateur et un mot de passe et recourt au gestionnaire `onLoad` pour traiter la réponse du serveur.

```
var login_str:String = "<login username=\""+username_txt.text+"\"
  password=\""+password_txt.text+"\" />";
var my_xml:XML = new XML(login_str);
var myLoginReply_xml:XML = new XML();
myLoginReply_xml.ignoreWhite = true;
myLoginReply_xml.onLoad = myOnLoad;
my_xml.sendAndLoad("http://www.flash-mx.com/mm/login_xml.cfm",
  myLoginReply_xml);
function myOnLoad(success:Boolean) {
  if (success) {
    if ((myLoginReply_xml.firstChild.nodeName == "packet") &&
      (myLoginReply_xml.firstChild.attributes.success == "true")) {
      gotoAndStop("loggedIn");
    } else {
      gotoAndStop("loginFailed");
    }
```

```
  } else {
    gotoAndStop("connectionFailed");
  }
}
```

### Voir également

`send (XML.send, méthode)`, `load (XML.load, méhode)`, `loaded (XML.loaded, propriété)`, `onLoad (XML.onLoad, gestionnaire)`

## status (XML.status, propriété)

```
statut public : Nombre
```

Définit automatiquement et renvoie une valeur numérique qui indique si un document XML a été correctement analysé dans un objet XML. Les codes d'états numériques sont indiqués ci-dessous, avec des descriptions :

- 0 Pas d'erreur : l'analyse est terminée.
- -2 Une section CDATA ne s'est pas correctement terminée.
- -3 La déclaration `XML` ne s'est pas terminée correctement.
- -4 La déclaration `DOCTYPE` ne s'est pas terminée correctement.
- -5 Un commentaire ne s'est pas correctement terminé.
- -6 Un élément XML a été malformulé.
- -7 Mémoire insuffisante.
- -8 Une valeur d'attribut ne s'est pas terminée correctement.
- -9 Une balise de début ne correspond pas à une balise de fin.
- -10 Une balise de début n'a pas de balise de fin correspondante.

**Disponibilité :** ActionScript 1.0 ; Flash Player 5

### Exemple

L'exemple suivant charge un paquet XML dans un fichier SWF. Un message d'état s'affiche pour indiquer si le code XML se charge et est analysé correctement. Ajoutez le code ActionScript suivant à votre fichier FLA ou AS :

```
var my_xml:XML = new XML();
my_xml.onLoad = function(success:Boolean) {
  if (success) {
  if (my_xml.status == 0) {
    trace("XML was loaded and parsed successfully");
  } else {
    trace("XML was loaded successfully, but was unable to be parsed.");
  }
```

```
  var errorMessage:String;
  switch (my_xml.status) {
  case 0 :
    errorMessage = "No error; parse was completed successfully.";
    break;
  case -2 :
    errorMessage = "A CDATA section was not properly terminated.";
    break;
  case -3 :
    errorMessage = "The XML declaration was not properly terminated.";
    break;
  case -4 :
    errorMessage = "The DOCTYPE declaration was not properly terminated.";
    break;
  case -5 :
    errorMessage = "A comment was not properly terminated.";
    break;
  case -6 :
    errorMessage = "An XML element was malformed.";
    break;
  case -7 :
    errorMessage = "Out of memory.";
    break;
  case -8 :
    errorMessage = "An attribute value was not properly terminated.";
    break;
  case -9 :
    errorMessage = "A start-tag was not matched with an end-tag.";
    break;
  case -10 :
    errorMessage = "An end-tag was encountered without a matching
    start-tag.";
    break;
  default :
    errorMessage = "An unknown error has occurred.";
    break;
  }
  trace("status: "+my_xml.status+" ("+errorMessage+")");
  } else {
  trace("Unable to load/parse XML. (status: "+my_xml.status+")");
  }
};
my_xml.load("http://www.helpexamples.com/flash/badxml.xml");
```

# XML, constructeur

```
public XML(text:String)
```

Crée un nouvel objet XML. Vous devez utiliser le constructeur pour créer un objet XML avant d'appeler une des méthodes de la classe XML.

*Remarque :* Utilisez les méthodes `createElement()` et `createTextNode()` pour ajouter des éléments et des nœuds de texte à une arborescence de document XML.

**Disponibilité :** ActionScript 1.0 ; Flash Player 5

## Paramètres

*`text`*`:String` - Chaîne ; texte XML analysé pour créer l'objet XML.

## Exemple

L'exemple suivant crée un objet XML vide :

```
var my_xml:XML = new XML();
```

L'exemple suivant crée un objet XML en analysant le texte XML spécifié par le paramètre `source` et remplit le nouvel objet XML avec l'arborescence de documents XML qui en résulte :

```
var other_xml:XML = new XML("<state name=\"California\"><city>San
  Francisco</city></state>");
```

## Voir également

`createElement (méthode nodeML.createElement)`, `createTextNode (méthode XML.createTextNode)`

# xmlDecl (XML.xmlDecl, propriété)

```
public xmlDecl : String
```

Chaîne qui spécifie des informations sur une déclaration XML du document. Après l'analyse du document XML dans un objet XML, cette propriété est définie sur le texte de la déclaration XML de document. Cette propriété est définie à l'aide d'une représentation sous forme de chaîne de la déclaration XML, pas d'un objet de nœud XML. En cas d'absence de déclaration XML au cours d'une opération d'analyse, la propriété est définie sur `undefined`. La méthode `XML.toString()` fournit les contenus de la propriété `XML.xmlDecl` avant tout autre texte de l'objet XML. Si la propriété `XML.xmlDecl` contient le type `undefined`, aucune déclaration XML n'est produite.

**Disponibilité :** ActionScript 1.0 ; Flash Player 5

## Exemple

L'exemple suivant crée un champ texte, `my_txt`, qui reprend les dimensions de la scène. Ce champ texte affiche les propriétés du paquet XML qui se charge dans le fichier SWF. La déclaration docType s'affiche dans `my_txt`. Ajoutez le code ActionScript suivant à votre fichier FLA ou AS :

```
var my_fmt:TextFormat = new TextFormat();
my_fmt.font = "_typewriter";
my_fmt.size = 12;
my_fmt.leftMargin = 10;

this.createTextField("my_txt", this.getNextHighestDepth(), 0, 0,
  Stage.width, Stage.height);
my_txt.border = true;
my_txt.multiline = true;
my_txt.wordWrap = true;
my_txt.setNewTextFormat(my_fmt);

var my_xml:XML = new XML();
my_xml.ignoreWhite = true;
my_xml.onLoad = function(success:Boolean) {
  var endTime:Number = getTimer();
  var elapsedTime:Number = endTime-startTime;
  if (success) {
    my_txt.text = "xmlDecl:"+newline+my_xml.xmlDecl+newline+newline;
    my_txt.text +=
  "contentType:"+newline+my_xml.contentType+newline+newline;
    my_txt.text +=
  "docTypeDecl:"+newline+my_xml.docTypeDecl+newline+newline;
    my_txt.text += "packet:"+newline+my_xml.toString()+newline+newline;
  } else {
    my_txt.text = "Unable to load remote XML."+newline+newline;
  }
  my_txt.text += "loaded in: "+elapsedTime+" ms.";
};
my_xml.load("http://www.helpexamples.com/crossdomain.xml");
var startTime:Number = getTimer();
```

La méthode `MovieClip.getNextHighestDepth()` employée dans cet exemple requiert Flash Player 7 ou ultérieur. Si votre fichier SWF contient un composant de version 2, employez les composants de version 2 de classe DepthManager plutôt que la méthode `MovieClip.getNextHighestDepth()`.

## Voir également

`docTypeDecl (XML.docTypeDecl, propriété)`

# XMLNode

```
Object
  |
  +-XMLNode
```

```
public class XMLNode
extends Object
```

Un document XML est représenté dans Flash par la classe XML. Chaque élément du document hiérarchique est représenté par un objet XMLNode.

**Disponibilité :** ActionScript 1.0 ; Flash Player 5

## Voir également

`XML`

## Résumé des propriétés

| Modificateurs | Propriété | Description |
|---|---|---|
| | `attributes:Object` | Objet contenant tous les attributs de l'instance XML spécifiée. |
| | `childNodes:Array` [lecture seule] | Tableau des enfants de l'objet XML spécifié. |
| | `firstChild:XMLNode` [lecture seule] | Evalue l'objet XML spécifié et fait référence au premier enfant dans la liste des enfants de nœud parent. |
| | `lastChild:XMLNode` [lecture seule] | Valeur XMLNode qui fait référence au dernier enfant de la liste des enfants du nœud. |
| | `localName:String` [lecture seule] | La partie locale du nom de nœud XML. |
| | `namespaceURI:String` [lecture seule] | Si le nœud XML a un préfixe, `namespaceURI` est la valeur de la déclaration `xmlns` de ce préfixe(URI), appelé en général l'URI d'espace de nom. |
| | `nextSibling:XMLNode` [lecture seule] | Valeur XMLNode qui fait référence au frère suivant de la liste des enfants du nœud. |
| | `nodeName:String` | Chaîne représentant le nom du nœud de l'objet XML. |
| | `nodeType:Number` [lecture seule] | Valeur `nodeType`, 1 pour un élément XML ou 3 pour un nœud texte. |
| | `nodeValue:String` | Valeur du nœud de l'objet XML. |

| Modificateurs | Propriété | Description |
|---|---|---|
| | `parentNode:XMLNode` [lecture seule] | Valeur XMLNode qui fait référence au nœud parent de l'objet XML spécifié ou renvoie `null` si le nœud n'a pas de parent. |
| | `prefix:String` [lecture seule] | Le préfixe du nom de nœud XML. |
| | `previousSibling:XMLNode` [lecture seule] | Valeur XMLNode qui fait référence au frère précédent de la liste des enfants du nœud. |

*Propriétés héritées de la classe Object*

```
constructeur (propriété Object.constructor), __proto__ (Object.__proto__
property), prototype (Object.prototype, propriété), __resolve
(Object.__resolve, propriété)
```

## Récapitulatif des constructeurs

| Signature | Description |
|---|---|
| `XMLNode(type:Number, value:String)` | Le constructeur XMLNode permet de créer une instance d'un nœud XML en fonction d'une chaîne indiquant son contenu et d'un nombre représentant son type de nœud. |

## Résumé de la méthode

| Modificateurs | Signature | Description |
|---|---|---|
| | `appendChild(newChild :XMLNode) : Void` | Ajoute le nœud spécifié à la liste des enfants de l'objet XML. |
| | `cloneNode(deep:Boolean) : XMLNode` | Construit et renvoie un nouveau nœud XML des mêmes type, nom, valeur et attributs que l'objet XML spécifié. |
| | `getNamespaceForPrefix(prefix:String) : String` | Renvoie l'URI d'espace de nom qui est associée au préfixe spécifié pour le nœud. |
| | `getPrefixForNamespace(nsURI:String) : String` | Renvoie le préfixe qui est associé à l'URI d'espace de nom spécifiée pour le nœud. |
| | `hasChildNodes() : Boolean` | Indique si l'objet XML comporte ou non des nœuds enfant. |

| Modificateurs | Signature | Description |
|---|---|---|
| | `insertBefore(newChild:XMLNode, insertPoint:XMLNode) : Void` | Insère un nœud `newChild` dans la liste d'enfants du code XML, avant le nœud `insertPoint`. |
| | `removeNode() : Void` | Supprime l'objet XML spécifié de son parent. |
| | `toString() : String` | Evalue l'objet XML spécifié, crée une représentation graphique de la structure XML, comprenant le nœud, les enfants et les attributs et renvoie le résultat sous forme de chaîne. |

*Méthodes héritées de la classe Object*

`addProperty (méthode Object.addProperty), hasOwnProperty (méthode Object.hasOwnProperty), isPropertyEnumerable (méthode Object.isPropertyEnumerable), isPrototypeOf (méthode Object.isPrototypeOf), registerClass (méthode Object.registerClass), toString (méthode Object.toString), unwatch (méthode Object.unwatch), valueOf (méthode Object.valueOf), watch (méthode Object.watch)`

# appendChild (XMLNode.appendChild, méthode)

`public appendChild(newChild:XMLNode) : Void`

Ajoute le nœud spécifié à la liste des enfants de l'objet XML. Cette méthode agit directement sur le nœud référencé par le paramètre `childNode` ; elle n'ajoute pas une copie du nœud. Si le nœud à ajouter existe déjà dans une autre arborescence, l'ajout du nœud au nouvel emplacement le supprimera de son emplacement actuel. Si le paramètre `childNode` fait référence à un nœud qui existe déjà dans une autre arborescence XML, le nœud enfant ajouté est placé dans la nouvelle structure après sa suppression de son nœud parent existant.

**Disponibilité :** ActionScript 1.0 ; Flash Player 5

## Paramètres

*`newChild`*`:XMLNode` - Valeur XMLNode qui représente le nœud à déplacer de son emplacement actuel vers la liste d'enfants de *`my_xml`*.

## Exemple

Cet exemple effectue les choses suivantes dans l'ordre indiqué :

- Crée deux documents XML vides, `doc1` et `doc2`.

- Crée un nœud avec la méthode `createElement()` et l'ajoute, avec la méthode `appendChild()`, au document XML appelé `doc1`.
- Indique comment déplacer un nœud avec la méthode `appendChild()`, en déplaçant le nœud racine de `doc1` vers `doc2`.
- Clone le nœud racine de `doc2` et l'ajoute à `doc1`.
- Crée un nœud et l'ajoute au nœud racine du document XML, `doc1`.

```
var doc1:XML = new XML();
var doc2:XML = new XML();

// create a root node and add it to doc1
var rootnode:XMLNode = doc1.createElement("root");
doc1.appendChild(rootnode);
trace ("doc1: " + doc1); // output: doc1: <root />
trace ("doc2: " + doc2); // output: doc2:

// move the root node to doc2
doc2.appendChild(rootnode);
trace ("doc1: " + doc1); // output: doc1:
trace ("doc2: " + doc2); // output: doc2: <root />

// clone the root node and append it to doc1
var clone:XMLNode = doc2.firstChild.cloneNode(true);
doc1.appendChild(clone);
trace ("doc1: " + doc1); // output: doc1: <root />
trace ("doc2: " + doc2); // output: doc2: <root />

// create a new node to append to root node (named clone) of doc1
var newNode:XMLNode = doc1.createElement("newbie");
clone.appendChild(newNode);
trace ("doc1: " + doc1); // output: doc1: <root><newbie /></root>
```

## attributes (XMLNode.attributes, propriété)

```
public attributes : Object
```

Objet contenant tous les attributs de l'instance XML spécifiée. L'objet XML.attributes contient une variable pour chaque attribut de l'instance XML. Ces variables étant définies comme faisant partie de l'objet, elles sont généralement appelées propriétés de l'objet. La valeur de chaque attribut est enregistrée dans la propriété correspondante comme une chaîne. Par exemple, si vous avez un attribut appelé color, vous récupérerez la valeur de l'attribut en spécifiant couleur comme nom de la propriété, comme indiqué par le code suivant :

```
var myColor:String = doc.firstChild.attributes.color;
```

**Disponibilité :** ActionScript 1.0 ; Flash Player 5

### Exemple

L'exemple ci-dessous montre comment lire et écrire les attributs du nœud XML :

```
var doc:XML = new XML("<mytag name='Val'> item </mytag>");
trace(doc.firstChild.attributes.name); // Val

doc.firstChild.attributes.order = "first";
trace (doc.firstChild); // <mytag order="first" name="Val"> item </mytag>

for (attr in doc.firstChild.attributes) {
  trace (attr + " = " + doc.firstChild.attributes[attr]);
}

// order = first
// name = Val
```

## childNodes (XMLNode.childNodes, propriété)

```
public childNodes : Tableau [lecture seule]
```

Tableau des enfants de l'objet XML spécifié. Chaque élément du tableau est une référence vers un objet XML qui représente un nœud enfant. Cette propriété est en lecture seule et ne peut pas être utilisée pour manipuler les nœuds enfants. Utilisez les méthodes `appendChild()`, `insertBefore()`, et `removeNode()` pour manipuler les nœuds enfants.

Cette propriété n'est pas définie pour les nœuds de texte (`nodeType == 3`).

**Disponibilité :** ActionScript 1.0 ; Flash Player 5

### Exemple

L'exemple suivant indique comment utiliser la propriété XML.childNodes pour renvoyer un tableau récapitulatif des nœuds enfants :

```
// create a new XML document
var doc:XML = new XML();

// create a root node
var rootNode:XMLNode = doc.createElement("rootNode");

// create three child nodes
var oldest:XMLNode = doc.createElement("oldest");
var middle:XMLNode = doc.createElement("middle");
var youngest:XMLNode = doc.createElement("youngest");

// add the rootNode as the root of the XML document tree
doc.appendChild(rootNode);

// add each of the child nodes as children of rootNode
rootNode.appendChild(oldest);
```

```
rootNode.appendChild(middle);
rootNode.appendChild(youngest);

// create an array and use rootNode to populate it
var firstArray:Array = doc.childNodes;
trace (firstArray);
// output: <rootNode><oldest /><middle /><youngest /></rootNode>

// create another array and use the child nodes to populate it
var secondArray:Array = rootNode.childNodes;
trace(secondArray);
// output: <oldest />,<middle />,<youngest />
```

### Voir également

nodeType (XMLNode.nodeType, propriété), appendChild (XMLNode.appendChild, méthode), insertBefore (XMLNode.insertBefore, méthode), removeNode (XMLNode.removeNode, méthode)

## cloneNode (XMLNode.cloneNode, méthode)

```
public cloneNode(deep:Boolean) : XMLNode
```

Construit et renvoie un nouveau nœud XML des mêmes type, nom, valeur et attributs que l'objet XML spécifié. Si `deep` est défini sur `true`, tous les nœuds enfants sont clonés de manière récursive, ce qui crée une copie exacte de l'arborescence du document de l'objet original.

Le clone du nœud qui est renvoyé n'est plus associé à l'arborescence de l'élément cloné. Par conséquent, `nextSibling`, `parentNode` et `previousSibling` ont tous une valeur `null`. Si le paramètre `deep` a la valeur `false`, ou si le nœud *`my_xml`* n'a pas de nœuds enfants, `firstChild` et `lastChild` sont aussi null.

**Disponibilité :** ActionScript 1.0 ; Flash Player 5

### Paramètres

*`deep`*`:Boolean` - Valeur booléenne ; si elle est définie sur `true`, les enfants de l'objet XML spécifié sont clonés de façon récursive.

### Valeur renvoyée

`XMLNode` - Objet XMLNode.

## Exemple

L'exemple suivant illustre comment utiliser la méthode `XML.cloneNode()` pour créer la copie d'un nœud :

```
// create a new XML document
var doc:XML = new XML();

// create a root node
var rootNode:XMLNode = doc.createElement("rootNode");

// create three child nodes
var oldest:XMLNode = doc.createElement("oldest");
var middle:XMLNode = doc.createElement("middle");
var youngest:XMLNode = doc.createElement("youngest");

// add the rootNode as the root of the XML document tree
doc.appendChild(rootNode);

// add each of the child nodes as children of rootNode
rootNode.appendChild(oldest);
rootNode.appendChild(middle);
rootNode.appendChild(youngest);

// create a copy of the middle node using cloneNode()
var middle2:XMLNode = middle.cloneNode(false);

// insert the clone node into rootNode between the middle and youngest nodes
rootNode.insertBefore(middle2, youngest);
trace(rootNode);
// output (with line breaks added):
// <rootNode>
// <oldest />
// <middle />
// <middle />
// <youngest />
// </rootNode>

// create a copy of rootNode using cloneNode() to demonstrate a deep copy
var rootClone:XMLNode = rootNode.cloneNode(true);

// insert the clone, which contains all child nodes, to rootNode
rootNode.appendChild(rootClone);
trace(rootNode);
// output (with line breaks added):
// <rootNode>
// <oldest />
// <middle />
// <middle />
// <youngest />
// <rootNode>
```

```
// <oldest />
// <middle />
// <middle />
// <youngest />
// </rootNode>
// </rootNode>
```

# firstChild (XMLNode.firstChild, propriété)

```
public firstChild : XMLNode [lecture seule]
```

Evalue l'objet XML spécifié et fait référence au premier enfant dans la liste des enfants de nœud parent. Cette propriété est `null` si le nœud n'a pas d'enfants. Cette propriété est `undefined` si le nœud est un nœud de texte. Il s'agit d'une propriété en lecture seule qui ne peut pas être utilisée pour manipuler les nœuds enfants ; utilisez les méthodes `appendChild()`, `insertBefore()` et `removeNode()` pour manipuler les nœuds enfants.

**Disponibilité :** ActionScript 1.0 ; Flash Player 5

## Exemple

L'exemple suivant indique comment utiliser `XML.firstChild` pour parcourir les enfants d'un nœud :

```
// create a new XML document
var doc:XML = new XML();

// create a root node
var rootNode:XMLNode = doc.createElement("rootNode");

// create three child nodes
var oldest:XMLNode = doc.createElement("oldest");
var middle:XMLNode = doc.createElement("middle");
var youngest:XMLNode = doc.createElement("youngest");

// add the rootNode as the root of the XML document tree
doc.appendChild(rootNode);

// add each of the child nodes as children of rootNode
rootNode.appendChild(oldest);
rootNode.appendChild(middle);
rootNode.appendChild(youngest);

// use firstChild to iterate through the child nodes of rootNode
for (var aNode:XMLNode = rootNode.firstChild; aNode != null; aNode =
  aNode.nextSibling) {
  trace(aNode);
}
```

```
// output:
// <oldest />
// <middle />
// <youngest />
```

L'exemple suivant provient du fichier FLA XML_languagePicker FLA situé dans le répertoire Examples. Vous pouvez le trouver dans la définition de fonction de gestionnaire d'événements, languageXML.onLoad :

```
// loop through the strings in each language node
// adding each string as a new element in the language array
for (var stringNode:XMLNode = childNode.firstChild; stringNode != null;
  stringNode = stringNode.nextSibling, j++) {
  masterArray[i][j] = stringNode.firstChild.nodeValue;
}
```

Pour afficher l'ensemble du script, ouvrez le fichier XML_languagePicker.fla situé dans le dossier ActionScript :

- Windows : *lecteur d'amorçage*\Program Files\Macromedia\Flash 8\Samples and Tutorials\Samples\ActionScript
- Macintosh : *disque dur Macintosh*/Applications/Macromedia Flash 8/Samples and Tutorials/Samples/ActionScript

### Voir également

`appendChild (XMLNode.appendChild, méthode)`, `insertBefore (XMLNode.insertBefore, méthode)`, `removeNode (XMLNode.removeNode, méthode)`

## getNamespaceForPrefix (XMLNode.getNamespaceForPrefix, méthode)

```
public getNamespaceForPrefix(prefix:String) : String
```

Renvoie l'URI d'espace de nom qui est associée au préfixe spécifié pour le nœud. Pour fixer l'URI, `getPrefixForNamespace()` remonte la hiérarchie XML depuis le nœud, si nécessaire, et renvoie l'URI de l'espace de nom de la première déclaration `xmlns` du `prefix` donné.

Si aucun espace de nom n'est défini pour ledit préfixe, la méthode renvoie `null`.

Si vous spécifiez une chaîne vide (`""`) en tant que `prefix` et que ce nœud a un espace de nom par défaut (par exemple `xmlns="http://www.example.com/"`), la méthode renvoie cet URI d'espace de nom.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

### Paramètres

*`prefix`*`:String` - Préfixe pour lequel la méthode renvoie le nom d'espace associé.

### Valeur renvoyée

`String` - Nom d'espace associé au préfixe spécifié.

### Exemple

L'exemple suivant crée un objet XML très simple et renvoie le résultat de l'appel à `getNamespaceForPrefix()`

```
function createXML():XMLNode {
  var str:String = "<Outer xmlns:exu=\"http://www.example.com/util\">"
    + "<exu:Child id='1' />"
    + "<exu:Child id='2' />"
    + "<exu:Child id='3' />"
    + "</Outer>";
  return new XML(str).firstChild;
}

var xml:XMLNode = createXML();
trace(xml.getNamespaceForPrefix("exu")); // output: http://www.example.com/
  util
trace(xml.getNamespaceForPrefix("")); // output: null
```

### Voir également

`getPrefixForNamespace (XMLNode.getPrefixForNamespace, méthode)`, `namespaceURI (XMLNode.namespaceURI, propriété)`

## getPrefixForNamespace (XMLNode.getPrefixForNamespace, méthode)

```
public getPrefixForNamespace(nsURI:String) : String
```

Renvoie le préfixe qui est associé à l'URI d'espace de nom spécifiée pour le nœud. Pour fixer le préfix, `getPrefixForNamespace()` remonte la hiérarchie XML depuis le nœud, si nécessaire, et renvoie le préfixe de la première déclaration `xmlns` avec un URI d'espace de nom correspondant à `nsURI`.

Si aucune affectation `xmlns` de l'URI considéré n'existe, la méthode renvoie `null`. S'il existe une affectation`xmlns` de l'URI considéré mais qu'aucun préfixe ne lui est associé, la méthode renvoie une chaîne vide (`""`).

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

### Paramètres

*`nsURI`*`:String` - URI d'espace de nom pour laquelle la méthode renvoie le préfixe associé.

## Valeur renvoyée

`String` - Préfixe associé au nom d'espace spécifié.

## Exemple

L'exemple suivant crée un objet XML très simple et renvoie le résultat de l'appel à la méthode `getPrefixForNamespace()`. Le nœud XML `Outer`, représenté par la variable `xmlDoc`, définit un URI d'espace de nom et l'affecte au préfixe `exu`. Le fait d'appeler la méthode `getPrefixForNamespace()` avec l'URI de l'espace de nom défini (« http://www.example.com/util ») renvoie le préfixe `exu`, mais l'appel de cette méthode avec un URI non défini ("http://www.example.com/other") renvoie `null`. Le premier nœud`exu:Child`, représenté par la variable `child1`, définit aussi un URI d'espace de nom (« http://www.example.com/child »), mais ne l'affecte pas à un préfixe. Le fait d'appeler cette méthode sur un URI d'espace de nom défini mais non affecté, renvoie une chaîne vide.

```
function createXML():XMLNode {
  var str:String = "<Outer xmlns:exu=\"http://www.example.com/util\">"
    + "<exu:Child id='1' xmlns=\"http://www.example.com/child\"/>"
    + "<exu:Child id='2' />"
    + "<exu:Child id='3' />"
    + "</Outer>";
  return new XML(str).firstChild;
}

var xmlDoc:XMLNode = createXML();
trace(xmlDoc.getPrefixForNamespace("http://www.example.com/util")); //
  output: exu
trace(xmlDoc.getPrefixForNamespace("http://www.example.com/other")); //
  output: null

var child1:XMLNode = xmlDoc.firstChild;
trace(child1.getPrefixForNamespace("http://www.example.com/child")); //
  output: [empty string]
trace(child1.getPrefixForNamespace("http://www.example.com/other")); //
  output: null
```

## Voir également

`getNamespaceForPrefix (XMLNode.getNamespaceForPrefix, méthode)`, `namespaceURI (XMLNode.namespaceURI, propriété)`

# hasChildNodes (XMLNode.hasChildNodes, méthode)

```
public hasChildNodes() : Boolean
```

Indique si l'objet XML comporte ou non des nœuds enfant.

**Disponibilité :** ActionScript 1.0 ; Flash Player 5

## Valeur renvoyée

`Boolean` - `true` si le nœud XMLNode comporte au moins un enfant ; `false` dans tous les autres cas.

## Exemple

L'exemple suivant crée un paquet XML : Si le nœud racine comporte des enfants, le code s'exécute en boucle au niveau de chaque enfant pour afficher le nom et la valeur de ce nœud. Ajoutez le code ActionScript suivant à votre fichier FLA ou AS :

```
var my_xml:XML = new XML("hankrudolph");
if (my_xml.firstChild.hasChildNodes()) {
// use firstChild to iterate through the child nodes of rootNode
  for (var aNode:XMLNode = my_xml.firstChild.firstChild; aNode != null;
  aNode=aNode.nextSibling) {
    if (aNode.nodeType == 1) {
      trace(aNode.nodeName+":\t"+aNode.firstChild.nodeValue);
    }
  }
}
```

Les informations suivantes apparaissent dans le panneau Sortie :

```
output:
username: hank
password: rudolph
```

# insertBefore (XMLNode.insertBefore, méthode)

```
public insertBefore(newChild:XMLNode, insertPoint:XMLNode) : Void
```

Insère un nœud `newChild` dans la liste d'enfants du code XML, avant le nœud `insertPoint`. Si `insertPoint` n'est *pas* un enfant de l'objet XMLNode, l'insertion échoue.

**Disponibilité :** ActionScript 1.0 ; Flash Player 5

## Paramètres

*`newChild`*`:XMLNode` - Objet XMLNode à insérer.

*insertPoint*:XMLNode - Objet XMLNode qui suit le nœud newChild après l'appel de la méthode.

### Exemple

Le code suivant insère un nouveau nœud XML entre deux nœuds existants :

```
var my_xml:XML = new XML("<a>1</a>\n<c>3</c>");
var insertPoint:XMLNode = my_xml.lastChild;
var newNode:XML = new XML("<b>2</b>\n");
my_xml.insertBefore(newNode, insertPoint);
trace(my_xml);
```

### Voir également

XML, cloneNode (XMLNode.cloneNode, méthode)

## lastChild (XMLNode.lastChild, propriété)

```
public lastChild : XMLNode [lecture seule]
```

Valeur XMLNode qui fait référence au dernier enfant de la liste des enfants du nœud. Cette propriété XML.lastChild est null si le nœud n'a pas d'enfants. Cette propriété ne peut pas être utilisée pour manipuler les nœuds enfants ; utilisez les méthodes appendChild(), insertBefore(), et removeNode() pour manipuler les nœuds enfants.

**Disponibilité :** ActionScript 1.0 ; Flash Player 5

### Exemple

L'exemple suivant utilise la propriété XML.lastChild pour parcourir les enfants d'un nœud XML, en partant du dernier élément de la liste d'enfants pour remonter jusqu'au premier :

```
// create a new XML document
var doc:XML = new XML();

// create a root node
var rootNode:XMLNode = doc.createElement("rootNode");

// create three child nodes
var oldest:XMLNode = doc.createElement("oldest");
var middle:XMLNode = doc.createElement("middle");
var youngest:XMLNode = doc.createElement("youngest");

// add the rootNode as the root of the XML document tree
doc.appendChild(rootNode);

// add each of the child nodes as children of rootNode
rootNode.appendChild(oldest);
rootNode.appendChild(middle);
```

```
rootNode.appendChild(youngest);

// use lastChild to iterate through the child nodes of rootNode
for (var aNode:XMLNode = rootNode.lastChild; aNode != null; aNode =
  aNode.previousSibling) {
  trace(aNode);
}

// output:
// <youngest />
// <middle />
// <oldest />
```

L'exemple suivant crée un nouveau paquet XML et utilise la propriété `XML.lastChild` pour parcourir les nœuds enfant du nœud racine :

```
// create a new XML document
var doc:XML = new XML("");

var rootNode:XMLNode = doc.firstChild;

// use lastChild to iterate through the child nodes of rootNode
for (var aNode:XMLNode = rootNode.lastChild; aNode != null;
  aNode=aNode.previousSibling) {
  trace(aNode);
}

// output:
// <youngest />
// <middle />
// <oldest />
```

### Voir également

`appendChild (XMLNode.appendChild, méthode)`, `insertBefore (XMLNode.insertBefore, méthode)`, `removeNode (XMLNode.removeNode, méthode)`, `XML`

## localName (XMLNode.localName, propriété)

```
public localName : String [lecture seule]
```

La partie locale du nom de nœud XML. Il s'agit du nom d'élément sans le préfixe d'espace de nom. Par exemple, le nœud `<contact:mailbox/>bob@example.com</contact:mailbox>` utilise le nom local « mailbox » et le préfixe « contact », ce qui forme « contact.mailbox ».

Vous pouvez accéder au préfixe par l'intermédiaire de la propriété `prefix` de l'objet nœud XML. La propriété `nodeName` renvoie le nom complet (ce qui inclut le préfixe et le nom local).

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

### Exemple

Cet exemple utilise un fichier SWF et un fichier XML situé dans le même répertoire. Le fichier XML, appelé « SoapSample.xml » contient le code suivant :

```
<?xml version="1.0"?>
<soap:Envelope xmlns:soap="http://www.w3.org/2001/12/soap-envelope">
<soap:Body xmlns:w="http://www.example.com/weather">
<w:GetTemperature>
<w:City>San Francisco</w:City>
</w:GetTemperature>
</soap:Body>
</soap:Envelope>
```

La source du fichier SWF contient le script suivant (remarquez les commentaires précisant les chaînes renvoyées) :

```
var xmlDoc:XML = new XML()
xmlDoc.ignoreWhite = true;
xmlDoc.load("SoapSample.xml")
xmlDoc.onLoad = function(success:Boolean)
{
  var tempNode:XMLNode = xmlDoc.childNodes[0].childNodes[0].childNodes[0];
  trace("w:GetTemperature localname: " + tempNode.localName); // Output:
  ... GetTemperature
  var soapEnvNode:XMLNode = xmlDoc.childNodes[0];
  trace("soap:Envelope localname: " + soapEnvNode.localName); // Output:
  ... Envelope
}
```

## namespaceURI (XMLNode.namespaceURI, propriété)

```
public namespaceURI : String [lecture seule]
```

Si le nœud XML a un préfixe, `namespaceURI` est la valeur de la déclaration `xmlns` de ce préfixe(URI), appelé en général l'URI d'espace de nom. La déclaration `xmlns` se trouve dans le nœud courant ou dans un nœud plus élevé dans la hiérarchie XML.

Si le nœud XML n'a pas de préfixe, la valeur de la propriété `namespaceURI` dépend de l'existence d'un espace de nom défini par défaut (par exemple `xmlns="http://www.example.com/"`). Si un espace de nom par défaut existe, la valeur de la propriété `namespaceURI` est celle de l'espace de nom par défaut. Si aucun espace de nom par défaut n'existe, la propriété `namespaceURI` de ce nœud est une chaîne vide (`""`).

Vous pouvez utiliser la méthode `getNamespaceForPrefix()` pour identifier le nom d'espace associé à un préfixe spécifique. La propriété `namespaceURI` renvoie le préfixe associé au nom de nœud.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

## Exemple

L'exemple suivant montre comment la propriété `namespaceURI` est affectée par l'emploi de préfixes. Un répertoire contient un fichier SWF et un fichier XML. Le fichier XML, appelé `SoapSample.xml` contient les balises suivantes :

```
<?xml version="1.0"?>
<soap:Envelope xmlns:soap="http://www.w3.org/2001/12/soap-envelope">
<soap:Body xmlns:w="http://www.example.com/weather">
<w:GetTemperature>
<w:City>San Francisco</w:City>
</w:GetTemperature>
</soap:Body>
</soap:Envelope>
```

La source du fichier SWF contient le script suivant (remarquez les commentaires précisant les chaînes renvoyées). Pour `tempNode`, représentant le nœud`w:GetTemperature`, la valeur de `namespaceURI` est définie dans la balise `soap:Body`. Pour `soapBodyNode`, représentant le nœud `soap:Body`, la valeur de `namespaceURI` est déterminée par la définition du préfixe `soap` du nœud supérieur, plutôt que la définition du préfixe `w`que le nœud `soap:Body` contient.

```
var xmlDoc:XML = new XML();
xmlDoc.load("SoapSample.xml");
xmlDoc.ignoreWhite = true;
xmlDoc.onLoad = function(success:Boolean)
{
  var tempNode:XMLNode = xmlDoc.childNodes[0].childNodes[0].childNodes[0];
  trace("w:GetTemperature namespaceURI: " + tempNode.namespaceURI);
    // Output: ... http://www.example.com/weather

  trace("w:GetTemperature soap namespace: " +
  tempNode.getNamespaceForPrefix("soap"));
    // Output: ... http://www.w3.org/2001/12/soap-envelope

  var soapBodyNode:XMLNode = xmlDoc.childNodes[0].childNodes[0];
  trace("soap:Envelope namespaceURI: " + soapBodyNode.namespaceURI);
    // Output: ... http://www.w3.org/2001/12/soap-envelope
}
```

L'exemple suivant emploie des balises XML sans préfixes. Il utilise un fichier SWF et un fichier XML situé dans le même répertoire. Le fichier XML, appelé `NoPrefix.xml` contient les balises suivantes :

```
<?xml version="1.0"?>
<rootnode>
<simplenode xmlns="http://www.w3.org/2001/12/soap-envelope">
<innernode />
```

```
</simplenode>
</rootnode>
```

La source du fichier SWF contient le script suivant (remarquez les commentaires précisant les chaînes renvoyées). Le `rootNode` n'a pas d'espace de nom par défaut, sa valeur `namespaceURI` est une chaîne vide. Le `simpleNode` définit un espace de nom par défaut, sa valeur `namespaceURI` est l'espace de nom par défaut. Le `innerNode` ne définit pas un espace de nom par défaut, il emploie l'espace de nom par défaut défini par `simpleNode`, sa valeur `namespaceURI` est donc la même que celle de `simpleNode`.

```
var xmlDoc:XML = new XML()
xmlDoc.load("NoPrefix.xml");
xmlDoc.ignoreWhite = true;
xmlDoc.onLoad = function(success:Boolean)
{
  var rootNode:XMLNode = xmlDoc.childNodes[0];
  trace("rootNode Node namespaceURI: " + rootNode.namespaceURI);
    // Output: [empty string]

  var simpleNode:XMLNode = xmlDoc.childNodes[0].childNodes[0];
  trace("simpleNode Node namespaceURI: " + simpleNode.namespaceURI);
    // Output: ... http://www.w3.org/2001/12/soap-envelope

  var innerNode:XMLNode = xmlDoc.childNodes[0].childNodes[0].childNodes[0];
  trace("innerNode Node namespaceURI: " + innerNode.namespaceURI);
    // Output: ... http://www.w3.org/2001/12/soap-envelope
}
```

### Voir également

`getNamespaceForPrefix (XMLNode.getNamespaceForPrefix, méthode)`,
`getPrefixForNamespace (XMLNode.getPrefixForNamespace, méthode)`

## nextSibling (XMLNode.nextSibling, propriété)

```
public nextSibling : XMLNode [lecture seule]
```

Valeur XMLNode qui fait référence au frère suivant de la liste des enfants du nœud. Cette propriété est `null` si le nœud n'a pas de nœud frère suivant. Cette propriété ne peut pas être utilisée pour manipuler les nœuds enfants ; utilisez les méthodes `appendChild()`, `insertBefore()`, et `removeNode()` pour manipuler les nœuds enfants.

**Disponibilité :** ActionScript 1.0 ; Flash Player 5

### Exemple

L'exemple suivant reprend une partie de l'exemple correspondant à la propriété `XML.firstChild` et indique comment utiliser la propriété `XML.nextSibling` pour parcourir les enfants d'un nœud XML :

```
for (var aNode:XMLNode = rootNode.firstChild; aNode != null; aNode =
  aNode.nextSibling) {
  trace(aNode);
}
```

### Voir également

`firstChild (XMLNode.firstChild, propriété)`, `appendChild (XMLNode.appendChild, méthode)`, `insertBefore (XMLNode.insertBefore, méthode)`, `removeNode (XMLNode.removeNode, méthode)`, `XML`

## nodeName (XMLNode.nodeName, propriété)

```
public nodeName : String
```

Chaîne représentant le nom du nœud de l'objet XML. Si l'objet XML est un élément XML (n`nodeType == 1`), `nodeName` est le nom de la balise qui représente le nœud dans le fichier XML. Par exemple, `TITLE` est le `nodeName` d'une balise `TITLE` HTML. Si l'objet XML est un nœud de texte (`nodeType == 3`), nodeName est `null`.

**Disponibilité :** ActionScript 1.0 ; Flash Player 5

### Exemple

L'exemple suivant crée un nœud element et un nœud text, puis vérifie le nom de nœud de ces derniers :

```
// create an XML document
var doc:XML = new XML();

// create an XML node using createElement()
var myNode:XMLNode = doc.createElement("rootNode");

// place the new node into the XML tree
doc.appendChild(myNode);

// create an XML text node using createTextNode()
var myTextNode:XMLNode = doc.createTextNode("textNode");

// place the new node into the XML tree
myNode.appendChild(myTextNode);

trace(myNode.nodeName);
```

```
trace(myTextNode.nodeName);

// output:
// rootNode
// null
```

L'exemple suivant crée un paquet XML : Si le nœud racine comporte des enfants, le code s'exécute en boucle au niveau de chaque enfant pour afficher le nom et la valeur de ce nœud. Ajoutez le code ActionScript suivant à votre fichier FLA ou AS :

```
var my_xml:XML = new XML("hankrudolph");
if (my_xml.firstChild.hasChildNodes()) {
  // use firstChild to iterate through the child nodes of rootNode
  for (var aNode:XMLNode = my_xml.firstChild.firstChild; aNode != null;
  aNode=aNode.nextSibling) {
    if (aNode.nodeType == 1) {
      trace(aNode.nodeName+":\t"+aNode.firstChild.nodeValue);
    }
  }
}
```

Les informations de nœud suivantes apparaissent dans le panneau Sortie :

```
output:
username: hank
password: rudolph
```

### Voir également

nodeType (XMLNode.nodeType, propriété)

# nodeType (XMLNode.nodeType, propriété)

```
public nodeType : Number [lecture seule]
```

Valeur `nodeType`, 1 pour un élément XML ou 3 pour un nœud texte.

La valeur `nodeType` est une valeur numérique provenant de l'énumération NodeType de la recommandation W3C DOM Level 1 : www.w3.org/tr/1998/REC-DOM-Level-1-19981001/level-one-core.html. Le tableau suivant énumère les valeurs :

| Valeur de l'entier | Constante définie |
|---|---|
| 1 | ELEMENT_NODE |
| 2 | ATTRIBUTE_NODE |
| 3 | TEXT_NODE |
| 4 | CDATA_SECTION_NODE |

| Valeur de l'entier | Constante définie |
|---|---|
| 5 | ENTITY_REFERENCE_NODE |
| 6 | ENTITY_NODE |
| 7 | PROCESSING_INSTRUCTION_NODE |
| 8 | COMMENT_NODE |
| 9 | DOCUMENT_NODE |
| 10 | DOCUMENT_TYPE_NODE |
| 11 | DOCUMENT_FRAGMENT_NODE |
| 12 | NOTATION_NODE |

Dans Flash Player, la classe XML intégrée prend uniquement en charge les types de nœud 1 (`ELEMENT_NODE`) et 3 (`TEXT_NODE`).

**Disponibilité :** ActionScript 1.0 ; Flash Player 5

## Exemple

L'exemple suivant crée un nœud element et un nœud text, puis vérifie le type de nœud de ces derniers :

```
// create an XML document
var doc:XML = new XML();

// create an XML node using createElement()
var myNode:XMLNode = doc.createElement("rootNode");

// place the new node into the XML tree
doc.appendChild(myNode);

// create an XML text node using createTextNode()
var myTextNode:XMLNode = doc.createTextNode("textNode");

// place the new node into the XML tree
myNode.appendChild(myTextNode);

trace(myNode.nodeType);
trace(myTextNode.nodeType);

// output:
// 1
// 3
```

## Voir également

nodeValue (XMLNode.nodeValue, propriété)

# nodeValue (XMLNode.nodeValue, propriété)

```
public nodeValue : String
```

Valeur du nœud de l'objet XML. Si l'objet XML est un nœud texte, le `nodeType` est 3, et le `nodeValue` est le texte du nœud. Si l'objet XML est un élément XML (`nodeType` est 1), `nodeValue` est `null` et en lecture seule

**Disponibilité :** ActionScript 1.0 ; Flash Player 5

## Exemple

L'exemple suivant crée un nœud element et un nœud text, puis vérifie leur valeur :

```
// create an XML document
var doc:XML = new XML();

// create an XML node using createElement()
var myNode:XMLNode = doc.createElement("rootNode");

// place the new node into the XML tree
doc.appendChild(myNode);

// create an XML text node using createTextNode()
var myTextNode:XMLNode = doc.createTextNode("textNode");

// place the new node into the XML tree
myNode.appendChild(myTextNode);

trace(myNode.nodeValue);
trace(myTextNode.nodeValue);

// output:
// null
// myTextNode
```

L'exemple suivant crée et analyse un package XML. Le code parcourt chaque nœud enfant et affiche sa valeur avec les propriétés `firstChild` et `firstChild.nodeValue`. Lorsque vous utilisez `firstChild` pour afficher le contenu du nœud, cette propriété préserve l'entité `&`. Cependant, lorsque vous utilisez de façon explicite `nodeValue`, cette entité est convertie en éperluette (&).

```
var my_xml:XML = new XML("mortongood&evil");
trace("using firstChild:");
for (var i = 0; i<my_xml.firstChild.childNodes.length; i++) {
  trace("\t"+my_xml.firstChild.childNodes[i].firstChild);
```

```
}
trace("");
trace("using firstChild.nodeValue:");
for (var i = 0; i<my_xml.firstChild.childNodes.length; i++) {
  trace("\t"+my_xml.firstChild.childNodes[i].firstChild.nodeValue);
}
```

Les informations suivantes apparaissent dans le panneau Sortie :

```
using firstChild:
morton
good&evil

using firstChild.nodeValue:
morton
good&evil
```

### Voir également

nodeType (XMLNode.nodeType, propriété)

## parentNode (XMLNode.parentNode, propriété)

```
public parentNode : XMLNode [lecture seule]
```

Valeur XMLNode qui fait référence au nœud parent de l'objet XML spécifié ou renvoie `null` si le nœud n'a pas de parent. Il s'agit d'une propriété en lecture seule qui ne peut pas être utilisée pour manipuler les nœuds enfants ; utilisez les méthodes `appendChild()`, `insertBefore()` et `removeNode()` pour manipuler les nœuds enfants.

**Disponibilité :** ActionScript 1.0 ; Flash Player 5

### Exemple

L'exemple suivant crée un paquet XML et affiche le nœud parent du nœud username dans le panneau Sortie :

```
var my_xml:XML = new XML("mortongood&evil");

// first child is the <login /> node
var rootNode:XMLNode = my_xml.firstChild;

// first child of the root is the <username /> node
var targetNode:XMLNode = rootNode.firstChild;
trace("the parent node of '"+targetNode.nodeName+"' is:
  "+targetNode.parentNode.nodeName);
trace("contents of the parent node are:\n"+targetNode.parentNode);
```

Sortie (sauts de ligne ajoutés pour plus de clarté):

```
the parent node of 'username' is: login
```

```
contents of the parent node are:
morton
good&evil
```

### Voir également

appendChild (XMLNode.appendChild, méthode), insertBefore (XMLNode.insertBefore, méthode), removeNode (XMLNode.removeNode, méthode), XML

## prefix (XMLNode.prefix, propriété)

```
public prefix : String [lecture seule]
```

Le préfixe du nom de nœud XML. Par exemple, le préfixe « contact » du nœud `<contact:mailbox/>bob@example.com</contact:mailbox>` et le nom local « mailbox », ce qui forme « contact.mailbox ».

La propriété `nodeName` d'un objet nœud XML renvoie le nom complet (ce qui inclut le préfixe et le nom local). Vous pouvez accéder à la section locale du nom de l'élément par l'intermédiaire de la propriété `localName`.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

### Exemple

Un répertoire contient un fichier SWF et un fichier XML. Le fichier XML, appelé « SoapSample.xml » contient le code suivant :

```
<?xml version="1.0"?>
<soap:Envelope xmlns:soap="http://www.w3.org/2001/12/soap-envelope">
<soap:Body xmlns:w="http://www.example.com/weather">
<w:GetTemperature>
<w:City>San Francisco</w:City>
</w:GetTemperature>
</soap:Body>
</soap:Envelope>
```

La source du fichier SWF contient le script suivant (remarquez les commentaires précisant les chaînes renvoyées) :

```
var xmlDoc:XML = new XML();
xmlDoc.ignoreWhite = true;
xmlDoc.load("SoapSample.xml");
xmlDoc.onLoad = function(success:Boolean)
{
  var tempNode:XMLNode = xmlDoc.childNodes[0].childNodes[0].childNodes[0];
```

```
  trace("w:GetTemperature prefix: " + tempNode.prefix); // Output: ... w
  var soapEnvNode:XMLNode = xmlDoc.childNodes[0];
  trace("soap:Envelope prefix: " + soapEnvNode.prefix); // Output: ... soap
}
```

# previousSibling (XMLNode.previousSibling, propriété)

```
public previousSibling : XMLNode [lecture seule]
```

Valeur XMLNode qui fait référence au frère précédent de la liste des enfants du nœud. La propriété a une valeur null si le nœud n'a pas de nœud frère précédent. Cette propriété ne peut pas être utilisée pour manipuler les nœuds enfants ; utilisez les méthodes `appendChild()`, `insertBefore()`, et `removeNode()` pour manipuler les nœuds enfants.

**Disponibilité :** ActionScript 1.0 ; Flash Player 5

## Exemple

L'exemple suivant reprend une partie de l'exemple correspondant à la propriété `XML.lastChild` et indique comment utiliser la propriété `XML.previousSibling` pour parcourir les enfants d'un nœud XML :

```
for (var aNode:XMLNode = rootNode.lastChild; aNode != null; aNode =
  aNode.previousSibling) {
  trace(aNode);
}
```

## Voir également

`lastChild (XMLNode.lastChild, propriété)`, `appendChild (XMLNode.appendChild, méthode)`, `insertBefore (XMLNode.insertBefore, méthode)`, `removeNode (XMLNode.removeNode, méthode)`, `XML`

# removeNode (XMLNode.removeNode, méthode)

```
public removeNode() : Void
```

Supprime l'objet XML spécifié de son parent. Supprime également tous les descendants du nœud.

**Disponibilité :** ActionScript 1.0 ; Flash Player 5

### Exemple

L'exemple suivant crée un paquet XML, puis supprime l'objet XML spécifié et ses nœuds descendants :

```
var xml_str:String = "<state name=\"California\"><city>San Francisco</
  city></state>";

var my_xml:XML = new XML(xml_str);
var cityNode:XMLNode = my_xml.firstChild.firstChild;
trace("before XML.removeNode():\n"+my_xml);
cityNode.removeNode();
trace("");
trace("after XML.removeNode():\n"+my_xml);

// output (line breaks added for clarity):
//
// before XML.removeNode():
// <state name="California">
// <city>San Francisco</city>
// </state>
//
// after XML.removeNode():
// <state name="California" />
```

## toString (XMLNode.toString, méthode)

```
public toString() : String
```

Evalue l'objet XML spécifié, crée une représentation graphique de la structure XML, comprenant le nœud, les enfants et les attributs et renvoie le résultat sous forme de chaîne.

Pour les objets XML de niveau supérieur (ceux créés avec le constructeur), la méthode `XML.toString()` produit la déclaration XML du document (enregistrée dans la propriété `XML.xmlDecl`), suivie de la déclaration `DOCTYPE` du document (enregistrée dans la propriété `XML.docTypeDecl`), suivie de la représentation de texte de tous les nœuds XML dans l'objet. La déclaration XML n'est pas produite si la propriété `XML.xmlDecl` n'est pas définie. La déclaration `DOCTYPE` n'est pas produite si la propriété `XML.docTypeDecl` est `undefined`.

**Disponibilité :** ActionScript 1.0 ; Flash Player 5

### Valeur renvoyée

`String` - Chaîne

### Exemple

Le code suivant utilise la méthode `toString()` pour convertir un objet XMLNode en chaîne, puis utilise la méthode `toUpperCase()` de la classe String :

```
var xString = "<first>Mary</first>"
    + "<last>Ng</last>"
var my_xml:XML = new XML(xString);
var my_node:XMLNode = my_xml.childNodes[1];
trace(my_node.toString().toUpperCase());
    // output: <LAST>NG</LAST>
```

### Voir également

`docTypeDecl (XML.docTypeDecl, propriété)`, `xmlDecl (XML.xmlDecl, propriété)`

## XMLNode(), constructeur

```
public XMLNode(type:Number, value:String)
```

Le constructeur XMLNode permet de créer une instance d'un nœud XML en fonction d'une chaîne et de son contenu et d'un nombre représentant son type de nœud.

**Disponibilité :** ActionScript 1.0 ; Flash Player 8

### Paramètres

*type*`:Number` - Entier représentant le type de nœud :

| Valeur de l'entier | Constante définie |
|---|---|
| 1 | ELEMENT_NODE |
| 2 | ATTRIBUTE_NODE |
| 3 | TEXT_NODE |
| 4 | CDATA_SECTION_NODE |
| 5 | NTITY_REFERENCE_NODE |
| 6 | ENTITY_NODE |
| 7 | PROCESSING_INSTRUCTION_NODE |
| 8 | COMMENT_NODE |
| 9 | DOCUMENT_NODE |
| 10 | DOCUMENT_TYPE_NODE |
| 11 | DOCUMENT_FRAGMENT_NODE |
| 12 | NOTATION_NODE |

Dans Flash Player, la classe XML prend uniquement en charge les types de nœud 1 (ELEMENT_NODE) et 3 (TEXT_NODE).

*value*:String - Pour un nœud texte, il s'agit du texte du nœud ; pour un élément nœud, il s'agit du contenu de la balise.

### Exemple

```
var ELEMENT_NODE:Number = 1;
var node1:XMLNode = new XMLNode(ELEMENT_NODE, "fullName");

var TEXT_NODE:Number = 3;
var node2:XMLNode = new XMLNode(TEXT_NODE, "Justin Case");

// Create a new XML document
var doc:XML = new XML();

// Create a root node
var rootNode:XMLNode = doc.createElement("root");

// Add the rootNode as the root of the XML document tree
doc.appendChild(rootNode);

// Build the rest of the document:
rootNode.appendChild(node1);
node1.appendChild(node2);

trace(doc);

// Output: Justin Case
```

# XMLSocket

```
Object
  |
  +-XMLSocket
```

```
public class XMLSocket
extends Object
```

La classe XMLSocket implémente les sockets client qui laissent l'ordinateur exécuter une communication Flash Player avec un ordinateur serveur identifié par une adresse IP ou un nom de domaine. La classe XMLSocket est utile pour les applications client-serveur qui requièrent un court délai, telles que des systèmes de dialogue en ligne en temps réel. Une solution de dialogue en ligne par HTTP interroge souvent le serveur et télécharge de nouveaux messages à l'aide d'une requête HTTP. Par contraste, une solution de dialogue en ligne XMLSocket maintient une connexion ouverte au serveur, qui laisse le serveur envoyer immédiatement des messages entrants sans requête du client. Pour utiliser la classe XMLSocket, l'ordinateur serveur doit exécuter un démon qui comprend le protocole utilisé par la classe XMLSocket. Le protocole est décrit dans la liste suivante :

- Les messages XML sont envoyés via une connexion socket à flux TCP/IP bidirectionnel simultané.
- Chaque message XML est un document XML complet, terminé par un octet zéro (0).
- Un nombre illimité de messages XML peut être envoyé et reçu via une connexion XMLSocket.

Les restrictions suivantes s'appliquent au mode et à l'emplacement de connexion d'un objet XMLSocket :

- La méthode XMLSocket.connect() permet une connexion uniquement à des numéros de port TCP supérieurs ou égaux à 1024. Une conséquence de cette restriction est que les démons du serveur qui communiquent avec l'objet XMLSocket doivent également être attribués aux numéros de port supérieurs ou égaux à 1024. Les numéros de port inférieurs à 1024 sont souvent utilisés par des services système tels que FTP, Telnet, et HTTP, les objets XMLSocket sont donc interdits au niveau de ces ports pour des raisons de sécurité. La restriction du numéro de port limite la possibilité d'accès à ces ressources et leur mauvaise utilisation.
- La méthode XMLSocket.connect() permet une connexion uniquement aux ordinateurs se trouvant dans le domaine de résidence du fichier SWF. Cette restriction ne s'applique pas aux fichiers SWF s'exécutant en local. (Cette restriction est identique aux règles de sécurité pour loadVariables(), XML.sendAndLoad() et XML.load().) Pour se connecter à un démon de serveur s'exécutant dans un domaine différent du domaine de résidence du SWF, vous pouvez créer un fichier de régulation de sécurité sur le serveur qui permette d'accéder à partir de domaines spécifiques.

La configuration d'un serveur en vue de la communication avec un objet XMLSocket peut être difficile à réaliser. Si votre application ne requiert pas d'interactivité en temps réel, utilisez la fonction loadVariables(), ou la connectivité serveur XML basée sur HTTP Flash (XML.load(), XML.sendAndLoad(), XML.send()), à la place de la classe XMLSocket. Pour utiliser les méthodes de la classe XMLSocket, vous devez d'abord utiliser le constructeur, new XMLSocket, pour créer un objet XMLSocket.

**Disponibilité :** ActionScript 1.0 ; Flash Player 5

## Résumé des propriétés

*Propriétés héritées de la classe Object*

`constructeur (propriété Object.constructor), __proto__ (Object.__proto__ property), prototype (Object.prototype, propriété), __resolve (Object.__resolve, propriété)`

## Résumé des événements

| Evénement | Description |
|---|---|
| `onClose = function() {}` | Appelé uniquement lorsqu'une connexion ouverte est fermée par le serveur. |
| `onConnect = function(success :Boolean) {}` | Appelé par Flash Player lorsqu'une requête de connexion lancée via `XMLSocket.connect()` a un résultat ou non. |
| `onData = function(src:Str ing) {}` | Appelé lorsqu'un message a été téléchargé à partir du serveur, terminé par un octet zéro (0). |
| `onXML = function(src:XML ) {}` | Appelé par Flash Player lorsque l'objet XML spécifié contenant un document XML arrive sur une connexion XMLSocket ouverte. |

## Récapitulatif des constructeurs

| Signature | Description |
|---|---|
| `XMLSocket()` | Crée un objet XMLSocket. |

### Résumé de la méthode

| Modificateurs | Signature | Description |
|---|---|---|
| | `close() : Void` | Ferme la connexion spécifiée par l'objet XMLSocket. |
| | `connect(url:String, port:Number) : Boolean` | Etablit une connexion avec l'hôte Internet spécifié à l'aide du port TCP spécifié (doit être de 1024 ou plus), et renvoie `true` ou `false`, , en fonction de la connexion établie ou non. |
| | `send(data:Object) : Void` | Convertit l'objet ou les données XML spécifiés dans le paramètre `object` en une chaîne et la transmet au serveur, suivie d'un octet zéro (0). |

*Méthodes héritées de la classe Object*

`addProperty (méthode Object.addProperty), hasOwnProperty (méthode Object.hasOwnProperty), isPropertyEnumerable (méthode Object.isPropertyEnumerable), isPrototypeOf (méthode Object.isPrototypeOf), registerClass (méthode Object.registerClass), toString (méthode Object.toString), unwatch (méthode Object.unwatch), valueOf (méthode Object.valueOf), watch (méthode Object.watch)`

## close (XMLSocket.close, méthode)

`public close() : Void`

Ferme la connexion spécifiée par l'objet XMLSocket.

**Disponibilité :** ActionScript 1.0 ; Flash Player 5

### Exemple

L'exemple simple suivant crée un objet XMLSocket, tente de se connecter au serveur, puis termine la connexion.

```
var socket:XMLSocket = new XMLSocket();
socket.connect(null, 2000);
socket.close();
```

### Voir également

`connect (XMLSocket.connect, méthode)`

# connect (XMLSocket.connect, méthode)

```
public connect(url:String, port:Number) : Boolean
```

Etablit une connexion avec l'hôte Internet spécifié à l'aide du port TCP spécifié (doit être de 1024 ou plus), et renvoie `true` ou `false`,, en fonction de la connexion établie ou non. Si vous ne connaissez pas le numéro de port de votre ordinateur hôte Internet, contactez votre administrateur réseau.

Si vous spécifiez `null` pour le paramètre `host`, l'hôte contacté est celui où réside le fichier SWF appelant `XMLSocket.connect()`. Par exemple, si le fichier SWF a été téléchargé à partir du site www.yoursite.com, le fait de spécifier `null` pour le paramètre hôte équivaut à entrer l'adresse IP pour www.yoursite.com.

Pour les fichiers SWF lus par une version antérieure à Flash Player 7, le paramètre `host` doit correspondre au superdomaine du fichier SWF envoyant cet appel. Par exemple, un fichier SWF à l'adresse www.someDomain.com peut charger des variables provenant d'un fichier SWF à l'adresse store.someDomain.com, étant donné que les deux fichiers sont dans le même superdomaine de someDomain.com.

Dans les fichiers SWF d'une version exécutée dans Flash Player 7 ou une version ultérieure, le paramètre `host` doit se trouver exactement dans le même domaine. Par exemple, un fichier SWF à l'adresse www.someDomain.com qui est publié pour Flash Player 5, mais s'exécute dans Flash Player 7 ou une version ultérieure peut charger des variables uniquement à partir de fichiers SWF qui sont également à l'adresse www.someDomain.com. Si vous souhaitez charger des variables à partir d'un domaine différent, vous pouvez placer un *fichier de régulation interdomaines* sur le serveur hébergeant le fichier SWF qui est utilisé.

Lorsque `load()` est exécuté, la propriété d'objet XML `loaded`est définie sur `false`. Lorsque le téléchargement des données XML se termine, la propriété `loaded` est définie sur `true` et le gestionnaire d'événements `onLoad` est appelé. Les données XML ne sont pas analysées avant la fin du téléchargement. Si l'objet XML contenait précédemment des arborescences XML, elles sont supprimées.

Si `XMLSocket.connect()` renvoie une valeur `true`, le niveau initial du processus de connexion est accessible ; puis la méthode `XMLSocket.onConnect` est appelée pour déterminer si la connexion finale a été établie ou non. Si `XMLSocket.connect()` renvoie `false`, la connexion ne peut pas être établie.

Lorsque vous employez cette méthode, considérez le modèle de sécurité Flash Player.

- Pour Flash Player 8, elle n'est pas autorisée si le fichier SWF appelant est le sandbox local-with-file-system.
- Pour Flash Player 7 et postérieur, les sites Web permettent l'accès à des ressources, par des demandeurs issus de différent domaines, via un fichier de régulation inter-domaines.

Pour plus d'informations, consultez :

- Chapitre 17, « Understanding Security » dans *Learning ActionScript 2.0 in Flash*
- Le livre blanc Flash Player 8 Security disponible à l'adresse : http://www.macromedia.com/go/fp8_security
- Le livre blanc Flash Player 8 Security-Related APIs disponible à l'adresse : http://www.macromedia.com/go/fp8_security_apis

**Disponibilité :** ActionScript 1.0 ; Flash Player 5

## Paramètres

*url*:String - Chaîne ; nom de domaine DNS avec tous ses qualificatifs ou adresse IP sous la forme *aaa.bbb.ccc.ddd*. Vous pouvez également spécifier `null` pour vous connecter au serveur hôte qui héberge le fichier SWF. Si le fichier SWF effectuant cet appel s'exécute dans un navigateur Web, `host` doit appartenir au même domaine que le fichier SWF. Pour plus de détails, reportez-vous aux informations sur les restrictions de domaines des fichiers SWF, dans la description principale de cette méthode.

*port*:Number - Nombre ; numéro de port TCP de l'hôte utilisé pour établir une connexion. Le numéro de port doit être supérieur ou égal à 1 024.

## Valeur renvoyée

Boolean - `true` si la connexion a été établie avec succès ; `false` dans tous les autres cas.

## Exemple

L'exemple suivant utilise `XMLSocket.connect()` pour la connexion à l'hôte hébergeant le fichier SWF et utilise la commande `trace` pour afficher la valeur renvoyée et indiquer ainsi l'aboutissement ou l'échec de la connexion :

```
var socket:XMLSocket = new XMLSocket()
socket.onConnect = function (success:Boolean) {
  if (success) {
  trace ("Connection succeeded!")
  } else {
  trace ("Connection failed!")
  }
}
if (!socket.connect(null, 2000)) {
  trace ("Connection failed!")
}
```

## Voir également

`onConnect (XMLSocket.onConnect, gestionnaire)`, `Instruction function`

# onClose (XMLSocket.onClose, gestionnaire)

```
onClose = function() {}
```

Appelé uniquement lorsqu'une connexion ouverte est fermée par le serveur. Par défaut, l'implémentation de cette méthode n'effectue aucune action. Pour annuler l'implémentation par défaut, vous devez attribuer une fonction contenant des actions personnalisées.

**Disponibilité :** ActionScript 1.0 ; Flash Player 5

## Exemple

L'exemple suivant utilise une instruction trace lorsqu'une connexion en cours est fermée par le serveur :

```
var socket:XMLSocket = new XMLSocket();
socket.connect(null, 2000);
socket.onClose = function () {
  trace("Connection to server lost.");
}
```

## Voir également

`onConnect (XMLSocket.onConnect, gestionnaire)`, `Instruction function`

# onConnect (XMLSocket.onConnect, gestionnaire)

```
onConnect = function(success:Boolean) {}
```

Appelé par Flash Player lorsqu'une requête de connexion lancée via `XMLSocket.connect()` a un résultat ou non. Si la connexion a réussi, le paramètre `success` est `true`; dans tous les autres cas le paramètre `success` est `false`.

Par défaut, l'implémentation de cette méthode n'effectue aucune action. Pour annuler l'implémentation par défaut, vous devez attribuer une fonction contenant des actions personnalisées.

**Disponibilité :** ActionScript 1.0 ; Flash Player 5

## Paramètres

*`success`*`:Boolean` - Valeur booléenne indiquant si une connexion socket est établie (`true` ou `false`).

## Exemple

L'exemple suivant illustre le processus de spécification d'une fonction de remplacement pour la méthode `onConnect` dans le cadre d'une application simple de dialogue en ligne.

Le script crée un objet XMLSocket avec la méthode constructeur, puis définit la fonction personnalisée à exécuter lorsque le gestionnaire d'événements onConnect est appelé. Cette fonction contrôle l'écran destiné aux utilisateurs, à condition que la connexion soit établie avec succès. Dans ce cas, les utilisateurs ont accès à l'écran principal de dialogue à partir de l'image appelée `startChat`. Si la connexion ne peut pas être établie, les utilisateurs sont dirigés vers un écran informatif, sur l'image appelée `connectionFailed`.

```
var socket:XMLSocket = new XMLSocket();
socket.onConnect = function (success) {
  if (success) {
    gotoAndPlay("startChat");
  } else {
    gotoAndStop("connectionFailed");
  }
}
```

Enfin, la connexion est initiée. Si `connect()`renvoie `false`, le fichier SWF est envoyé directement vers une image appelée `connectionFailed`, `onConnect` n'est plus appelé. Si `connect()` renvoie `true`, le fichier SWF revient à une image appelée `waitForConnection`, qui correspond à l'écran « Please wait ». Le fichier SWF revient à l'image `waitForConnection` jusqu'à ce que le gestionnaire `onConnect` soit appelé. Le délai de cet appel ne peut être prédit avec précision en raison de la latence du réseau.

```
if (!socket.connect(null, 2000)) {
  gotoAndStop("connectionFailed");
} else {
  gotoAndStop("waitForConnection");
}
```

### Voir également

`connect (XMLSocket.connect, méthode)`, `Instruction function`

## onData (XMLSocket.onData, gestionnaire)

```
onData = function(src:String) {}
```

Appelé lorsqu'un message a été téléchargé à partir du serveur, terminé par un octet zéro (0). Vous pouvez neutraliser `XMLSocket.onData` pour intercepter les données envoyées par le serveur sans l'analyser comme XML. Ceci est utile si vous transmettez de manière arbitraire des paquets de données formatées et si vous préférez manipuler directement les données lorsqu'elles arrivent, plutôt que de faire analyser les données comme XML par Flash Player.

Par défaut, la méthode `XMLSocket.onData` appelle la méthode `XMLSocket.onXML`. Si vous neutralisez `XMLSocket.onData` par un comportement personnalisé, `XMLSocket.onXML` n'est pas appelé, sauf si vous l'appelez dans votre implémentation de `XMLSocket.onData`.

**Disponibilité :** ActionScript 1.0 ; Flash Player 5

### Paramètres

*src*:String - Chaîne contenant les données envoyées par le serveur.

### Exemple

Dans cet exemple, le paramètre src est une chaîne contenant du texte XML téléchargé à partir du serveur. Le terminateur de zéro (0) octet n'est pas inclus dans la chaîne.

```
XMLSocket.prototype.onData = function (src) {
  this.onXML(new XML(src));
}
```

## onXML (XMLSocket.onXML, gestionnaire)

```
onXML = function(src:XML) {}
```

Appelé par Flash Player lorsque l'objet XML spécifié contenant un document XML arrive sur une connexion XMLSocket ouverte. Une connexion XMLSocket peut être utilisée pour transférer un nombre illimité de documents XML entre le client et le serveur. Chaque document se termine par un octet zéro (0). Lorsque Flash Player reçoit l'octet zéro, il analyse tous les XML reçus depuis l'octet zéro précédent ou depuis que la connexion a été établie s'il s'agit du premier message reçu. Chaque lot de XML analysé est traité comme un document XML unique et transmis à la méthode onXML.

Par défaut, l'implémentation de cette méthode n'effectue aucune action. Pour annuler l'implémentation par défaut, vous devez attribuer une fonction contenant des actions que vous définissez.

**Disponibilité :** ActionScript 1.0 ; Flash Player 5

### Paramètres

*src*:XML - Objet XML contenant un document XML analysé reçu d'un serveur.

### Exemple

La fonction suivante remplace l'implémentation par défaut de la méthode onXML dans le cadre d'une application simple de discussion. La fonction myOnXML force l'application de discussion à reconnaître un élément XML unique, MESSAGE, au format suivant :

```
<MESSAGE USER="John" TEXT="Hello, my name is John!" />.
```

La fonction suivante, `displayMessage()`, est considérée comme une fonction définie par l'utilisateur qui affiche le message reçu par l'utilisateur :

```
var socket:XMLSocket = new XMLSocket();
socket.onXML = function (doc) {
  var e = doc.firstChild;
  if (e != null && e.nodeName == "MESSAGE") {
    displayMessage(e.attributes.user, e.attributes.text);
  }
}
```

### Voir également

`Instruction function`

## send (XMLSocket.send, méthode)

```
public send(data:Object) : Void
```

Convertit l'objet ou les données XML spécifiés dans le paramètre `object` en une chaîne et la transmet au serveur, suivie d'un octet zéro (0). Si `object` est un objet XML, la chaîne est la représentation textuelle XML de l'objet XML. L'opération d'envoi est asynchrone ; elle est immédiatement renvoyée, mais les données peuvent être transmises plus tard. La méthode `XMLSocket.send()` ne renvoie pas de valeur indiquant si les données ont bien été transmises.

Si l'objet *`myXMLSocket`* n'est pas connecté au serveur (à l'aide de XMLSocket.connect()), l'opération `XMLSocket.send()` échoue.

**Disponibilité :** ActionScript 1.0 ; Flash Player 5

### Paramètres

*`data`*`:Object` - Objet XML ou toute autre donnée à transmettre au serveur.

### Exemple

L'exemple suivant indique comment spécifier un nom d'utilisateur et un mot de passe pour envoyer l'objet `my_xml` au serveur :

```
var myXMLSocket:XMLSocket = new XMLSocket();
var my_xml:XML = new XML();
var myLogin:XMLNode = my_xml.createElement("login");
myLogin.attributes.username = usernameTextField;
myLogin.attributes.password = passwordTextField;
my_xml.appendChild(myLogin);
myXMLSocket.send(my_xml);
```

### Voir également

`connect (XMLSocket.connect, méthode)`

## XMLSocket, constructeur

```
public XMLSocket()
```

Crée un objet XMLSocket. L'objet XMLSocket n'est initialement pas connecté à un serveur. Vous devez appeler XMLSocket.connect() pour connecter l'objet à un serveur.

**Disponibilité :** ActionScript 1.0 ; Flash Player 5

### Exemple

L'exemple suivant crée un objet XMLSocket :

```
var socket:XMLSocket = new XMLSocket();
```

# XMLUI

```
Object
  |
  +-XMLUI
```

```
public class XMLUI
extends Object
```

L'objet XMLUI permet la communication avec les fichiers SWF qui sont utilisés comme interfaces utilisateur personnalisées pour les fonctions d'extensibilité de l'outil de programmation Flash (telles que Comportements, Commandes, Effets et Outils).

Macromedia Flash MX 2004 et Macromedia Flash MX Professional 2004 présentent plusieurs fonctions d'extensibilité incluant Comportements, Commandes (JavaScript API), Effets, et Outils. Avec ces fonctions, les utilisateurs expérimentés peuvent développer ou automatiser la fonctionnalité de l'outil de programmation. Le moteur XML vers UI fonctionne avec ces fonctions d'extensibilité pour créer des boîtes de dialogue qui s'affichent lorsque l'extension nécessite ou valide des paramètres. Les boîtes de dialogue peuvent être définies à l'aide de balises XML ou en créant un fichier SWF pour l'affichage. L'objet XMLUI fournit un mécanisme par lequel un utilisateur avancé peut communiquer avec un fichier SWF utilisé de cette manière.

**Disponibilité :** ActionScript 1.0 ; Flash Player 7

### Résumé des propriétés

*Propriétés héritées de la classe Object*

| |
|---|
| constructeur (propriété Object.constructor), __proto__ (Object.__proto__ property), prototype (Object.prototype, propriété), __resolve (Object.__resolve, propriété) |

Résumé de la méthode

| Modificateurs | Signature | Description |
|---|---|---|
| `static` | `accept() : Void` | Ferme la boîte de dialogue XMLUI et lui attribue l'état " accept ". |
| `static` | `cancel() : Void` | Permet de quitter la boîte de dialogue XMLUI en cours avec un état " annuler ". |
| `static` | `get(name:String) : String` | Extrait la valeur de la propriété spécifiée de la boîte de dialogue XMLUI actuelle. |
| `static` | `set(name:String, value:String) : Void` | Modifie la valeur de la propriété spécifiée de la boîte de dialogue XMLUI actuelle. |

*Méthodes héritées de la classe Object*

`addProperty (méthode Object.addProperty), hasOwnProperty (méthode Object.hasOwnProperty), isPropertyEnumerable (méthode Object.isPropertyEnumerable), isPrototypeOf (méthode Object.isPrototypeOf), registerClass (méthode Object.registerClass), toString (méthode Object.toString), unwatch (méthode Object.unwatch), valueOf (méthode Object.valueOf), watch (méthode Object.watch)`

# accept (XMLUI.accept, méthode)

```
public static accept() : Void
```

Ferme la boîte de dialogue XMLUI et lui attribue l'état « accept ». Identique à l'utilisateur en cliquant sur le bouton OK.

**Disponibilité :** ActionScript 1.0 ; Flash Player 7

# cancel (XMLUI.cancel, méthode)

```
public static cancel() : Void
```

Permet de quitter la boîte de dialogue XMLUI en cours avec un état « annuler ». Identique à l'utilisateur en cliquant sur le bouton Annuler.

**Disponibilité :** ActionScript 1.0 ; Flash Player 7

## get (XMLUI.get, méthode)

```
public static get(name:String) : String
```

Extrait la valeur de la propriété spécifiée de la boîte de dialogue XMLUI actuelle.

**Disponibilité :** ActionScript 1.0 ; Flash Player 7

### Paramètres

*name*:String - Nom de la propriété XMLUI à extraire.

### Valeur renvoyée

String - Renvoie la valeur de la propriété sous forme de chaîne.

## set (XMLUI.set, méthode)

```
public static set(name:String, value:String) : Void
```

Modifie la valeur de la propriété spécifiée de la boîte de dialogue XMLUI actuelle.

**Disponibilité :** ActionScript 1.0 ; Flash Player 7

### Paramètres

*name*:String - Nom de la propriété XMLUI à modifier.

*value*:String - Valeur selon laquelle la propriété spécifiée sera définie.

CHAPITRE 3

# ActionScript déconseillé

3

L'évolution d'ActionScript a rendu caducs de nombreux éléments du langage. Cette section liste les articles déconseillés et suggère d'autres possibilités quand cela est possible. Si des éléments déconseillés fonctionnent encore dans Flash Player 8, Macromedia vous recommande de ne plus les employer dans votre code. La prise en charge à long terme des éléments déconseillés n'est pas garantie.

## Classe déconseillée

| Modificateurs | Nom de classe | Description |
|---|---|---|
| | Color | *Déconseillé* depuis Flash Player 8. La class Color a été déconseillée au profit de la classe flash.geom.ColorTransform. |

## Fonctions déconseillées

| Modificateurs | Nom de la fonction | Description |
|---|---|---|
| | call(frame:Object) | *Déconseillée* depuis Flash Player 5. Cette action a été déconseillée au profit de l'instruction `function`. |
| | chr(number:Number) | *Déconseillée* depuis Flash Player 5. Cette action a été déconseillée au profit de `String.fromCharCode()`. |
| | TextFormat.getTextExtent(text:String, [width:Number]) | *Déconseillé* depuis Flash Player 8. N'a pas été remplacée. |
| | ifFrameLoaded([scene:String], frame:Object) | *Déconseillée* depuis Flash Player 5. Cette fonction est déconseillée. Macromedia vous recommande d'employer la propriété `MovieClip._framesloaded`. |

| Modificateurs | Nom de la fonction | Description |
|---|---|---|
| | int(value:Number) | *Déconseillée* depuis Flash Player 5. Cette action a été déconseillée au profit de `Math.round()`. |
| | length(expression:String, variable:Object) | *Déconseillée* depuis Flash Player 5. Cette fonction, de même que les fonctions de chaîne, est déconseillée. Macromedia vous recommande d'employer les méthodes de la classe String et la propriété `String.length` pour effectuer les mêmes opérations. |
| | mbchr(number:Number) | *Déconseillée* depuis Flash Player 5. Cette fonction a été déconseillée au profit de la méthode `String.fromCharCode()`. |
| | mblength(string:String) | *Déconseillée* depuis Flash Player 5. Cette fonction a été déconseillée au profit de la méthode et des propriétés de la classe String. |
| | mbord(character:String) | *Déconseillée* depuis Flash Player 5. Cette action a été déconseillée au profit de `String.charCodeAt()`. |
| | mbsubstring(value:String, index:Number, count:Number) | *Déconseillée* depuis Flash Player 5. Cette action a été déconseillée au profit de `String.substr()`. |
| | ord(character:String) | *Déconseillée* depuis Flash Player 5. Cette fonction a été déconseillée au profit de la méthode et des propriétés de la classe String. |
| | random(value:Number) | *Déconseillée* depuis Flash Player 5. Cette action a été déconseillée au profit de `Math.random()`. |
| | substring(string:String, index:Number, count:Number) | *Déconseillée* depuis Flash Player 5. Cette action a été déconseillée au profit de `String.substr()`. |
| | tellTarget(target:String, instruction(s)) | *Déconseillée* depuis Flash Player 5. Macromedia vous recommande d'employer la notation avec point (.) et l'instruction `with`. |
| | toggleHighQuality() | *Déconseillée* depuis Flash Player 5. Cette action a été déconseillée au profit de `_quality`. |

# Propriétés déconseillées

| Modificateurs | Nom de la propriété | Description |
|---|---|---|
| | Button._highquality | *Déconseillée* à partir de Flash Player 7. Il est recommandé d'utiliser `Button._quality`. |
| | MovieClip._highquality | *Déconseillée* à partir de Flash Player 7. Il est recommandé d'utiliser `MovieClip._quality`. |
| | TextField._highquality | *Déconseillée* à partir de Flash Player 7. Il est recommandé d'utiliser `TextField._quality`. |
| | _highquality | *Déconseillée* depuis Flash Player 5. Cette propriété a été déconseillée au profit de `_quality`. |
| | maxscroll | *Déconseillée* depuis Flash Player 5. Cette propriété a été déconseillée au profit de `TextField.maxscroll`. |
| | scroll | *Déconseillée* depuis Flash Player 5. Cette propriété a été déconseillée au profit de `TextField.scroll`. |

# Opérateurs déconseillés

| Opérateur | Description |
|---|---|
| <>(inégalité) | *Déconseillé* depuis Flash Player 5. Cet opérateur est déconseillé. Macromedia vous recommande d'employer l'opérateur `!= (inequality)`. |
| add (concaténation (chaînes)) | *Déconseillé* depuis Flash Player 5. Macromedia vous recommande d'employer l'opérateur d'addition (+) lorsque vous créez du contenu pour Flash Player 5 ou ultérieur. Cet opérateur n'est pas pris en charge dans Flash Player 8 ou ultérieur. |
| and (ET logique) | *Déconseillé* depuis Flash Player 5. Macromedia vous recommande d'employer l'opérateur ET (AND) logique (`&&`). |
| eq (égalité (chaînes)) | *Déconseillé* depuis Flash Player 5. Cet opérateur a été déconseillé au profit de l'opérateur `== (equality)`. |
| ge (plus grand que ou égal à (chaînes)) | *Déconseillé* depuis Flash Player 5. Cet opérateur a été déconseillé au profit de >= (plus grand que ou égal à). |
| gt (plus grand que (chaînes)) | *Déconseillé* depuis Flash Player 5. Cet opérateur a été déconseillé au profit de l'opérateur > (plus grand que). |

| Opérateur | Description |
|---|---|
| le (plus petit que ou égal à (chaînes)) | *Déconseillé* depuis Flash Player 5. Cet opérateur a été déconseillé dans Flash 5 au profit de <= (plus petit que ou égal à). |
| lt (plus petit que (chaînes)) | *Déconseillé* depuis Flash Player 5. Cet opérateur a été déconseillé au profit de l'opérateur < (plus petit que). |
| ne (inégalité (chaînes)) | *Déconseillé* depuis Flash Player 5. Cet opérateur a été déconseillé au profit de l'opérateur `!= (inequality)`. |
| non (NON logique) | *Déconseillé* depuis Flash Player 5. Cet opérateur a été déconseillé au profit de l'opérateur `! (logical NOT)`. |
| ou (OU logique) | *Déconseillé* depuis Flash Player 5. Cet opérateur a été déconseillé au profit de l'opérateur `|| (logical OR)`. |

# Index

## Symboles



## A

## B

## C

## D

## E



## F

## G

## H



## I

## K



## L

# M

## N

## O

## P

## R



## S

## T

## U



## V



## W



## X

## Y