ADOBE® FLASH® LITE™ 2.x et 3.x

Guide de référence du langage Adobe® ActionScript®

Adobe

Guide de référence du langage ActionScript™ Flash® Lite™ 2.x et 3.x d'Adobe®

Adobe Systems Incorporated, 345 Park Avenue, San Jose, California 95110, USA.

Sommaire

Chapitre 1 : Eléments du langage ActionScript

Cette section fournit des informations sur la syntaxe, l'utilisation et des exemples de code concernant les fonctions et les propriétés globales (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.

Résumé des directives de compilation

Directive	Description
#endinitclip	Directive de compilation ; indique la fin d'un bloc d'actions d'initialisation.
#include	Directive de compilation : inclut le contenu du fichier spécifié, comme si les commandes du fichier faisaient partie du script d'appel.
#initclip	Directive de compilation ; indique le début d'un bloc d'actions d'initialisation.

#endinitclip, directive

```
#endinitclip
```

Directive de compilation ; indique la fin d'un bloc d'actions d'initialisation.

Disponibilité

Flash Lite™ 2.0

Exemple

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

#include, directive

```
#include "[path]filename.as"
```

Remarque : *Ne placez pas de point-virgule (;) à la fin de la ligne qui contient l'instruction #include.*

Directive de compilation : 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 or Windows XP: C:\Documents and Settings*utilisateur* \Local

  Settings\Application Data\Adobe\Flash 10*langue*\Configuration\Include

  Windows Vista®: C:\Users*utilisateur* \Local Settings\ Application Data\Adobe\Flash 8*langue*\Configuration\Include

  Macintosh® OS X: Disque dur/Users/Library/Application Support/Adobe/Flash 10/*langue*/Configuration/Include
- Le répertoire *programme Flash* *langue*\First Run\Include. Si vous enregistrez un fichier à cet endroit, il est copié dans le répertoire Include global lors du démarrage suivant 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.)

Remarque : *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.*

Disponibilité

Flash Lite 2.0

Paramètres

`[chemin]nom_fichier.as` - *nom_fichier.as* Nom de fichier et 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"
```

#initclip, directive

`#initclip` *`order`*

Remarque : *Ne placez pas de point-virgule (;) à la fin de la ligne qui contient l'instruction #initclip.*

Directive de compilation ; 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 s'exécutent lorsqu'un symbole de clip est défini. Si le clip est un symbole exporté, les actions d'initialisation s'exécutent avant les actions de l'image 1 du fichier SWF. Sinon, elles s'exécutent immédiatement avant les actions portant sur l'image qui contient la première occurrence du symbole de clip correspondant.

Les actions d'initialisation ne s'exécutent qu'une seule fois lors de la lecture d'un fichier SWF ; utilisez-les pour les initialisations uniques, telles que les opérations de définition de classe et d'inscription.

Disponibilité
Flash Lite 2.0

Paramètres
`order` - 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.

false, constante

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 Lite 1.1

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
```

Infinity, constante

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

Disponibilité
Flash Lite 2.0

Voir aussi
`POSITIVE_INFINITY (propriété Number.POSITIVE_INFINITY)`

-Infinity, constante

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

Disponibilité
Flash Lite 2.0

Voir aussi
`NEGATIVE_INFINITY (propriété Number.NEGATIVE_INFINITY)`

NaN, constante

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 Lite 1.1

Voir aussi
`isNaN, fonction`, `NaN (propriété Number.NaN)`

newline, constante

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 Lite 1.1

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 aussi

trace, fonction

null, constante

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 Lite 1.1

Exemple

Dans un contexte numérique, null renvoie 0. Vous pouvez exécuter des tests d'égalité avec null. Dans cette instruction, le nœud d'arbre binaire ne comporte pas d'enfant gauche, par conséquent le champ de cet enfant gauche peut être défini sur null.

```
if (tree.left == null) {
    tree.left = new TreeNode();
}
```

true, constante

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 Lite 1.1

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 aussi

`false, constante`, `Boolean`

undefined, constante

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` 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 précédente, 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 Lite 1.1

Exemple

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

Dans la première section du code, l'opérateur d'égalité (`==`) compare la valeur de `x` à la valeur `undefined` ; le résultat s'affiche dans le panneau Sortie. Dans la première section du code, l'opérateur d'égalité (`==`) compare la valeur de `x` à la valeur `undefined` ; le résultat s'affiche dans le fichier journal.

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 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], [elementN]) : Array`	Crée un tableau vide ou convertit les éléments spécifiés en tableau.
	Boolean `(expression:Object) : Boolean`	Convertit le paramètre ***`expression`*** en une valeur booléenne et renvoie `true` ou `false`.
	call `(frame:Object)`	**Déconseillé depuis Flash Player 5.** Cette action est déconseillée au profit de l'instruction `function`. Exécute le script dans l'image appelée sans positionner la tête de lecture sur celle-ci.
	chr `(number:Number) : String`	**Déconseillé depuis Flash Player 5.** Cette fonction est déconseillée au profit de `String.fromCharCode()`. Convertit les numéros de code ASCII en caractères.
	clearInterval `(intervalID:Number)`	Annule un intervalle créé par un appel à `setInterval()`.
	duplicateMovieClip `(target:Object, newname:String, depth:Number)`	Crée une occurrence de clip pendant la lecture du fichier SWF.

Modificateurs	Signature	Description
	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 (%).
	eval (expression:Object) : Object	Accède aux variables, propriétés, objets ou clips en fonction de leur nom.
	fscommand (command:String, parameters:String)	Permet à un fichier SWF de communiquer avec le lecteur Flash Lite ou l'environnement d'un périphérique mobile (tel qu'un système d'exploitation).
	fscommand2 (command:String, parameters:String)	Permet au fichier SWF de communiquer avec le lecteur Flash Lite ou une application hôte sur un périphérique mobile.
	getProperty (my_mc:Object, property:Object) : Object	**Déconseillé** depuis Flash Player 5. La syntaxe à point, introduite dans Flash Player 5, est à préférer. Renvoie la valeur de la propriété spécifiée pour le clip ***my_mc***.
	getTimer () : Number	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 () : String	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, statement(s):Object)	**Déconseillé depuis Flash Player 5.** Cette fonction est déconseillée. Adobe recommande d'employer la propriété `MovieClip._framesloaded`. Vérifie si le contenu d'une image spécifique est disponible localement.
	int (value:Number) : Number	**Déconseillé depuis Flash Player 5.** Cette fonction est déconseillée au profit de `Math.round()`. Convertit un nombre décimal en valeur entière en tronquant la valeur décimale.
	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.
	isNaN (expression:Object) : Boolean	Evalue le paramètre et renvoie `true` si la valeur est `NaN` (not a number - n'est pas un nombre).
	length (expression:String, variable:Object) : Number	**Déconseillé depuis Flash Player 5.** Cette fonction, de même que les fonctions de chaîne, est déconseillée. Adobe recommande d'employer 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.
	loadMovie (url:String, target:Object, [method:String])	Charge un fichier SWF ou JPEG dans Flash Player pendant la lecture du fichier SWF d'origine.

Modificateurs	Signature	Description
	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)	**Déconseillé depuis Flash Player 5.** Cette fonction est déconseillée au profit de la méthode `String.fromCharCode()`. Convertit un numéro de code ASCII en caractère multi-octets.
	mblength (string:String) : Number	**Déconseillé depuis Flash Player 5.** Cette fonction est déconseillée au profit de la propriété `String.length`. Renvoie la longueur de la chaîne de caractères multi-octets.
	mbord (character:String) : Number	**Déconseillé depuis Flash Player 5.** Cette fonction est déconseillée au profit de la méthode `String.charCodeAt()`. Convertit le caractère spécifié en nombre multi-octets.
	mbsubstring (value:String, index:Number, count:Number) : String	**Déconseillé depuis Flash Player 5.** Cette fonction est déconseillée au profit de la méthode `String.substr()`. Extrait une nouvelle chaîne de caractères multi-octets d'une chaîne de caractères multi-octets.
	nextFrame ()	Place la tête de lecture sur l'image suivante.
	nextScene ()	Place la tête de lecture sur l'image 1 de la séquence suivante.
	Number (expression:Object) : Number	Convertit le paramètre `expression` en valeur numérique.
	Object ([value:Object]) : 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) : Number	**Déconseillé depuis Flash Player 5.** Cette fonction est déconseillée au profit des méthodes et des propriétés de la classe String. Convertit les caractères en numéros de code ASCII.
	parseFloat (string:String) : Number	Convertit une chaîne en nombre à virgule flottante.
	parseInt (expression:String, [radix:Number]) : Number	Convertit une chaîne en entier.
	play ()	Fait avancer la tête de lecture au sein du scénario.

Modificateurs	Signature	Description
	prevFrame()	Place la tête de lecture sur l'image précédente.
	prevScene()	Place la tête de lecture sur l'image 1 de la séquence précédente.
	random(value:Number) : Number	**Déconseillé depuis Flash Player 5.** Cette fonction est déconseillée au profit de `Math.random()`. Renvoie un entier aléatoire compris entre 0 et un inférieur au nombre entier spécifié dans le paramètre `value`.
	removeMovieClip(target:Object)	Supprime le clip spécifié.
	setInterval(functionName:Object, interval:Number, [param:Object], objectName:Object, methodName:String) : Number	Appelle une fonction ou une méthode ou 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.
	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) : String	Renvoie une chaîne représentant le paramètre spécifié.
	substring(string:String, index:Number, count:Number) : String	**Déconseillé depuis Flash Player 5.** Cette fonction est déconseillée au profit de `String.substr()`. Extrait une partie d'une chaîne.
	targetPath(targetObject:Object) : String	Renvoie une chaîne contenant le chemin cible de `movieClipObject`.
	tellTarget(target:String, statement(s):Object)	**Déconseillé depuis Flash Player 5.** Adobe recommande d'utiliser une notation de type point (.) et l'instruction `with`. Cette fonction applique les instructions spécifiées dans le paramètre `statements` au scénario spécifié par le paramètre `target`.
	toggleHighQuality()	**Déconseillé depuis Flash Player 5.** Cette fonction est déconseillée au profit de `_quality`. Active et désactive l'anti-aliasing dans Flash Player.
	trace(expression:Object)	Evalue l'expression et renvoie le résultat.

Modificateurs	Signature	Description
	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 cette chaîne.
	unloadMovie(target)	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 chargés par l'intermédiaire de la fonction loadMovieNum() de Flash Player.

Array, fonction

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

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

Permet de 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 Lite 2.0

Paramètres

`numElements` [facultatif] - 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` [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.

Valeur renvoyée

`Array` - 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
```

Remarque : *Contrairement au constructeur de classe Array, la fonction Array() n'utilise pas le mot-clé new.*

Voir aussi

Array

Boolean, fonction

```
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 zéro ; sinon, la valeur renvoyée est `false`.

Si *expression* est une chaîne, la valeur renvoyée est l'une des valeurs suivantes :

- Dans les fichiers publiés pour Flash Player 6 ou une version plus récente, la chaîne est tout d'abord convertie en nombre. Sa valeur est `true` si le nombre est différent de zéro, `false` dans le cas contraire.
- 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 *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`.

Remarque : *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 Lite 2.0

Paramètres

`expression:Object` - Expression à convertir en valeur booléenne.

Valeur renvoyée

`Boolean` - 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 booléenne, tandis que la classe Boolean crée un objet booléen. 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 aussi

`Boolean`

call, fonction

```
call(frame)
```

Déconseillé depuis Flash Player 5. Cette action est déconseillée au profit de 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'actions 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 Lite 1.0

Paramètres
`frame:Object` - Etiquette ou numéro d'une image dans le scénario.

Voir aussi
`Array, fonction`, `call (méthode Function.call)`

chr, fonction

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

Déconseillé depuis Flash Player 5. Cette fonction est déconseillée au profit de `String.fromCharCode()`.

Convertit les numéros de code ASCII en caractères.

Disponibilité
Flash Lite 1.0

Paramètres
`number:Number` - Numéro de code ASCII.

Valeur renvoyée
`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 aussi
`fromCharCode (méthode String.fromCharCode)`

clearInterval, fonction

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

Annule un intervalle créé par un appel à setInterval().

Disponibilité
Flash Lite 2.0

Paramètres
`intervalID:Number` - 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 de 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 aussi

`setInterval, fonction`

duplicateMovieClip, fonction

```
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. La fonction ou la méthode `removeMovieClip()` permet de supprimer une occurrence de clip créée avec `duplicateMovieClip()`.

Disponibilité

Flash Lite 2.0

Paramètres

`target:Object` - 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` - Identificateur unique du clip dupliqué.

`depth:Number` - 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

L'exemple suivant entraîne la création d'une occurrence de clip appelée `img_mc`. 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 aussi

removeMovieClip, fonction, duplicateMovieClip (méthode MovieClip.duplicateMovieClip), removeMovieClip (méthode MovieClip.removeMovieClip)

escape, fonction

```
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 Lite 2.0

Paramètres

`expression:String` - Expression à convertir en chaîne et à coder au format URL.

Valeur renvoyée

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

Voir aussi

unescape, fonction

eval, fonction

```
eval(expression:Object) : Objecteval(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 Lite 1.0

Paramètres
`expression:Object` - 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.)

Valeur renvoyée
`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 aussi
`Array`, `set variable, instruction`

fscommand, fonction

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

La fonction `fscommand()` permet à un fichier SWF de communiquer avec le lecteur Flash Lite ou l'environnement d'un périphérique mobile (tel qu'un système d'exploitation). Les paramètres définissent le nom de l'application en cours de démarrage et les paramètres correspondants, séparés par des virgules.

Commande	Paramètres	Rôle
`launch`	`application-path, arg1, arg2,..., argn`	Cette commande lance une autre application sur un périphérique mobile. Le nom de l'application et ses paramètres sont transmis à l'aide d'un argument unique. **Remarque** : Cette fonctionnalité dépend du système d'exploitation. Utilisez cette commande avec circonspection, dans la mesure où elle a recours au périphérique hôte pour exécuter une opération non prise en charge. Ceci risque d'entraîner un blocage sur ce périphérique. Cette commande est prise en charge uniquement lorsque le lecteur Flash Lite est en cours d'exécution en mode autonome. Elle n'est pas prise en charge lorsque le lecteur s'exécute dans le contexte d'une autre application (par exemple, en tant que module externe dans un navigateur).
`activateTextField`	`"" (ignoré)`	Cette commande active de façon asynchrone le champ texte sélectionné, ce qui autorise les modifications par l'utilisateur. En raison de son comportement asynchrone, cette commande est traitée à la fin de l'image. Le code ActionScript qui suit immédiatement l'appel à `fscommand()` s'exécute en premier. Si aucun texte n'est sélectionné lorsque la commande est traitée, rien ne se produit. Cette commande donne le focus a un champ texte qui a été transmis auparavant à la méthode `Selection.setFocus()` et active le champ texte en vue de sa modification. Cette commande a uniquement un effet lorsque le téléphone prend en charge la modification de texte en ligne. Cette commande peut être appelée dans le cadre d'un rappel de l'écouteur d'événements `Selection.onSetFocus()`. Les champs texte deviennent alors actifs et peuvent être modifiés lorsqu'ils sont sélectionnés. **Remarque** : dans la mesure où la fonction `fscommand()` s'exécute de façon asynchrone, le champ texte ne devient pas immédiatement actif ; il le devient à la fin de l'image.

Disponibilité

Flash Lite 1.1

Paramètres

`command:String` - Chaîne transmise à l'application hôte ou commande passée au lecteur Flash Lite.

`parameters:String` - Chaîne transmise à l'application hôte ou valeur passée au lecteur Flash Lite.

Exemple

Dans l'exemple suivant, la fonction `fscommand()` ouvre wap.yahoo.com sur le navigateur services/Web des téléphones de la gamme Series 60 :

```
on(keyPress "9") {
    status = fscommand("launch", "z:\\system\apps\browser\browser.app,http://wap.yahoo.com");
}
```

fscommand2, fonction

```
fscommand2(command:String, parameter1:String,...parameterN:String) : Void
```

Permet au fichier SWF de communiquer avec le lecteur Flash Lite ou une application hôte sur un périphérique mobile.

Pour utiliser `fscommand2()` afin d'envoyer un message au lecteur Flash Lite, vous devez utiliser les commandes et les paramètres prédéfinis. Consultez la section "Commandes fscommand2" de la rubrique « Eléments du langage ActionScript » pour prendre connaissance des commandes et paramètres pouvant être spécifiés pour la fonction `fscommand()`. Ces valeurs contrôlent les fichiers SWF lus par le lecteur Flash Player.

La fonction `fscommand2()` est similaire à la fonction `fscommand()`, à l'exception des différences suivantes :

- La fonction `fscommand2()` peut contenir autant d'arguments que nécessaire. Par opposition, `fscommand()` ne peut recevoir qu'un seul argument.
- Flash Lite exécute `fscommand2()` immédiatement (en d'autres termes, dans l'image) alors que `fscommand()` est exécuté à la fin du traitement de l'image.
- La fonction `fscommand2()` renvoie une valeur permettant de rapporter si la commande a réussi, échoué ou bien son résultat.

Remarque : Aucune des commandes `fscommand2()` n'est disponible dans les lecteurs Web.

Disponibilité

Flash Lite 1.1

Commandes fscommand2() déconseillées

Certaines commandes `fscommand2()` de Flash Lite 1.1 sont désormais déconseillées dans Flash Lite 2.0. Le tableau suivant répertorie les commandes `fscommand2()` remplacées :

Commande	Remplacée par
Escape	Fonction globale `escape`
GetDateDay	Méthode `getDate()` de l'objet Date
GetDateMonth	Méthode `getMonth()` de l'objet Date
GetDateWeekday	Méthode `getDay()` de l'objet Date
GetDateYear	Méthode `getYear()` de l'objet Date
GetLanguage	Propriété `System.capabilities.language`
GetLocaleLongDate	Méthode `getLocaleLongDate()` de l'objet Date
GetLocaleShortDate	Méthode `getLocaleShortDate()` de l'objet Date
GetLocaleTime	Méthode `getLocaleTime()` de l'objet Date
GetTimeHours	Méthode `getHours()` de l'objet Date
GetTimeMinutes	Méthode `getMinutes()` de l'objet Date
GetTimeSeconds	Méthode `getSeconds()` de l'objet Date
GetTimeZoneOffset	Méthode `getTimeZoneOffset()` de l'objet Date
SetQuality	`MovieClip._quality`
Unescape	Fonction globale `unescape()`

Paramètres

`command:String` - Chaîne transmise à l'application hôte ou commande passée au lecteur Flash Lite.

`parameters:String` - Chaîne transmise à l'application hôte ou valeur passée au lecteur Flash Lite.

getProperty, fonction

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

Déconseillé depuis Flash Player 5. La syntaxe à point, introduite dans Flash Player 5, est à préférer.

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

Disponibilité

Flash Lite 1.0

Paramètres

`my_mc:Object` - Nom d'occurrence d'un clip pour lequel la propriété est extraite.

`property:Object` - Propriété d'un clip.

Valeur renvoyée

`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 Sortie :

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

getTimer, fonction

```
getTimer() : Number
```

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

Disponibilité

Flash Lite 1.0

Valeur renvoyée

`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);
```

getURL, fonction

```
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.

Remarque : Cette fonction n'est pas prise en charge pour les périphériques BREW.

Disponibilité
Flash Lite 1.0

Paramètres
`url:String` - 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] - 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 permet d'envoyer des 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");
};
```

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 aussi

loadVariables, fonction, send (méthode XML.send), sendAndLoad (méthode XML.sendAndLoad)

getVersion, fonction

```
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 Lite 2.0

Valeur renvoyée

`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 aussi

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

gotoAndPlay, fonction

```
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 Lite 1.0

Paramètres

`scene:String` [facultatif] - Chaîne spécifiant le nom de la séquence cible de la tête de lecture.

`frame:Object` - Nombre représentant le numéro d'image ou 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 aussi

`gotoAndPlay (méthode MovieClip.gotoAndPlay)`, `nextFrame, fonction`, `play, fonction`, `prevFrame, fonction`

gotoAndStop, fonction

```
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 Lite 1.0

Paramètres

`scene:String` [facultatif] - Chaîne spécifiant le nom de la séquence cible de la tête de lecture.

`frame:Object` - Nombre représentant le numéro d'image ou 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 aussi

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

ifFrameLoaded, fonction

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

Déconseillé depuis Flash Player 5. Cette fonction est déconseillée. Adobe recommande d'employer 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 Lite 1.0

Paramètres

`scene:String` [facultatif] - Chaîne qui spécifie le nom de la séquence à charger.

`frame:Object` - Numéro ou étiquette d'image devant être chargés avant l'exécution de l'instruction suivante.

`statement(s):Object` - Instructions d'exécution si la séquence spécifiée, ou la séquence et l'image, sont chargées.

Voir aussi

addListener (méthode MovieClipLoader.addListener)

int, fonction

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

Déconseillé depuis Flash Player 5. Cette fonction est déconseillée au profit de `Math.round()`.

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

Disponibilité
Flash Lite 1.0

Paramètres
`value:Number` - Nombre devant être arrondi à un entier.

Valeur renvoyée
`Number` - Le nombre entier tronqué.

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

isFinite, fonction

```
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 Lite 2.0

Paramètres
`expression:Object` - Valeur booléenne, variable ou toute autre expression à évaluer.

Valeur renvoyée
`Boolean` - Valeur booléenne.

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

```
isFinite(56)
// returns true

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

isNaN, fonction

```
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 Lite 2.0

Paramètres
`expression:Object` - Valeur booléenne, variable ou toute autre expression à évaluer.

Valeur renvoyée

`Boolean` - 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 aussi

`NaN, constante`, `NaN (propriété Number.NaN)`

length, fonction

`length(expression)length(variable)`

Déconseillé depuis Flash Player 5. Cette fonction, de même que les fonctions de chaîne, est déconseillée. Adobe recommande d'employer 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 Lite 1.0

Paramètres

`expression:String` - Chaîne.

`variable:Object` - Nom d'une variable.

Valeur renvoyée

`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 aussi

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

loadMovie, fonction

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

Charge un fichier SWF ou JPEG dans Flash Player pendant la lecture du fichier SWF d'origine. Les fichiers JPEG enregistrés au format progressif ne sont pas pris en charge.

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()`.

Disponibilité

Flash Lite 1.1

Paramètres

`url:String` - 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` - Référence à un clip ou 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 est utilisée pour des 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 aussi

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

loadMovieNum, fonction

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

Charge un fichier SWF ou JPEG dans l'un des niveaux de Flash Player pendant la lecture du fichier SWF.

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 la cadence 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 les fichiers SWF chargés avec `loadMovieNum()`.

Disponibilité
Flash Lite 1.1

Paramètres
`url:String` - 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` - 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 est utilisée pour des 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 aussi
`unloadMovieNum, fonction`, `loadMovie, fonction`, `loadClip (méthode MovieClipLoader.loadClip)`, `_level, propriété`

loadVariables, fonction

```
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
```

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. 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, l'*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 inter-domaines* 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 inter-domaines » 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 Lite 1.1

Paramètres

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

`target:Object` - 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 est utilisée pour des 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 = setInterval(checkParamsLoaded, 100);
```

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

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

Voir aussi

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

loadVariablesNum, fonction

```
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, l'*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, l'*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 inter-domaines* sur le serveur hébergeant le fichier SWF. Pour plus d'informations, voir « A propos de l'autorisation de chargement de données inter-domaines » 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 Lite 1.1

Paramètres

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

`level:Number` - 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 est utilisée pour des 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 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 = setInterval(checkParamsLoaded, 100);

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

Voir aussi

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

mbchr, fonction

```
mbchr(number)
```

Déconseillé depuis Flash Player 5. Cette fonction est déconseillée au profit de la méthode `String.fromCharCode()`.

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

Disponibilité

Flash Lite 1.0

Paramètres

`number:Number` - Nombre à convertir en caractère multi-octets.

Voir aussi

fromCharCode (méthode String.fromCharCode)

mblength, fonction

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

Déconseillé depuis Flash Player 5. Cette fonction est déconseillée au profit de la propriété `String.length`.

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

Disponibilité
Flash Lite 1.0

Paramètres
`string:String` - Chaîne à mesurer.

Valeur renvoyée
`Number` - La longueur de la chaîne de caractères multi-octets.

Voir aussi
`String`, `length (propriété String.length)`

mbord, fonction

```
mbord(character) : Number
```

Déconseillé depuis Flash Player 5. Cette fonction est déconseillée au profit de la méthode `String.charCodeAt()`.

Convertit le caractère spécifié en nombre multi-octets.

Disponibilité
Flash Lite 1.0

Paramètres
`caractère:String` - Caractère à convertir en nombre multi-octets.

Valeur renvoyée
`Number` - Le caractère converti.

Voir aussi
`charCodeAt (méthode String.charCodeAt)`

mbsubstring, fonction

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

Déconseillé depuis Flash Player 5. Cette fonction est déconseillée au profit de la méthode `String.substr()`.

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

Disponibilité
Flash Lite 1.0

Paramètres
`value:String` - Chaîne multi-octets à partir de laquelle il convient d'extraire une nouvelle chaîne multi-octets.

`index:Number` - Numéro du premier caractère à extraire.

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

Valeur renvoyée

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

Voir aussi

`substr (méthode String.substr)`

nextFrame, fonction

```
nextFrame() : Void
```

Place la tête de lecture sur l'image suivante.

Disponibilité

Flash Lite 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 aussi

`prevFrame, fonction`

nextScene, fonction

```
nextScene() : Void
```

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

Disponibilité

Flash Lite 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 aussi

`prevScene, fonction`

Number, fonction

```
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`, la valeur renvoyée est la suivante :
- - Dans les fichiers publiés pour Flash Player 6 ou version précédente, le résultat est 0.
- - Dans les fichiers publiés pour Flash Player 7 ou version ultérieure, le résultat est `NaN`.

Disponibilité

Flash Lite 2.0

Paramètres

`expression:Object` - 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.

Valeur renvoyée

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

Exemple

Dans l'exemple suivant, un champ 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 aussi

NaN, constante, Number, parseInt, fonction, parseFloat, fonction

Object, fonction

```
Object( [value] ) : 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 Lite 2.0

Paramètres

`value:Object` [facultatif] - Valeur de type numérique, chaîne ou booléenne.

Valeur renvoyée

`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 aussi

Object

on, gestionnaire

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

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

Disponibilité

Flash Lite 2.0

Paramètres

`mouseEvent:Object` - *mouseEvent* est un déclencheur appelé *événement*. 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 pendant que le pointeur de la souris le survole.
- `releaseOutside` 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 et relâche le bouton de la souris. Les événements `press` et `dragOut` précèdent toujours l'événement `releaseOutside`. (Cet événement est pris en charge dans Flash Lite seulement si `System.capabilities.hasMouse` est `true` ou si `System.capabilities.hasStylus` est `true`.)
- `rollOut`Le pointeur quitte la zone du bouton. (Cet événement est pris en charge dans Flash Lite seulement si `System.capabilities.hasMouse` est `true` ou si `System.capabilities.hasStylus` est `true`.)
- `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. Pour la section *key* du paramètre, spécifiez une constante de touche, comme indiqué par le conseil de code dans le 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 aussi

`onClipEvent, gestionnaire`, `Key`

onClipEvent, gestionnaire

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

Déclenche les actions définies pour une instance spécifique de clip.

Disponibilité

Flash Lite 2.0

Paramètres

`movieEvent:Object` - *movieEvent* est un déclencheur appelé *événement*. 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. (Cet événement est pris en charge dans Flash Lite seulement si `System.capabilities.hasMouse` est `true`.)
- `mouseDown` L'action commence dès que l'utilisateur appuie sur le bouton gauche de la souris. (Cet événement est pris en charge dans Flash Lite seulement si `System.capabilities.hasMouse` est `true` ou si `System.capabilities.hasStylus` est `true`.)
- `mouseUp` L'action commence dès que l'utilisateur relâche le bouton gauche de la souris. (Cet événement est pris en charge dans Flash Lite seulement si `System.capabilities.hasMouse` est `true` ou si `System.capabilities.hasStylus` est `true`.)
- `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 et apparaissent dans le champ 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 aussi

`Key`, `_xmouse (propriété MovieClip._xmouse)`, `_ymouse (propriété MovieClip._ymouse)`, `constantes`

ord, fonction

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

Déconseillé depuis Flash Player 5. Cette fonction est déconseillée au profit des méthodes et des propriétés de la classe String.

Convertit les caractères en numéros de code ASCII.

Disponibilité

Flash Lite 1.0

Paramètres

`character:String` - Caractère à convertir en numéro de code ASCII.

Valeur renvoyée

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

Voir aussi

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

parseFloat, fonction

```
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 Lite 2.0

Paramètres

`string:String` - Chaîne à lire et convertir en nombre à virgule flottante.

Valeur renvoyée

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

Exemple

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

```
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 aussi

`NaN, constante`, `parseInt, fonction`

parseInt, fonction

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

Convertit une chaîne en entier. Si la chaîne spécifiée par les 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 Lite 2.0

Paramètres

`expression:String` - Chaîne à convertir en entier.

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

Valeur renvoyée

`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 aussi

`NaN, constante`, `parseFloat, fonction`

play, fonction

```
play() : Void
```

Fait avancer la tête de lecture au sein du scénario.

Disponibilité

Flash Lite 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 aussi

gotoAndPlay, fonction, gotoAndPlay (méthode MovieClip.gotoAndPlay)

prevFrame, fonction

```
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 Lite 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 aussi

nextFrame, fonction, prevFrame (méthode MovieClip.prevFrame)

prevScene, fonction

```
prevScene() : Void
```

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

Disponibilité

Flash Lite 1.0

Voir aussi

nextScene, fonction

random, fonction

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

Déconseillé depuis Flash Player 5. Cette fonction est déconseillée au profit de `Math.random()`.

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

Disponibilité
Flash Lite 1.1

Paramètres
`value:Number` - Entier.

Valeur renvoyée
`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 aussi
random (méthode Math.random)

removeMovieClip, fonction

`removeMovieClip(target:Object)`

Supprime le clip spécifié.

Disponibilité
Flash Lite 1.0

Paramètres
`target:Object` - Chemin cible d'une occurrence de clip créée à l'aide de la fonction `duplicateMovieClip()` ou nom de l'occurrence d'un clip créé à l'aide de la fonction `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 aussi
duplicateMovieClip, fonction, duplicateMovieClip (méthode MovieClip.duplicateMovieClip), attachMovie (méthode MovieClip.attachMovie), removeMovieClip (méthode MovieClip.removeMovieClip) createEmptyMovieClip (méthode MovieClip.createEmptyMovieClip)

setInterval, fonction

```
setInterval(functionName:Object, interval:Number [, param1:Object, param2, ..., paramN]) :
Number
setInterval(objectName:Object, methodName:String, interval:Number [, param1:Object, param2,
..., paramN]) : Number
```

Appelle une fonction ou une méthode ou un objet à des intervalles périodiques pendant la lecture d'un fichier SWF. Vous pouvez utiliser une fonction interval pour mettre à jour les variables en fonction d'une base de données ou pour mettre à jour l'heure affichée.

Si *interval* est supérieur à la cadence d'images du fichier SWF, la fonction interval n'est appelée que lorsque la tête de lecture est placée sur une image, ce qui réduit l'impact des actualisations d'écran.

Remarque : Dans Flash Lite 2.0, l'intervalle transmis par cette méthode est ignoré si sa valeur est inférieure à la cadence d'images du fichier SWF et si la fonction interval est appelée uniquement pour la cadence d'images du fichier SWF. Si l'intervalle est supérieur à la cadence d'images du fichier SWF, l'événement est appelé sur l'image suite, à l'issue de l'intervalle.

Disponibilité
Flash Lite 2.0

Paramètres
`functionName:Object` - Nom de fonction ou référence à une fonction anonyme.

`interval:Number` - Nombre de millisecondes séparant les appels du paramètre *functionName* ou *methodName*.

`param:Object` [facultatif] - Paramètres transmis au paramètre *functionName* ou *methodName*. Les paramètres multiples doivent être séparés par des virgules : *param1*, *param2*, ..., *paramN*.

`objectName:Object` - Objet contenant la méthode *methodName*.

`methodName:String` - Méthode d'*objectName*.

Valeur renvoyée
`Number` - Un entier d'identification que vous pouvez transmettre à `clearInterval()` pour annuler l'intervalle.

Exemple
Utilisation 1 : L'exemple suivant appelle une fonction anonyme toutes les 1 000 millisecondes (1 seconde).

```
setInterval( function(){ trace("interval called"); }, 1000 );
```

Utilisation 2 : L'exemple suivant définit deux gestionnaires d'événements et les appelle. Le premier appel à `setInterval()` appelle la fonction `callback1()`, qui contient une instruction `trace()`. Le deuxième appel à `setInterval()` transmet la chaîne `"interval called"` à la fonction `callback2()` en tant que paramètre.

```
function callback1() {
    trace("interval called");
}

function callback2(arg) {
    trace(arg);
}

setInterval( callback1, 1000 );
setInterval( callback2, 1000, "interval called" );
```

Utilisation 3 : cet exemple a recours à la méthode d'un objet. Vous devez utiliser cette syntaxe lorsque vous devez appeler une méthode qui est définie pour un objet.

```
obj = new Object();
obj.interval = function() {
    trace("interval function called");
}

setInterval( obj, "interval", 1000 );

obj2 = new Object();
obj2.interval = function(s) {
    trace(s);
}
setInterval( obj2, "interval", 1000, "interval function called" );
```

Vous devez utiliser la deuxième forme de la syntaxe `setInterval()` pour appeler la méthode d'un objet, comme indiqué dans l'exemple suivant :

```
setInterval( obj2, "interval", 1000, "interval function called" );
```

Lorsque vous utilisez cette fonction, vous devez porter une attention particulière à la mémoire utilisée par le fichier SWF. Par exemple, le fait de supprimer un clip d'un fichier SWF, n'annule pas la fonction `setInterval()` au sein de ce dernier. Annulez systématiquement la fonction `setInterval()` à l'aide de `clearInterval()` lorsque vous n'avez plus besoin de cette dernière, comme indiqué dans l'exemple suivant :

```
// create an event listener object for our MovieClipLoader instance
var listenerObjectbject = new Object();
listenerObject.onLoadInit = function(target_mc:MovieClip) {
    trace("start interval");
    /* after the target movie clip loaded, create a callback which executes
    about every 1000 ms (1 second) and calls the intervalFunc function. */
    target_mc.myInterval = setInterval(intervalFunc, 1000, target_mc);
};
function intervalFunc(target_mc) {
    // display a trivial message which displays the instance name and arbitrary text.
    trace(target_mc+" has been loaded for "+getTimer()/1000+" seconds.");
    /* when the target movie clip is clicked (and released) you clear the interval
    and remove the movie clip. If you don't clear the interval before deleting
    the movie clip, the function still calls itself every second even though the
    movie clip instance is no longer present. */
    target_mc.onRelease = function() {
    trace("clear interval");
    clearInterval(this.myInterval);
    // delete the target movie clip
    removeMovieClip(this);
    };
}
var jpeg_mcl:MovieClipLoader = new MovieClipLoader();
jpeg_mcl.addListener(listenerObject);
jpeg_mcl.loadClip("http://www.helpexamples.com/flash/images/image1.jpg",
    this.createEmptyMovieClip("jpeg_mc", this.getNextHighestDepth()));
```

Si vous utilisez `setInterval()` dans des classes, vous devez utiliser `ce` mot clé lorsque vous appelez la fonction. Ce mot clé est indispensable pour que la fonction `setInterval()` accède aux membres des classes. Ceci est illustré par l'exemple suivant. Lorsque le fichier FLA contient un bouton `deleteUser_btn`, ajoutez le code ActionScript à l'image 1 :

```
var me:User = new User("Gary");
this.deleteUser_btn.onRelease = function() {
    trace("Goodbye, "+me.username);
    clearInterval(me.intervalID);
    delete me;
};
```

Créez ensuite le fichier User.as dans le répertoire du fichier FLA. Saisissez le code ActionScript suivant :

```
class User {
    var intervalID:Number;
    var username:String;
    function User(param_username:String) {
    trace("Welcome, "+param_username);
    this.username = param_username;
    this.intervalID = setInterval(this, "traceUsername", 1000, this.username);
    }
    function traceUsername(str:String) {
    trace(this.username+" is "+getTimer()/1000+" seconds old, happy birthday.");
    }
}
```

Voir aussi

clearInterval, fonction

setProperty, fonction

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

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

Disponibilité

Flash Lite 1.0

Paramètres

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

`property:Object` - Propriété à définir.

`expression:Object` - Nouvelle valeur littérale de la propriété ou 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 aussi

getProperty, fonction

startDrag, fonction

```
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.

Remarque : Cette méthode n'est prise en charge par Flash Lite que si `System.capabilities.hasMouse` est défini sur `true` ou si `System.capabilities.hasStylus` est défini sur `true`.

Disponibilité

Flash Lite 2.0

Paramètres

`target:Object` - Chemin cible du clip à faire glisser.

`lock: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,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 aussi

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

stop, fonction

```
stop() : Void
```

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

Disponibilité
Flash Lite 1.0

Voir aussi
gotoAndStop, fonction, gotoAndStop (méthode MovieClip.gotoAndStop)

stopAllSounds, fonction

```
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 Lite 1.0

Exemple
Le code suivant crée un champ 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 aussi
Sound

stopDrag, fonction

```
stopDrag() : Void
```

Arrête l'opération de déplacement en cours.

Remarque : Cette méthode n'est prise en charge par Flash Lite que si `System.capabilities.hasMouse` est défini sur `true` ou si `System.capabilities.hasStylus` est défini sur `true`.

Disponibilité
Flash Lite 2.0

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 aussi
`startDrag, fonction`, `_droptarget (MovieClip._droptarget, propriété)`, `startDrag (méthode MovieClip.startDrag)` `stopDrag (méthode MovieClip.stopDrag)`

String, fonction

```
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 valeur renvoyée est `"true"` ou `"false"`.
- Si *expression* est un clip, la valeur renvoyée est le chemin cible du clip utilisant la notation à barre oblique (/).

Si *expression* est `undefined`, les valeurs renvoyées sont les 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 Lite 1.0

Paramètres
`expression:Object` - Expression à convertir en chaîne.

Valeur renvoyée
`String` - 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 aussi

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

substring, fonction

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

Déconseillé depuis Flash Player 5. Cette fonction est déconseillée au profit de `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 Lite 1.0

Paramètres

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

`index:Number` - Numéro du premier caractère à extraire.

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

Valeur renvoyée

`String` - La sous-chaîne extraite.

Voir aussi

substr (méthode String.substr)

targetPath, fonction

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

Renvoie une chaîne contenant le chemin cible d'un objet MovieClip, Button ou TextField. 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 Lite 2.0

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.

Valeur renvoyée

`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 aussi

`eval, fonction`

tellTarget, fonction

```
tellTarget("target") { statement(s); }
```

Déconseillé depuis Flash Player 5. Adobe recommande d'utiliser une notation de type point (.) et l'instruction `with`.

Cette fonction applique les instructions spécifiées dans le paramètre *statements* au scénario spécifié par 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 obliger les clips à atteindre une image spécifique.

A partir de Flash 5 ou d'une 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 Lite 1.0

Paramètres

`target:String` - Chaîne qui spécifie le chemin cible du scénario à contrôler.

`statement(s):Object` - Instructions à exécuter lorsque la condition renvoie la valeur `true`.

Exemple

Cette instruction tellTarget contrôle la balle de l'occurrence de clip sur le scénario principal. L'image 1 de l'occurrence ball 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 ball, 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 aussi

`with, instruction`

toggleHighQuality, fonction

`toggleHighQuality()`

Déconseillé depuis Flash Player 5. Cette fonction est déconseillée au profit de `_quality`.

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

Disponibilité

Flash Lite 1.0

Exemple

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

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

Voir aussi

`_highquality, propriété`, `_quality, propriété`

trace, fonction

`trace(expression:Object)`

Vous pouvez utiliser Flash Debug Player pour capturer les sorties de la fonction `trace()` et les écrire dans le fichier journal.

Instruction ; évalue l'expression et affiche les résultats dans le panneau Sortie en mode de test.

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 Sortie. L'instruction `trace()` est similaire à la fonction JavaScript `alert`.

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 Lite 1.0

Paramètres
`expression:Object` - 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 Sortie.

Exemple
L'exemple suivant utilise une instruction `trace()` pour afficher dans le panneau Sortie les méthodes et propriétés du champ 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
*/
```

unescape, fonction

```
unescape(x: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 cette chaîne.

Disponibilité
Flash Lite 1.1

Paramètres
`string:String` - Chaîne comportant des séquences d'échappement hexadécimales.

Valeur renvoyée
`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 Sortie.

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

unloadMovie, fonction

```
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 Lite 1.1

Paramètres

`target` - 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 aussi

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

unloadMovieNum, fonction

```
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 Lite 1.1

Paramètres

`level:Number` - 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 aussi

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

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
	$version	**Déconseillé** depuis Flash Lite Player 2.0. Cette action est déconseillée au profit de la propriété `System.capabilities.version`. Contient le numéro de version de Flash Lite.
	_cap4WayKeyAS	**Déconseillé** depuis Flash Lite Player 2.0. Cette action est déconseillée au profit de la propriété `System.capabilities.has4WayKeyAS`. Indique si Flash Lite exécute les expressions ActionScript liées aux gestionnaires d'événements clés associés aux touches Droite, Gauche, Haut et Bas.
	_capCompoundSound	**Déconseillé** depuis Flash Lite Player 2.0. Cette action est déconseillée au profit de la propriété `System.capabilities.hasCompoundSound`. Indique si Flash Lite peut traiter les données sons composites.

Modificateurs	Propriété	Description
	_capEmail	**Déconseillé** depuis Flash Lite Player 2.0. Cette action est déconseillée au profit de la propriété `System.capabilities.hasEmail`. Indique si le client Flash Lite peut envoyer des messages électroniques à l'aide de la commande ActionScript `GetURL()`.
	_capLoadData	**Déconseillé** depuis Flash Lite Player 2.0. Cette action est déconseillée au profit de la propriété `System.capabilities.hasDataLoading`. Indique si l'application hôte peut charger de façon dynamique d'autres données via des appels aux fonctions `loadMovie()`, `loadMovieNum()`, `loadVariables()` et `loadVariablesNum()`.
	_capMFi	**Déconseillé** depuis Flash Lite Player 2.0. Cette action est déconseillée au profit de la propriété `System.capabilities.hasMFi`. Indique si le périphérique peut lire des données son au format audio Melody pour i-mode (MFi).
	_capMIDI	**Déconseillé** depuis Flash Lite Player 2.0. Cette action est déconseillée au profit de la propriété `System.capabilities.hasMIDI`. Indique si le périphérique peut diffuser des sons au format MIDI (Musical Instrument Digital Interface).
	_capMMS	**Déconseillé** depuis Flash Lite Player 2.0. Cette action est déconseillée au profit de la propriété `System.capabilities.hasMMS`. Indique si Flash Lite peut envoyer des messages MMS (Multimedia Messaging Service) à l'aide de la commande ActionScript `GetURL()`. Dans l'affirmative, cette variable est définie et prend la valeur 1. Sinon, elle reste non définie.
	_capSMAF	**Déconseillé** depuis Flash Lite Player 2.0. Cette action est déconseillée au profit de la propriété `System.capabilities.hasSMAF`. Indique si le périphérique peut diffuser des fichiers multimédias au format SMAF (Synthetic music Mobile Application Format). Dans l'affirmative, cette variable est définie et prend la valeur 1. Sinon, elle reste non définie.
	_capSMS	**Déconseillé** depuis Flash Lite Player 2.0. Cette action est déconseillée au profit de la propriété `System.capabilities.hasSMS`. Indique si Flash Lite peut envoyer des messages SMS (Short Message Service) à l'aide de la commande ActionScript `GetURL()`.
	_capStreamSound	**Déconseillé** depuis Flash Lite Player 2.0. Cette action est déconseillée au profit de la propriété `System.capabilities.hasStreamingAudio`. Indique si le périphérique peut diffuser des sons en flux continu (synchronisés).
	_focusrect	Propriété (globale) ; spécifie si un rectangle jaune doit s'afficher autour du bouton ou du clip qui a le focus du clavier.

Modificateurs	Propriété	Description
	_forceframerate	Demande au lecteur Flash Lite de procéder au rendu à la cadence d'images spécifiée.
	_global	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 est déconseillée au profit de `_quality`. Spécifie le niveau d'anti-aliasing appliqué au fichier SWF actuel.
	_level	Référence au scénario racine de `_level`*N*.
	maxscroll	**Déconseillé depuis Flash Player 5.** Cette propriété est déconseillée au profit de `TextField.maxscroll`. Indique le numéro de ligne de la première ligne de texte visible dans un champ 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.
	scroll	**Déconseillé depuis Flash Player 5.** Cette propriété est déconseillée au profit de `TextField.scroll`. Contrôle l'affichage des informations dans un champ texte associé à une variable.
	_soundbuftime	Etablit 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.

$version, propriété

`$version`

Déconseillé depuis Flash Lite Player 2.0. Cette action est déconseillée au profit de la propriété `System.capabilities.version`.

Variable String ; contient le numéro de version de Flash Lite. Cette variable indique les numéros majeur et secondaire et le numéro de création interne, qui correspond généralement à `0` dans les versions officielles. Le numéro majeur signalé pour tous les produits Flash Lite 1.x est 5. Flash Lite 1.0 dispose d'un numéro mineur, 1. Flash Lite 1.1 a 2 pour numéro mineur.

Disponibilité

Flash Lite 1.1

Exemple

Dans le lecteur Flash Lite 1.1, le code suivant donne la valeur « 5, 2, 12, 0 » à `myVersion` :

```
myVersion = $version;
```

Voir aussi

`version (propriété capabilities.version)`

_cap4WayKeyAS, propriété

`_cap4WayKeyAS`

Déconseillé depuis Flash Lite Player 2.0. Cette action est déconseillée au profit de la propriété `System.capabilities.has4WayKeyAS`.

Variable numérique ; indique si Flash Lite exécute les expressions ActionScript liées aux gestionnaires d'événements clés associés aux touches Droite, Gauche, Haut et Bas. Cette variable est définie et n'a la valeur 1 que lorsque l'application hôte applique le mode quadridirectionnel de navigation par touches pour parcourir les contrôles Flash (boutons et champs de saisie). Dans le cas contraire, cette variable n'est pas définie.

Lorsque l'une des quatre touches de direction est sollicitée, si la valeur de la variable `_cap4WayKeyAS` est `1`, Flash Lite recherche le gestionnaire de cette touche en premier. S'il n'en trouve pas, l'application parcourt les contrôles Flash. Cependant, si un gestionnaire d'événements est trouvé, aucune action de navigation ne se produit pour cette touche. Par exemple, si un gestionnaire associé à la touche Bas est détecté, l'utilisateur ne peut pas procéder à la navigation.

Disponibilité
Flash Lite 1.1

Exemple
L'exemple suivant définit `canUse4Way` sur `1` dans Flash Lite 1.1, mais laisse cette variable non définie dans Flash Lite 1.0 (néanmoins, certains téléphones Flash Lite 1.1 ne prennent pas en charge les touches quadridirectionnelles, par conséquent le code exact dépend du téléphone visé) :

```
canUse4Way = _cap4WayKeyAS;
    if (canUse4Way == 1) {
    msg = "Use your directional joypad to navigate this application";
    } else {
    msg = "Please use the 2 key to scroll up, the 6 key to scroll right,

the 8 key to scroll down, and the 4 key to scroll left.";
    }
```

Voir aussi
`capabilities (System.capabilities)`

_capCompoundSound, propriété

`_capCompoundSound`

Déconseillé depuis Flash Lite Player 2.0. Cette action est déconseillée au profit de la propriété `System.capabilities.hasCompoundSound`.

Variable numérique ; indique si Flash Lite peut traiter les données sons composites. Dans l'affirmative, cette variable est définie et prend la valeur 1. Sinon, elle reste non définie. Par exemple, un fichier Flash peut contenir le même son aux formats MIDI et MFi. Le lecteur lit ensuite les données au format requis en fonction du format pris en charge par le périphérique. Cette variable définit si le lecteur Flash Lite prend en charge cette fonctionnalité sur le combiné en cours d'utilisation.

Disponibilité
Flash Lite 1.1

Exemple

Dans l'exemple suivant, `useCompoundSound` est défini sur `1` dans Flash Lite 1.1, mais reste non défini dans Flash Lite 1.0 :

```
useCompoundSound = _capCompoundSound;

if (useCompoundSound == 1) {
    gotoAndPlay("withSound");
} else {
    gotoAndPlay("withoutSound");
```

Voir aussi

capabilities (System.capabilities)

_capEmail, propriété

`_capEmail`

Déconseillé depuis Flash Lite Player 2.0. Cette action est déconseillée au profit de la propriété `System.capabilities.hasEmail`.

Variable numérique ; indique si le client Flash Lite peut envoyer des messages électroniques à l'aide de la commande ActionScript `GetURL()`. Dans l'affirmative, cette variable est définie et prend la valeur 1. Sinon, elle reste non définie.

Disponibilité

Flash Lite 1.1

Exemple

Si l'application hôte peut envoyer des messages électroniques à l'aide de la commande ActionScript `GetURL()`, l'exemple suivant définit `canEmail()` sur `1`:

```
canEmail = _capEmail;

if (canEmail == 1) {
    getURL("mailto:someone@somewhere.com?subject=foo&body=bar");
}
```

Voir aussi

capabilities (System.capabilities)

_capLoadData, propriété

`_capLoadData`

Déconseillé depuis Flash Lite Player 2.0. Cette action est déconseillée au profit de la propriété `System.capabilities.hasDataLoading`.

Variable numérique ; indique si l'application hôte peut charger de façon dynamique des données supplémentaires en appelant les fonctions `loadMovie()`, `loadMovieNum()`, `loadVariables()` et `loadVariablesNum()`. Le cas échéant, cette variable est définie et sa valeur est `1`; dans le cas contraire, cette variable n'est pas définie.

Disponibilité

Flash Lite 1.1

Exemple

Si l'application hôte peut effectuer un chargement dynamique des animations et des variables, l'exemple suivant définit `CanLoad` sur `1` :

```
canLoad = _capLoadData;

if (canLoad == 1) {
    loadVariables("http://www.somewhere.com/myVars.php", GET);
} else {
    trace ("client does not support loading dynamic data");
}
```

Voir aussi

capabilities (System.capabilities)

_capMFi, propriété

`_capMFi`

Déconseillé depuis Flash Lite Player 2.0. Cette action est déconseillée au profit de la propriété `System.capabilities.hasMFi`.

Variable numérique ; indique si le périphérique peut lire des données son au format audio Melody pour i-mode (MFi). Dans l'affirmative, cette variable est définie et prend la valeur 1. Sinon, elle reste non définie.

Disponibilité

Flash Lite 1.1

Exemple

Si le périphérique peut lire des données audio MFi, l'exemple suivant définit `canMfi` sur `1`:

```
canMFi = _capMFi;

if (canMFi == 1) {
    // send movieclip buttons to frame with buttons that trigger events

sounds
    tellTarget("buttons") {
    gotoAndPlay(2);
    }
}
```

Voir aussi

hasMFI (propriété capabilities.hasMFI)

_capMIDI, propriété

`_capMIDI`

Déconseillé depuis Flash Lite Player 2.0. Cette action est déconseillée au profit de la propriété `System.capabilities.hasMIDI`.

Variable numérique ; indique si le périphérique peut diffuser des sons au format MIDI (Musical Instrument Digital Interface). Dans l'affirmative, cette variable est définie et prend la valeur 1. Sinon, elle reste non définie.

Disponibilité
Flash Lite 1.1

Exemple
Si le périphérique peut lire des données audio MFi, l'exemple suivant définit `_capMIDI` sur `1`:

```
canMIDI = _capMIDI;

if (canMIDI == 1) {
    // send movieclip buttons to frame with buttons that trigger events

sounds
    tellTarget("buttons") {
    gotoAndPlay(2);
    }
}
```

Voir aussi
capabilities (System.capabilities)

_capMMS, propriété

`_capMMS`

Déconseillé depuis Flash Lite Player 2.0. Cette action est déconseillée au profit de la propriété `System.capabilities.hasMMS`.

Variable numérique ; indique si Flash Lite peut envoyer des messages MMS (Multimedia Messaging Service) à l'aide de la commande ActionScript `GetURL()`. Dans l'affirmative, cette variable est définie et prend la valeur 1. Sinon, elle reste non définie.

Disponibilité
Flash Lite 1.1

Exemple
L'exemple suivant définit `canMMS` sur `1` dans Flash Lite 1.1, mais laisse cette variable non définie dans Flash Lite 1.0 (néanmoins, certains téléphones Flash Lite 1.1 ne prennent pas en charge l'envoi de messages MMS, par conséquent le code exact dépend du téléphone visé) :

```
on(release) {
    canMMS = _capMMS;
    if (canMMS == 1) {
    // send an MMS
    myMessage = "mms:4156095555?body=sample mms message";
    } else {
    // send an SMS
    myMessage = "sms:4156095555?body=sample sms message";
    }
    getURL(myMessage);
}
```

Voir aussi
capabilities (System.capabilities)

_capSMAF, propriété

`_capSMAF`

Déconseillé depuis Flash Lite Player 2.0. Cette action est déconseillée au profit de la propriété `System.capabilities.hasSMAF`.

Variable numérique, indique si le périphérique peut diffuser des fichiers multimédias au format SMAF (Synthetic music Mobile Application Format). Dans l'affirmative, cette variable est définie et prend la valeur 1. Sinon, elle reste non définie.

Disponibilité

Flash Lite 1.1

Exemple

L'exemple suivant définit `canSMAF` sur `1` dans Flash Lite 1.1, mais laisse cette variable non définie dans Flash Lite 1.0 (néanmoins, certains téléphones Flash Lite 1.1 ne prennent pas en charge l'envoi de messages SMAF, par conséquent le code exact dépend du téléphone visé) :

```
canSMAF = _capSMAF;

if (canSMAF) {
    // send movieclip buttons to frame with buttons that trigger events

sounds
    tellTarget("buttons") {
    gotoAndPlay(2);
    }
}
```

Voir aussi

`capabilities (System.capabilities)`

_capSMS, propriété

`_capSMS`

Déconseillé depuis Flash Lite Player 2.0. Cette action est déconseillée au profit de la propriété `System.capabilities.hasSMS`.

Variable numérique ; indique si Flash Lite peut envoyer des messages SMS (Short Message Service) à l'aide de la commande ActionScript `GetURL()`. Dans l'affirmative, cette variable est définie et prend la valeur 1. Sinon, elle reste non définie.

Disponibilité

Flash Lite 1.1

Exemple

L'exemple suivant définit `canSMS` sur `1` dans Flash Lite 1.1, mais laisse cette variable non définie dans Flash Lite 1.0 (néanmoins, certains téléphones Flash Lite 1.1 ne prennent pas en charge l'envoi de messages SMAF, par conséquent le code exact dépend du téléphone visé) :

```
on(release) {
    canSMS = _capSMS;
    if (canSMS) {
    // send an SMS
    myMessage = "sms:4156095555?body=sample sms message";
    getURL(myMessage);
    }
}
```

Voir aussi

capabilities (System.capabilities)

_capStreamSound, propriété

```
_capStreamSound
```

Déconseillé depuis Flash Lite Player 2.0. Cette action est déconseillée au profit de la propriété `System.capabilities.hasStreamingAudio`.

Variable numérique, indique si le périphérique peut diffuser des sons en flux continu (synchronisés). Dans l'affirmative, cette variable est définie et prend la valeur 1. Sinon, elle reste non définie.

Disponibilité

Flash Lite 1.1

Exemple

L'exemple suivant lit du son en continu si `canStreamSound` est activée :

```
on(press) {
    canStreamSound = _capStreamSound;
    if (canStreamSound) {
    // play a streaming sound in a movieclip with this button
    tellTarget("music") {
    gotoAndPlay(2);
    }
    }
    }
```

Voir aussi

capabilities (System.capabilities)

_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`, 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` si vous ne souhaitez pas afficher ce rectangle jaune. Cette propriété peut être remplacée pour des occurrences spécifiques.

Si la propriété `_focusrect` a la valeur `false`, 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`. Pour restaurer toutes les fonctionnalités de clavier d'un bouton ou d'un clip spécifique, vous pouvez annuler cette propriété globale à l'aide de `Button._focusrect` ou `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.

Remarque : Pour le lecteur Flash Lite 2.0, lorsque la propriété `_focusrect` est désactivée (par exemple `Button.focusRect = false` ou `MovieClip.focusRect = false`), le bouton ou le clip reçoit toujours l'ensemble des événements. Ce comportement est différent du lecteur Flash, car lorsque la propriété `_focusrect` est désactivée, le bouton ou le clip reçoit les événements `rollOver` et `rollOut`, mais pas `press` et `release`.

D'autre part, pour Flash Lite 2.0, vous pouvez modifier la couleur du rectangle de focus à l'aide de la commande fscommand2 `SetFocusRectColor`. Ce comportement diffère de Flash Player, où la couleur du rectangle de focus est limitée au jaune.

Disponibilité
Flash Lite 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;
```

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

_forceframerate, propriété

`_forceframerate`

Si cette propriété est définie sur `true`, elle demande au lecteur Flash Lite de procéder au rendu à la cadence d'images spécifiée. Vous pouvez utiliser cette propriété pour les sons pseudo-synchronisés lorsque le contenu inclut le son du périphérique. Elle est définie sur `false` par défaut, ce qui oblige Flash Lite à procéder au rendu de façon normale. Lorsqu'elle est définie sur `true`, le lecteur Flash Lite risque de ne pas rendre certaines images pour préserver la cadence.

Disponibilité
Flash Lite 2.0

_global, propriété

`_global.identifier`

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, à condition qu'elles ne soient pas masquées par des identificateurs portant le même nom dans les domaines internes.

Remarque : Lorsque vous définissez une variable globale, vous devez utiliser le nom entièrement qualifié de la variable, par exemple _global.variableName. Le non-respect de cette règle crée une variable locale du même nom qui masquera la variable globale que vous essayez de définir.

Disponibilité
Flash Lite 2.0

Valeur renvoyée
Référence à l'objet global qui contient les principales classes ActionScript, telles que String, Object, Math et Array.

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&lt;=1) {
    return 1;
    } else {
    return n*factorial(n-1);
    }
}
// Note: factorial 4 == 4*3*2*1 == 24
trace(factorial(4)); // output: 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 = "global";
trace("_global.myVar: " + _global.myVar); // _global.myVar: global
trace("myVar: " + myVar); // myVar: global

myVar = "local";
trace("_global.myVar: " + _global.myVar); // _global.myVar: global
trace("myVar: " + myVar); // myVar: local
```

Voir aussi
`set variable, instruction`

_highquality, propriété

`_highquality`

Déconseillé depuis Flash Player 5. Cette propriété est déconseillée au profit de `_quality`.

Spécifie le niveau d'anti-aliasing appliqué au fichier SWF actuel. Spécifiez 2 (qualité supérieure) pour appliquer la meilleure qualité. Spécifiez 1 (haute qualité) pour appliquer le lissage. Spécifiez 0 (faible qualité) pour empêcher l'anti-aliasing.

Disponibilité
Flash Lite 1.0

Exemple
Le code ActionScript suivant est placé sur le scénario principal, puis définit la propriété de qualité globale de façon à appliquer le lissage (anti-aliasing). `_highquality = 1;`

Voir aussi
`_quality, propriété`

_level, propriété

`_level`*N*

Référence au scénario racine de `_level`*N*. Vous devez utiliser `loadMovieNum()` pour charger des fichiers SWF dans Flash Player avant d'utiliser la propriété `_level` pour les cibler. Vous pouvez également utiliser `_level`*N* pour cibler un fichier SWF chargé 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 Lite 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 aussi
`loadMovie, fonction`, `swapDepths (méthode MovieClip.swapDepths)`

maxscroll, propriété

variable_name`.maxscroll`

Déconseillé depuis Flash Player 5. Cette propriété est déconseillée au profit de `TextField.maxscroll`.

Indique le numéro de ligne de la première ligne de texte visible dans un champ 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 texte. Cette propriété peut être récupérée, mais pas modifiée.

Disponibilité
Flash Lite 1.1

Voir aussi
`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 Lite 2.0

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 (à l'aide de `_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 aussi
`_root, propriété`, `targetPath, fonction`

_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 sur les valeurs suivantes :

Valeur	Description	Anti-aliasing des graphiques	Lissage des bitmaps
"LOW"	Qualité de rendu inférieure.	Les graphiques ne sont pas anti-aliasés.	Les bitmaps ne sont pas lissés
"MEDIUM"	Qualité de rendu moyenne. Ce niveau de qualité convient aux animations qui ne contiennent pas de texte.	Les graphiques sont anti-aliasés en utilisant une grille de 2 x 2 pixels.	Les bitmaps ne sont pas lissés
"HIGH"	Qualité de rendu supérieure. Il s'agit du paramètre de qualité de rendu par défaut de Flash.	Les graphiques sont anti-aliasés en utilisant une grille de 4 x 4 pixels.	Les bitmaps ne sont pas lissés

Disponibilité
Flash Lite 2.0

Exemple
L'exemple suivant définit la qualité de 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 Lite 2.0

Paramètres
movieClip : String - Nom d'occurrence d'un clip.

action : String - Action ou champ.

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 aussi

_lockroot (propriété MovieClip._lockroot), _parent, propriété, targetPath, fonction

scroll, propriété

textFieldVariableName.scroll = *x*

Déconseillé depuis Flash Player 5. Cette propriété est déconseillée au profit de `TextField.scroll`.

Contrôle l'affichage des informations dans un champ texte associé à une variable. La propriété `scroll` définit l'emplacement à partir duquel le champ texte commence à afficher le contenu ; une fois l'emplacement défini, Flash Player le met à jour lorsque l'utilisateur fait défiler le champ 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 Lite 1.1

Exemple

Le code suivant est associé à un bouton Vers le haut qui fait défiler le champ texte intitulé myText :

```
on (release) {
myText.scroll = myText.scroll + 1;
}
```

Voir aussi

maxscroll (propriété TextField.maxscroll), scroll (propriété TextField.scroll)

_soundbuftime, propriété

_soundbuftime:Number = integer

Etablit le nombre de secondes de son en diffusion continue à placer en mémoire tampon. La valeur par défaut est de 5 secondes.

Disponibilité

Flash Lite 2.0

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 avant 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 afin de 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'un champ est appelé, `this` contient une référence à l'objet qui contient le champ appelé.

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. Crée ApplyThis.as et entre 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 AS, ajoutez le code ActionScript 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 un fichier FLA ou AS, ajoutez le code ActionScript suivant :

```
var obj:Simple = new Simple();
obj.num = 0;
obj.func = function() {
return true;
};
obj.callfunc();
// output: true
```

Vous risquez d'obtenir une erreur de syntaxe lorsque vous utilisez une version incorrecte de Simple.as.

Disponibilité

Flash Lite 2.0

Exemple

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 aussi

constantes, onClipEvent, gestionnaire

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.
+= (affectation d'addition)	Affecte à *expression1* la valeur de ***expression1*** + ***expression2***.
[] (opérateur d'accès au tableau)	Initialise un nouveau tableau ou tableau multidimensionnel avec les éléments spécifiés (*a0*, etc.) ou accède aux éléments dans un tableau.
= (affectation)	Affecte la valeur de ***expression2*** (le paramètre de droite) à la variable, à l'élément de tableau ou à la propriété dans *expression1*.
& (AND au niveau du bit)	Convertit ***expression1 and expression2*** en entiers 32 bits non signés et applique une opération booléenne AND sur chaque bit des entiers entrés en tant que paramètres.
&= (affectation AND au niveau du bit)	Affecte à ***expression1*** la valeur de ***expression1& expression2***.
<< (décalage gauche au niveau du bit)	Convertit ***expression1*** et ***expression2*** en entiers 32 bits et décale tous les bits de ***expression1*** vers la gauche du nombre d'unités spécifié par l'entier résultant de la conversion de ***expression2***.
<<= (décalage gauche au niveau du bit et affectation)	Cet opérateur effectue un décalage vers la gauche au niveau du bit (<<) et stocke ensuite le contenu dans ***expression1***.
~ (NOT au niveau du bit)	Connu également sous la forme de complément d'opérateur du un ou opérateur de complément au niveau du bit.
| (OR au niveau du bit)	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.
|= (affectation OR au niveau du bit)	Affecte à *expression1* la valeur de *expression1* | *expression2*.
>> (décalage droit au niveau du bit)	Convertit *expression1* et *expression2* en entiers 32 bits et décale tous les bits de *expression1* vers la droite du nombre d'unités spécifié par l'entier résultant de la conversion de *expression2*.
>>= (décalage droit au niveau du bit et affectation)	Cet opérateur effectue un décalage vers la droite au niveau du bit et stocke ensuite le contenu dans *expression1*.
>>> (décalage droit non signé au niveau du bit)	Identique à l'opérateur de décalage droit au niveau du bit (>>), sauf 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.
>>>= (décalage droit non signé au niveau du bit et affectation)	Effectue un décalage vers la droite au niveau du bit non signé et stocke ensuite le contenu dans *expression1*.
^ (XOR au niveau du bit)	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.

Opérateur	Description
`^=` (affectation XOR au niveau du bit)	Affecte à ***expression1*** la valeur de ***expression1 ^ expression2***.
`/*` (séparateur de bloc de commentaires)	Démarque une ou plusieurs lignes de commentaires de script.
`,` (virgule)	Evalue ***expression1***, puis ***expression2***, etc.
`add` (concaténation (chaînes))	**Déconseillé depuis Flash Player 5.** Adobe recommande d'utiliser l'opérateur d'addition (+) lorsque vous créez du contenu pour Flash Player 5 ou version ultérieure. **Remarque** : Flash Lite 2.0 remplace également l'opérateur `add` au profit de l'opérateur d'addition (+). Concatène au moins deux chaînes.
`?:` (conditionnel)	Oblige Flash à évaluer *expression1*, et si la valeur de *expression1* est `true`, la valeur de *expression2* est renvoyée ; sinon, la valeur de *expression3* est renvoyée.
`--` (décrément)	Opérateur unaire de pré et post-décrémentation qui soustrait 1 de *expression*.
`/` (division)	Divise *expression1* par *expression2*.
`/=` (affectation de division)	Affecte à ***expression1*** la valeur de ***expression1 / expression2***.
`.` (point)	Permet de naviguer au sein des hiérarchies de clips pour accéder aux clips incorporés (enfants), aux variables ou aux propriétés.
`==` (égalité)	Vérifie si deux expressions sont égales.
`eq` (égalité (chaînes))	**Déconseillé depuis Flash Player 5.** Cet opérateur est déconseillé au profit de l'opérateur `== (equality)`. Renvoie `true` si la représentation de chaîne de *expression1* est égale à celle de *expression2*, sinon renvoie `false`.
`>` (Supérieur à)	Compare deux expressions et détermine si *expression1* est supérieure à *expression2* ; le cas échéant, cet opérateur renvoie `true`.
`gt` (supérieur à (chaînes))	**Déconseillé depuis Flash Player 5.** Cet opérateur est déconseillé au profit de l'opérateur > (supérieur à). Compare la chaîne représentant *expression1* avec la chaîne représentant *expression2* et renvoie `true` si *expression1* est supérieure à *expression2* ; renvoie `false` dans le cas contraire.
`>=` (supérieur ou égal à)	Compare deux expressions et détermine si *expression1* est supérieure ou égale à *expression2* (`true`) ou si *expression1* est inférieure à *expression2* (`false`).
`ge` (supérieur ou égal à (chaînes))	**Déconseillé depuis Flash Player 5.** Cet opérateur est déconseillé au profit de l'opérateur >= (supérieur ou égal à). Renvoie `true` si *expression1* est supérieure ou égale à *expression2*, sinon renvoie `false`.
`++` (incrément)	Opérateur unaire de pré et post-incrémentation qui ajoute 1 à *expression*.
`!=` (inégalité)	Recherche l'inverse de l'opérateur d'égalité (==).
`<>` (inégalité)	**Déconseillé depuis Flash Player 5.** Cet opérateur est déconseillé. Adobe recommande d'utiliser l'opérateur `!= (inégalité)`. Recherche l'inverse de l'opérateur d'égalité (==).
instanceof	Teste si `object` est une occurrence de `classConstructor` ou une sous-classe de `classConstructor`.

Opérateur	Description
`<` (inférieur à)	Compare deux expressions et détermine si *expression1* est inférieure à *expression2* ; le cas échéant, cet opérateur renvoie `true`.
`lt` (inférieur à (chaînes))	**Déconseillé depuis Flash Player 5. Cet opérateur a été déconseillé en faveur de l'opérateur < (inférieur à).** Renvoie true si l'expression1 est inférieure à l'expression2 ; false sinon.
`<=` (inférieur ou égal à)	Compare deux expressions et détermine si *expression1* est inférieure ou égale à *expression2* ; le cas échéant, cet opérateur renvoie `true`.
`le` (inférieur ou égal à (chaînes))	**Déconseillé depuis Flash Player 5.** Cet opérateur est déconseillé dans Flash 5 au profit de l'opérateur <= (inférieur ou égal à). Renvoie `true` si *expression1* est inférieure ou égale à *expression2*, sinon renvoie `false`.
`//` (séparateur de commentaires sur une ligne)	Signale le début d'un commentaire de script.
`&&` (AND logique)	Effectue une opération booléenne sur les valeurs de l'une ou des deux expressions.
`et` (AND logique)	**Déconseillé depuis Flash Player 5.** Adobe recommande d'utiliser l'opérateur logique AND (`&&`). Effectue une opération logique AND (`&&`) dans Flash Player 4.
`!` (NOT logique)	Inverse la valeur booléenne d'une variable ou d'une expression.
`pas` (NOT logique)	**Déconseillé depuis Flash Player 5.** Cet opérateur est déconseillé au profit de l'opérateur `! (NOT logique)`. Effectue une opération NOT logique (!) dans Flash Player 4.
`||` (OR logique)	Evalue *expression1* (l'expression située à gauche de l'opérateur) et renvoie `true` si cette expression renvoie `true`.
`ou` (OR logique)	**Déconseillé**depuis Flash Player 5. Cet opérateur est déconseillé en faveur de l'opérateur `|| (OR logique)`. 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*.
`%=` (affectation modulo)	Affecte à *expression1* la valeur de *expression1* `%` *expression2*.
`*` (multiplication)	Multiplie deux expressions numériques.
`*=` (affectation de multiplication)	Affecte à *expression1* la valeur de *expression1* `*` *expression2*.
new	Crée un objet, initialement anonyme, et appelle la fonction identifiée par le paramètre `constructor`.
`ne` (inégalité (chaînes))	**Déconseillé depuis Flash Player 5.** Cet opérateur est déconseillé au profit de l'opérateur `!= opérateur` (inégalité). Renvoie `true` si *expression1* n'est pas égale à *expression2* et `false` dans le cas contraire.
`{}` (initialiseur d'objet)	Crée un objet et l'initialise avec les paires de propriétés spécifiées *name* et *value*.
`()` (parenthèses)	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.

Opérateur	Description
`===` (égalité stricte)	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.
`!==` (inégalité stricte)	Recherche l'inverse de l'opérateur d'égalité stricte (`===`).
`"` (séparateur de chaîne)	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 en tant que variable, valeur numérique ou autre élément ActionScript.
`-` (soustraction)	Utilisé pour la négation ou la soustraction.
`-=` (affectation de soustraction)	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 en 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.

Remarque : Flash Lite 2.0 prend en charge l'opérateur d'addition (+) pour l'ajout d'expressions numériques et la concaténation de chaînes. Flash Lite 1 prend en charge l'opérateur d'addition (+) pour l'ajout d'expressions numériques (par exemple `var1 = 1 + 2 // renvoie : 3`). Pour Flash Lite 1.x, vous devez utiliser l'opérateur `add` pour concaténer des chaînes.

Disponibilité

Flash Lite 2.0

Opérandes

`expression1` - Un nombre ou une chaîne.

`expression2` - Un nombre ou une 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 Sortie.

```
var name:String = "Cola";
var instrument:String = "Drums";
trace(name + " plays " + instrument); // output: Cola plays Drums
```

Remarque : Flash Lite 1.x ne prend pas en charge l'opérateur d'addition (+) pour la concaténation des chaînes. Pour Flash Lite 1.x, vous devez utiliser l'opérateur `add` pour concaténer des chaînes.

Utilisation 2 : Cette instruction additionne les entiers 2 et 3, puis affiche l'entier obtenu, 5, dans le panneau 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 Sortie :

```
trace(2.5 + 3.25); // output: 5.75
```

Utilisation 3 : Le type de données des variables associées aux champs texte dynamique et de saisie est String. Dans l'exemple suivant, la variable `deposit` est un champ 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 texte deposit, la fonction `trace()` envoie la valeur 4751345,23 vers le panneau 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;
```

Cet 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 Lite 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 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 aussi

`+, 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 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 définit le nom du clip déterminé en concaténant la chaîne « mc » avec la valeur de la variable `i` sur « left_corner ».

```
name["mc" + i] = "left_corner";
```

Disponibilité

Flash Lite 2.0

Opérandes

`myArray : Object` - Nom d'un tableau.

`a0, a1,...aN : Object` - Eléments d'un tableau ; tout type natif ou occurrence d'objet, y compris les tableaux imbriqués.

`i : Number` - Index entier supérieur ou égal à 0.

`myObject : Object` - Nom d'un objet.

`propertyName : String` - 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 (y compris une occurrence de tableau).

Utilisation 3 : Une propriété de l'objet ; soit un type natif, soit une occurrence d'objet (y compris une 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 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 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 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 aussi

`Array`, `Object`, `eval`, `fonction`

=, opérateur d'affectation

`expression1 = expression2`

Affecte la valeur de *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 de *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 Lite 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; // en miles trace
(mercury.diameter); // Renvoie : 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; // en kilomètres trace (mercury.diameter);
// Renvoie : 4878
```

Voir aussi

`==, opérateur d'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 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 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 Lite 2.0

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 le code ActionScript suivant, 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.

L'exemple suivant illustre 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 aussi

`&=, opérateur d'affectation AND au niveau du bit`, `^, opérateur XOR au niveau du bit`, `^=, opérateur d'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 Lite 2.0

Opérandes

`expression1 : Number` - Nombre.

`expression2 : Number` - Nombre.

Valeur renvoyée

`Number` - Valeur de *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 aussi

`&`, opérateur AND au niveau du bit, `^`, opérateur XOR au niveau du bit, `^=`, opérateur d'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 entiers 32 bits et décale tous les bits de *expression1* vers la gauche du nombre d'unités spécifié par l'entier résultant de la conversion de *expression2*. Les positions de bits qui sont vidées suite à cette opération sont remplies par des 0 et les bits déplacés vers la gauche sont supprimés. 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 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 Lite 2.0

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 aussi

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'affectationOpérateur >>> 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 <<= BA = (A << B)
```

Disponibilité

Flash Lite 2.0

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 aussi

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 hexadécimale, ~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 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 est un entier compris entre -2147483648 et 2147483647.

Disponibilité

Flash Lite 2.0

Opérandes

`expression : Number` - Nombre.

Valeur renvoyée

`Number` - Résultat de l'opération au niveau du bit.

Exemple

L'exemple suivant montre l'utilisation de l'opérateur NOT (~) au niveau du bit avec des 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 aussi

`&`, opérateur AND au niveau du bit, `&=`, opérateur d'affectation AND au niveau du bit, `^`, opérateur XOR au niveau du bit, `^=`, opérateur d'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 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 Lite 2.0

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 aussi

`&`, opérateur AND au niveau du bit, `&=`, opérateur d'affectation AND au niveau du bit, `^`, opérateur XOR au niveau du bit, `^=`, opérateur d'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; and x = x | y;
```

Disponibilité
Flash Lite 2.0

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 aussi
`&`, opérateur AND au niveau du bit, `&=`, opérateur d'affectation AND au niveau du bit, `^`, opérateur XOR au niveau du bit, `^=`, opérateur d'affectation XOR au niveau du bit`|`, opérateur 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 de *expression1* vers la droite du nombre d'unités spécifié par l'entier résultant de la conversion de *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) de *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 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 Lite 2.0

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 aussi

`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; and A = (A >> B);
```

Disponibilité

Flash Lite 2.0

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 aussi

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 droit au niveau du bit (>>), sauf 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 Lite 2.0

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 aussi

`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; and A = (A >>> B);
```

Disponibilité

Flash Lite 2.0

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.

Voir aussi

`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 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 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 Lite 2.0

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 aussi

`&`, opérateur AND au niveau du bit, `&=`, opérateur d'affectation AND au niveau du bit, `^=`, opérateur d'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 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 Lite 2.0

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 aussi

`&`, 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 de délimitation de bloc de commentaires

```
/* 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'interprète 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 Lite 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 aussi

`//`, opérateur de commentaires sur une ligne

, opérateur virgule

```
(expression1 , expression2 [, expressionN... ])
```

Evalue *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 Lite 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 de *expression1*, *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 `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 aussi

`(), opérateur parenthèses`

add, opérateur de concaténation de chaînes

string1 `add` *string2*

Déconseillé depuis Flash Player 5. Adobe recommande d'utiliser l'opérateur d'addition (+) lorsque vous créez du contenu pour Flash Player 5 ou version ultérieure.

Remarque : Flash Lite 2.0 remplace également l'opérateur `add` au profit de l'opérateur d'addition (+).

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. Vous devez utiliser l'opérateur d'ajout (+) pour concaténer des chaînes lorsque vous créez du contenu pour Flash Player 4 ou ses versions antérieures.

Disponibilité

Flash Lite 1.0

Opérandes

`string1 : String` - Chaîne.

`string2 : String` - Chaîne.

Valeur renvoyée

`String` - Chaîne concaténée.

Voir aussi

`+, opérateur d'addition`

Opérateur conditionnel ?:

expression1 `?` *expression2* `:` *expression3*

Oblige Flash à évaluer *expression1*, et si la valeur de *expression1* est `true`, la valeur de *expression2* est renvoyée ; sinon, la valeur de *expression3* est renvoyée.

Disponibilité

Flash Lite 1.0

Opérandes

`expression1 : Object` - Expression qui renvoie une valeur booléenne, généralement une expression de comparaison telle que expression `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--
```

Opérateur unaire de pré et post-décrémentation qui soustrait 1 de *expression*. L'*expression* peut être une variable, un élément de tableau ou une propriété d'objet. La forme pré-décrémentale de l'opérateur (*--expression*) soustrait 1 de *expression* et renvoie le résultat. La forme post-décrémentale de l'opérateur (*expression--*) soustrait 1 de *expression* et renvoie la valeur initiale de *expression* (la valeur avant soustraction).

Disponibilité

Flash Lite 1.0

Opérandes

`expression : Number` - Nombre ou variable évaluée sous forme de nombre

Valeur renvoyée

`Number` - Résultat de la valeur décrémentée.

Exemple

La forme pré-décrémentale 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-décrémentale de l'opérateur décrémente `x` pour obtenir 2 (`x - 1 = 2`) et renvoie la valeur d'origine de `x` dans `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 de 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 Lite 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 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 aussi

`%, opérateur modulo`

/=, opérateur d'affectation de division

expression1 /= expression2

Affecte à *expression1* la valeur de *expression1 / expression2*. Par exemple, les deux instructions suivantes sont équivalentes :

```
x /= y; and x = x / y;
```

Disponibilité

Flash Lite 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 aussi

`/`, opérateur de division

. opérateur point

`object.property_or_method`
`instancename.variable`
`instancename.childinstance`
`instancename.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 Lite 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 y est imbriquée.

Valeur renvoyée

`Object` - Méthode, propriété ou clip nommé 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 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 d'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 données différent, ActionScript tente de convertir le type de données de *expression2* pour le faire correspondre à celui de *expression1*.

Disponibilité
Flash Lite 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 aussi

`! opérateur NOT logique`, `!= opérateur d'inégalité`, `Opérateur !== d'inégalité stricte`, `&&, opérateur AND logique`, `||, opérateur OR logique`, `===, opérateur d'égalité stricte`

eq, opérateur d'égalité (chaînes)

expression1 `eq` *expression2*

Déconseillé depuis Flash Player 5. Cet opérateur est déconseillé au profit de l'opérateur `== (equality)`.

Compare l'égalité de deux expressions et renvoie `true` si la chaîne représentant *expression1* est égale à celle de *expression2* ; sinon, l'opération renvoie `false`.

Disponibilité

Flash Lite 1.0

Opérandes

`expression1 : Object` - Nombres, chaînes ou variables.

`expression2 : Object` - Nombres, chaînes ou variables.

Valeur renvoyée

`Boolean` - Résultat de la comparaison.

Voir aussi

`==, opérateur d'égalité`

Opérateur > supérieur à

expression1 `>` *expression2*

Compare deux expressions et détermine si *expression1* est supérieure à *expression2* ; le cas échéant, cet opérateur renvoie `true`. Si *expression1* est inférieure 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 Lite 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 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 est déconseillé au profit de l'opérateur > (supérieur à).

Compare la chaîne représentant *expression1* avec la chaîne représentant *expression2* et renvoie `true` si *expression1* est supérieure à *expression2* ; renvoie `false` dans le cas contraire.

Disponibilité
Flash Lite 1.0

Opérandes
`expression1 : Object` - Nombres, chaînes ou variables.

`expression2 : Object` - Nombres, chaînes ou variables.

Valeur renvoyée
`Boolean` - Résultat booléen de la comparaison.

Voir aussi
`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 Lite 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");
}
```

ge, opérateur supérieur ou égal à (chaînes)

expression1 `ge` *expression2*

Déconseillé depuis Flash Player 5. Cet opérateur est déconseillé au profit de l'opérateur >= (supérieur ou égal à).

Compare la chaîne représentant *expression1* à la chaîne représentant *expression2* et renvoie `true` si *expression1* est supérieure ou égale à *expression2* ; renvoie `false` dans le cas contraire.

Disponibilité

Flash Lite 1.0

Opérandes

`expression1 : Object` - Nombres, chaînes ou variables.

`expression2 : Object` - Nombres, chaînes ou variables.

Valeur renvoyée

`Boolean` - Résultat de la comparaison.

Voir aussi

Opérateur >= supérieur ou égal à

++, opérateur incrément

++expression
expression++

Opérateur unaire de pré et post-incrémentation qui ajoute 1 à *expression*. L'*expression* peut être une variable, un élément de tableau ou une propriété d'objet. La forme pré-incrémentale de l'opérateur (*++expression*) ajoute 1 à *expression* et renvoie le résultat. La forme post-incrémentale de l'opérateur (*expression*++) ajoute 1 à *expression* et renvoie la valeur initiale de *expression* (la valeur avant addition).

La forme pré-incrémentale de l'opérateur incrémente `x` pour obtenir 2 (`x + 1 = 2`) et renvoie le résultat dans `y` :

```
var x:Number = 1;
var y:Number = ++x;
trace("x:"+x); //traces x:2
trace("y:"+y); //traces y:2
```

La forme post-incrémentale de l'opérateur incrémente `x` pour obtenir 2 (`x + 1 = 2`) et renvoie la valeur d'origine de `x` dans `y` :

```
var x:Number = 1;
var y:Number = x++;
trace("x:"+x); //traces x:2
trace("y:"+y); //traces y:1
```

Disponibilité
Flash Lite 1.0

Opérandes
`expression : Number` - Nombre ou variable évaluée sous forme de nombre

Valeur renvoyée
`Number` - Résultat de l'incrémentation.

Exemple
L'exemple suivant utilise ++ comme opérateur de post-incrémentation pour générer l'exécution d'une boucle `while` cinq fois :

```
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 Sortie :

```
0,1,2,3,4,5,6,7,8,9
```

!= opérateur d'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 données différent, ActionScript tente de convertir le type de données de *expression2* pour le faire correspondre à celui de *expression1*.

Disponibilité

Flash Lite 2.0

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 aussi

`! opérateur NOT logique`, `Opérateur !== d'inégalité stricte`, `&&, opérateur AND logique`, `||, opérateur OR logique`, `==, opérateur d'égalité`, `===, opérateur d'égalité stricte`

Opérateur <> d'inégalité

expression1 `<>` *expression2*

Déconseillé depuis Flash Player 5. Cet opérateur est déconseillé. Adobe recommande d'utiliser l'opérateur `!=` (inégalité).

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 Lite 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 aussi
`!= opérateur d'inégalité`

instanceof, opérateur

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 Lite 2.0

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 occurrence ou une sous-classe de `classConstructor`, `instanceof` renvoie `true`. Dans le cas contraire il renvoie `false`. `_global instanceof Object` renvoie également `false`.

Voir aussi

typeof, opérateur

Opérateur < inférieur à

expression1 < *expression2*

Compare deux expressions et détermine si *expression1* est inférieure à *expression2* ; le cas échéant, 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 Lite 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 illustrent les résultats `true` et `false` des comparaisons numériques et de 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
```

lt, opérateur 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 inférieure à *expression2*, sinon renvoie `false`.

Disponibilité

Flash Lite 1.0

Opérandes

`expression1 : Object` - Nombres, chaînes ou variables.

`expression2 : Object` - Nombres, chaînes ou variables.

Valeur renvoyée

`Boolean` - Résultat de la comparaison.

Voir aussi

Opérateur < inférieur à

Opérateur <= inférieur ou égal à

expression1 <= *expression2*

Compare deux expressions et détermine si *expression1* est inférieure ou égale à *expression2* ; le cas échéant, 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 Lite 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 illustrent les résultats `true` et `false` des comparaisons numériques et de 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
```

le, opérateur inférieur ou égal à (chaînes)

expression1 le *expression2*

Déconseillé depuis Flash Player 5. Cet opérateur est déconseillé dans Flash 5 au profit de l'opérateur <= (inférieur ou égal à).

Compare *expression1* à *expression2* et renvoie `true` si *expression1* est inférieure ou égal à *expression2*, sinon renvoie `false`.

Disponibilité

Flash Lite 1.0

Opérandes

`expression1 : Object` - Nombres, chaînes ou variables.

`expression2 : Object` - Nombres, chaînes ou variables.

Valeur renvoyée

`Boolean` - Résultat de la comparaison.

Voir aussi

`Opérateur <= inférieur ou égal à`

//, opérateur 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'interprète d'ActionScript.

Disponibilité

Flash Lite 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 aussi

`/*, opérateur de délimitation de bloc de commentaires`

&&, opérateur AND logique

`expression1` `&&` *`expression2`*

Effectue une opération booléenne sur les valeurs de l'une ou des deux expressions. Evalue *expression1* (l'expression située à gauche de l'opérateur) et renvoie `false` si cette expression renvoie `false`. Si *expression1* renvoie `true`, *expression2* (l'expression située à droite de l'opérateur) est évaluée. Si *expression2* renvoie `true`, le résultat final est `true` ; sinon renvoie `false`. L'expression `true&&true` renvoie la valeur true. L'expression `true&&false` renvoie la valeur false. L'expression `false&&false` renvoie la valeur false et enfin l'expression `false&&true` renvoie l'expression false.

Disponibilité

Flash Lite 1.0

Opérandes

`expression1 : Number` - Valeur booléenne ou expression qui se convertit en valeur booléenne.

`expression2 : Number` - Valeur booléenne ou 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 un tour ou marque des points durant la partie. Le script affiche le texte « You Win the Game ! » dans le panneau 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 aussi

`! opérateur NOT logique`, `!= opérateur d'inégalité`, `Opérateur !== d'inégalité stricte`, `||, opérateur OR logique`, `==, opérateur d'égalité`, `===, opérateur d'égalité stricte`

AND, opérateur and logique

condition1 and *condition2*

Déconseillé depuis Flash Player 5. Adobe recommande d'utiliser l'opérateur logique AND (`&&`).

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 Lite 1.0

Opérandes

`condition1 : Boolean` - Condition ou expression qui renvoie `true` ou `false`.

`condition2 : Boolean` - Condition ou expression qui renvoie `true` ou `false`.

Valeur renvoyée

`Boolean` - Résultat booléen de l'opération logique.

Voir aussi

`&&, 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`. Par conséquent, `!true` renvoie `false` et `!false` renvoie `true`.

Disponibilité
Flash Lite 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 cette dernière est `true`, l'instruction trace() envoie une chaîne au panneau 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 aussi
`!= opérateur d'inégalité`, `Opérateur !== d'inégalité stricte`, `&&, opérateur AND logique`, `||, opérateur OR logique`, `==, opérateur d'égalité`, `===, opérateur d'égalité stricte`

NOT, opérateur Sauf logique

`not` *`expression`*

Déconseillé depuis Flash Player 5. Cet opérateur est déconseillé en faveur de l'opérateur `!` (NOT logique).

Effectue une opération NOT logique `(!)` dans Flash Player 4.

Disponibilité
Flash Lite 1.0

Opérandes
`expression : Object` - Variable ou autre expression qui se convertit en valeur booléenne.

Valeur renvoyée
`Boolean` - Résultat de l'opération logique.

Voir aussi
`! opérateur NOT logique`

||, opérateur OR logique

`expression1` `||` *`expression2`*

Evalue *expression1* (l'expression située à gauche de l'opérateur) et renvoie `true` si cette expression renvoie `true`. 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 renvoie `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` 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 Lite 1.0

Opérandes

`expression1 : Number` - Valeur booléenne ou expression qui se convertit en valeur booléenne.

`expression2 : Number` - Valeur booléenne ou 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 la valeur `false`, le bloc `if` est exécuté car une condition renvoie la valeur `true`.

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");
}
/* The following is sent to the Output panel: /* The following is sent to the log file: fx1
called IF statement entered */
```

Voir aussi

`! opérateur NOT logique`, `!= opérateur d'inégalité`, `Opérateur !== d'inégalité stricte`, `&&, opérateur AND logique`, `==, opérateur d'égalité`, `===, opérateur d'égalité stricte`

or, opérateur OR logique

condition1 `or` *condition2*

Déconseillédepuis Flash Player 5. Cet opérateur est déconseillé en faveur de l'opérateur `|| (OR logique)`.

Evalue *condition1* et *condition2*, si l'une des deux expressions est `true`, l'expression entière est `true`.

Disponibilité

Flash Lite 1.0

Opérandes

`condition1 : Boolean` - Une expression qui renvoie `true` ou `false`.

`condition2 : Boolean` - Une expression qui renvoie `true` ou `false`.

Valeur renvoyée

`Boolean` - Résultat de l'opération logique.

Voir aussi

`||, 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. L'*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 Lite 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

`Nombre` - 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 aussi

`/, opérateur de 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; and x = x % y;
```

Disponibilité

Flash Lite 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

`Nombre` - 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 aussi

`%, 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 Lite 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 Lite 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` (n'est pas un nombre).

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 aussi

opérateur * (multiplication)

new, opérateur

`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 Lite 2.0

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 aussi

`Opérateur d'accès au tableau [], {}, opérateur initialiseur d'objet`

ne, opérateur différent de (chaînes)

`expression1 ne expression2`

Déconseillé depuis Flash Player 5. Cet opérateur est déconseillé au profit de l'opérateur `!= opérateur` (inégalité).

Compare *expression1* à *expression2* et renvoie `true` si *expression1* n'est pas égale à *expression2* ; sinon renvoie `false`.

Disponibilité

Flash Lite 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` ; sinon, renvoie `false`.

Voir aussi

`!= opérateur d'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 Lite 2.0

Opérandes

`object : Object` - Objet à créer. *name1,2,...N* Nom 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 aussi

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 : Evalue 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 Lite 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 des parenthèses pour contrôler l'ordre d'exécution des expressions (la valeur de chaque expression apparaît dans le panneau Sortie) :

```
trace((2 + 3)*(4 + 5)); // displays 45
trace((2 + 3) * (4 + 5)); // writes 45
trace(2 + (3 * (4 + 5))); // displays 29
trace(2 + (3 * (4 + 5))); // writes 29
trace(2+(3*4)+5); // displays 19
trace(2 + (3 * 4) + 5); // writes19
```

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 aussi

`with, instruction`

===, opérateur d'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 Lite 2.0

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 lesquelles 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 aussi

`! opérateur NOT logique`, `!= opérateur d'inégalité`, `Opérateur !== d'inégalité stricte`, `&&, opérateur AND logique`, `||, opérateur OR logique`, `==, opérateur d'égalité`

Opérateur !== d'inégalité stricte

expression1 `!==` *expression2*

Recherche l'inverse 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 égale à *expression2*, et que leurs types de données sont égaux, 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.

Disponibilité
Flash Lite 2.0

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 aussi
`! opérateur NOT logique`, `!= opérateur d'inégalité`, `&&, opérateur AND logique`, `||, opérateur OR logique`, `==, opérateur d'égalité`, `===, opérateur d'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 en tant que variable, valeur numérique ou autre élément ActionScript.

Disponibilité
Flash Lite 1.0

Opérandes
`text : String` - Séquence de zéros ou de plusieurs caractères.

Exemple

L'exemple suivant utilise les guillemets pour indiquer que la valeur de la variable *yourGuess* correspond à la chaîne littérale `"Prince Edward Island"` et non au nom d'une variable. La valeur de `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 in the Output panel
// writes Prince Edward Island to the log file
```

Voir aussi

`String`, `String, fonction`

-, opérateur de 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.

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 Lite 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.

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 d'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` (n'est pas un nombre) est renvoyé.

Disponibilité

Flash Lite 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

`Nombre` - 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 aussi

`-, opérateur de soustraction`

Opérateur : de type

```
[ 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 ou une affectation de variable, cet opérateur spécifie le type de variable. Lorsqu'il fait partie d'une déclaration ou d'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 point (`.`). 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 Lite 2.0

Opérandes
`variableName : Object` - Identificateur d'une variable.

`type` : Type de données natif, nom de classe que vous avez défini ou nom d'interface.

`functionName` : Identificateur d'une fonction.

`parameter` : Identificateur d'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 `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 aussi
`set variable, instruction`, `Array, fonction`

typeof, opérateur

`typeof(expression)`

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`.

Disponibilité
Flash Lite 2.0

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	`chaîne`
Movie clip	`movieclip`
Bouton	`object`
Text field	`object`
Number	`nombre`
Boolean	`boolean`
Object	`object`
Function	`fonction`

Voir aussi

`instanceof, opérateur`

void, opérateur

`void` *`expression`*

L'opérateur `void` évalue une expression, puis supprime sa valeur en renvoyant `undefined`. L'opérateur `void` est souvent utilisé dans les comparaisons incluant l'opérateur `==` pour tester les valeurs non définies.

Disponibilité

Flash Lite 2.0

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 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 précis au sein d'une instruction `switch`.
case	Définit une condition pour l'instruction `switch`.
classe	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.
par défaut	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 ; renvoie `false` dans le cas contraire.
do..while	Semblable à une boucle `while`, à la différence que les instructions sont exécutées une fois avant l'évaluation initiale de la condition.
dynamique	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	Evalue 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	Evalue l'expression ***init*** (initialiser) une fois, puis amorce une séquence de bouclage.
Instruction 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.
fonction	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	Evalue 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.

Instruction	Description
set variable	Associe une valeur à une variable.
statique	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 super-classe 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	Evalue 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)`***.

break, instruction

`break`

Apparaît au sein d'une boucle (`for`, `for..in`, `do..while` ou `while`) ou dans un bloc d'instructions associées à un cas précis au sein d'une instruction `switch`. Lorsqu'elle est utilisée dans une boucle, l'instruction `break` oblige Flash à ignorer le reste du corps de la boucle, arrête l'action de la boucle et exécute l'instruction qui suit l'instruction de bouclage. Dans le cadre d'une instruction `switch`, l'instruction `break` oblige 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 Lite 1.0

Exemple

L'exemple suivant utilise l'instruction `break` pour sortir d'une boucle infinie :

```
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 aussi

_forceframerate, propriété

case, instruction

`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 l'instruction break dans le paramètre *statement(s)*, l'exécution continue avec l'instruction `case` suivante au lieu de sortir de l'instruction `switch`.

Disponibilité

Flash Lite 1.0

Paramètres

expression : `String` - Toute expression.

Exemple

L'exemple suivant définit les conditions de l'instruction `switchthisMonth`. 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 aussi

break, instruction

class, instruction

[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 exécuter 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 appliquer 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 imbriquer 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 lors de l'exécution et y accéder, faites précéder l'instruction class par le mot-clé `dynamic`. Pour déclarer qu'une classe implémente une interface, employez le mot-clé `implements`. Pour créer des sous-classes d'une classe, employez le mot-clé `extends`. (Une classe ne peut étendre qu'une seule autre classe, mais peut implémenter plusieurs interfaces.) Vous pouvez employer `implements` et `extends` dans la même instruction. Les exemples suivants présentent des emplois typiques 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 Lite 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 aussi

`dynamic, instruction`

continue, instruction

```
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 Lite 1.0

Exemple

Dans la boucle `while` suivante, l'instruction `continue` oblige l'interprète de Flash à ignorer le reste du corps de la boucle et à atteindre le haut 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` oblige l'interprète de Flash à ignorer le reste du corps de la boucle et à atteindre le bas 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` oblige l'interprète de 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, `continue` oblige l'interprète de Flash à ignorer le reste du corps de la boucle et à passer de nouveau au début de la boucle, où la prochaine valeur de l'énumération est traitée :

```
for (i in _root) {
    if (i == "$version") {
    continue;
    }
    trace(i);
}
```

Voir aussi

`do..while, instruction`

default, instruction

`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 `default`. L'instruction case `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 Lite 2.0

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 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 aussi

`switch, instruction`

delete, instruction

`delete` *reference*

Détruit la référence d'objet spécifiée par le paramètre *reference* et renvoie `true` si la référence est supprimée correctement ; renvoie `false` dans le cas contraire. Cet opérateur permet de libérer la mémoire mobilisée par les scripts. Vous pouvez exploiter 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 employé 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 Lite 2.0

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 aussi

`set variable, instruction`

do..while, instruction

`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 `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 Lite 1.0

Paramètres

condition : `Boolean` - La 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 aussi

`break, instruction`

dynamic, instruction

```
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.

Disponibilité
Flash Lite 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 aussi
`class, instruction`

else, instruction

```
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 s'exécute.

Disponibilité
Flash Lite 1.0

Paramètres
condition : `Boolean` - Une expression qui renvoie `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 aussi
ifFrameLoaded, fonction

else if, instruction

```
if (condition){ statement(s); }
    else if (condition){ statement(s);}
```

Evalue 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'interprète de Flash exécute les instructions qui suivent la condition entre accolades ({}). Si la condition `else if` renvoie `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 Lite 1.0

Paramètres
condition : `Boolean` - Une expression qui renvoie `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 aussi
ifFrameLoaded, fonction

extends, instruction

Utilisation 1 :

```
class className extends otherClassName {}
```

Utilisation 2 :

```
interface interfaceName extends otherInterfaceName {}
```

Remarque : Pour utiliser ce mot-clé, vous devez spécifier ActionScript 2.0 et Flash Player 6 ou une version plus récente dans l'onglet Flash de la boîte de dialogue Paramètres de publication de votre fichier FLA. 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.

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`. Toute interface qui étend une autre interface reprend toutes les déclarations de méthode de l'interface d'origine.

Disponibilité

Flash Lite 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 crée une occurrence d'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 aussi

class, instruction

for, instruction

```
    for(init; condition; next) {
statement(s);
}
```

Evalue 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, `statement` est exécutée 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 s'exécute.

Disponibilité

Flash Lite 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 aussi

++, 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'instruction `for..in`.

Certaines propriétés ne peuvent pas être énumérées par l'instruction `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..` effectue une itération sur les propriétés des objets dans le chaînage de prototype de l'objet itéré. Les propriétés de l'objet sont énumérées en premier, puis les propriétés de son prototype immédiat, puis les propriétés du prototype du prototype, etc. L'instruction `for..in` n'énumère pas deux fois le même nom de propriété. 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 s'exécute.

Si vous écrivez une boucle `for..in` dans un fichier de classe (un fichier externe AS), les membres de l'occurrence ne sont 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 Lite 2.0

Paramètres
variableIterant : `String` - Nom d'une variable devant servir d'itération, référençant chaque propriété d'un objet ou d'un élément dans un tableau.

Exemple
L'exemple suivant utilise une boucle `for..in` pour effectuer une itération 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` pour effectuer une itération 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 effectuer 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 plusieurs enfants, `_RedRadioButton_`, `_GreenRadioButton_`, et `_BlueRadioButton_`.

```
for (var name in RadioButtonGroup) { RadioButtonGroup[name].gotoAndStop(2); }
```

function, instruction

Utilisation 1 : (Déclare une fonction nommée.)

```
function functionname([parameter0, parameter1,...parameterN]){statement(s)}
```

Utilisation 2 : (Déclare une fonction anonyme et renvoie une référence à cette dernière.)

```
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 `fonction` 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 Lite 2.0

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");
}
```

get, instruction

```
function get property () { // your statements here }
```

Remarque : Pour utiliser ce mot-clé, vous devez spécifier ActionScript 2.0 et Flash Player 6 ou une version plus récente dans l'onglet Flash de la boîte de dialogue Paramètres de publication de votre fichier FLA. 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.

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.

Disponibilité
Flash Lite 2.0

Paramètres
property : `String` - Mot que vous utilisez pour faire référence à la propriété qui est lue par `get` ; cette valeur doit être identique à celle 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 aussi

addProperty (méthode Object.addProperty)

if, instruction

`if(condition) { statement(s); }`

Evalue 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 s'exécute.

Disponibilité

Flash Lite 1.0

Paramètres

condition : `Boolean` - Une expression qui renvoie `true` ou `false`.

Exemple

Dans l'exemple suivant, la condition entre parenthèses évalue la variable `name` pour vérifier que sa valeur littérale est bien `"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 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 aussi

`else, instruction`

implements, instruction

`className` `implements` *`interface01`* `[, interface02 , ...]`

Remarque : Pour utiliser ce mot-clé, vous devez spécifier ActionScript 2.0 et Flash Player 6 ou une version plus récente dans l'onglet Flash de la boîte de dialogue Paramètres de publication de votre fichier FLA. 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.

Spécifie qu'une classe doit définir toutes les méthodes déclarées dans l'interface (ou les interfaces) en cours d'implémentation.

Disponibilité

Flash Lite 2.0

Exemple

Reportez-vous à la section `interface`.

Voir aussi

`class, instruction`

import, instruction

`import` *`className`*
`import` *`packageName`*`.*`

Remarque : Pour utiliser ce mot-clé, vous devez spécifier ActionScript 2.0 et Flash Player 6 ou une version plus récente dans l'onglet Flash de la boîte de dialogue Paramètres de publication de votre fichier FLA. Cette instruction est prise en charge dans le panneau Actions, ainsi que dans les fichiers de classe externes.

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 Lite 2.0

Paramètres
className : `String` - Nom qualifié d'une classe définie dans un fichier de classe externe.

interface, instruction

```
interface InterfaceName [extends InterfaceName ] {}
```

Remarque : Pour utiliser ce mot-clé, vous devez spécifier ActionScript 2.0 et Flash Player 6 ou une version plus récente dans l'onglet Flash de la boîte de dialogue Paramètres de publication de votre fichier FLA. 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.

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.

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 aussi

`class, instruction`

intrinsic, instruction

```
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 Lite 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 aussi

`class, instruction`

private, instruction

```
    class className{
    private var name;
    private function name() {
    // your statements here
    }
}
```

Remarque : Pour utiliser ce mot-clé, vous devez spécifier ActionScript 2.0 et Flash Player 6 ou une version plus récente dans l'onglet Flash de la boîte de dialogue Paramètres de publication de votre fichier FLA. 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.

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 réservé aux définitions de classe et ne permet pas de créer des définitions d'interface.

Disponibilité

Flash Lite 2.0

Paramètres

name : `String` - Nom de la variable ou de la fonction à spécifier en tant que privée.

Exemple

L'exemple suivant démontre comment masquer certaines propriétés au sein d'une classe à l'aide du mot-clé `private`. Créez un fichier AS intitulé Login.as.

```
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 le même répertoire que Login.as, créez un document FLA ou AS. Entrez le code ActionScript suivant dans l'image 1 du scénario.

```
import Login;
var gus:Login = new Login("Gus", "Smith");
trace(gus.username); // output: Gus
trace(gus.password); // output: undefined
trace(gus.loginPassword); // error
```

Dans la mesure où `loginPassword` est une variable privée, vous ne pouvez pas y accéder en dehors du fichier de classe Login.as. Les tentatives d'accès à la variable privée génèrent un message d'erreur.

Voir aussi

`public, instruction`

public, instruction

```
class className{
    public var name;
    public function name() {
    // your statements here } }
```

Remarque : Pour utiliser ce mot-clé, vous devez spécifier ActionScript 2.0 et Flash Player 6 ou une version plus récente dans l'onglet Flash de la boîte de dialogue Paramètres de publication de votre fichier FLA. 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.

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 Lite 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 aussi

`private, instruction`

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 Lite 2.0

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 aussi
`Array, fonction`

set, instruction

```
function set property(varName) {
    // your statements here
}
```

Remarque : Pour utiliser ce mot-clé, vous devez spécifier ActionScript 2.0 et Flash Player 6 ou une version plus récente dans l'onglet Flash de la boîte de dialogue Paramètres de publication de votre fichier FLA. 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.

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.

Disponibilité
Flash Lite 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 cet exemple, la fonction `get` s'exécute lorsque la valeur est représentée. La fonction `set` se déclenche uniquement lorsque vous lui transmettez une valeur, comme indiqué sur la ligne :

```
gus.username = "Rupert";
```

Voir aussi

`getProperty`, `fonction`

set variable, instruction

`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 Lite 2.0

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);
```

static, instruction

```
class className{
    static var name;
    static function name() {
    // your statements here } }
```

Remarque : Pour utiliser ce mot-clé, vous devez spécifier ActionScript 2.0 et Flash Player 6 ou une version plus récente dans l'onglet Flash de la boîte de dialogue Paramètres de publication de votre fichier FLA. 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.

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 d'occurrence de la classe en utilisant la syntaxe `someClassName.name`. Si vous créez une occurrence de la classe, vous pouvez également accéder à un membre statique en utilisant l'occurrence, mais uniquement par le biais d'une fonction non statique qui accède au membre statique.

Ce mot-clé est réservé aux définitions de classe et ne permet pas de créer des définitions d'interface.

Disponibilité
Flash Lite 2.0

Paramètres
name : `String` - Nom de la variable ou de la fonction à spécifier en tant que publique.

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 aussi
`private, instruction`

super, instruction

```
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 Lite 2.0

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).

switch, instruction

```
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 Lite 1.0

Paramètres
expression - Toute expression.

Exemple
Dans l'exemple suivant, si le paramètre `String.fromCharCode(Key.getAscii())` renvoie `A`, l'instruction `trace()` suivant `case "A"` s'exécute ; si le paramètre renvoie `a`, l'instruction `trace()` suivant `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 aussi

`===, opérateur d'égalité stricte`

throw, instruction

`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 Sortie.

De manière générale, le système renvoie des occurrences de la classe Error ou de ses sous-classes (consultez la section Exemple).

Disponibilité

Flash Lite 2.0

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 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 aussi

Error

try..catch..dernière instruction

```
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 avec 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'interprète d'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 Lite 2.0

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`. Etant 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. Etant 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 suivant 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 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 aussi

Error

var, instruction

`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, elles 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 Lite 2.0

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 mot-clé `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
```

while, instruction

```
while(condition) { statement(s); }
```

Evalue 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* 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 s'exécute.

Disponibilité

Flash Lite 1.0

Paramètres

condition : `Boolean` - Une expression qui renvoie `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 représentée. 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 Sortie.

```
0
3
6
9
12
15
18
```

Voir aussi

`continue, instruction`

with, instruction

```
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 si deux des propriétés spécifiées sont `length` et `concat`, ces propriétés 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'occurrence `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* dans 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 Lite 2.0

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 `r` ne 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 Sortie.

```
area = 28.2743338823081
x = -3
y = 3
```

Commandes fscommand2

Les commandes suivantes sont disponibles pour la fonction `fscommand2()`. Pour obtenir une description de la fonction `fscommand2()`, consultez l'entrée « Fonction fscommand2 » dans la section « Fonctions globales ».

Commandes fscommand2

Commande	Description
ExtendBacklightDuration	Cette fonction prolonge le rétroéclairage pendant la période spécifiée.
FullScreen	Définit la taille de la zone d'affichage à utiliser pour le rendu.
GetBatteryLevel	Renvoie le niveau actuel de la batterie.
GetDevice	Définit un paramètre qui identifie le périphérique servant à exécuter Flash Lite.
GetDeviceID	Définit un paramètre qui représente l'identificateur unique du périphérique (par exemple, le numéro de série).
GetFreePlayerMemory	Renvoie la quantité de mémoire heap, en kilo-octets, disponible actuellement pour Flash Lite.
GetMaxBatteryLevel	Renvoie le niveau maximum de la batterie du périphérique.
GetMaxSignalLevel	Renvoie la force maximale du signal sous forme de valeur numérique.
GetMaxVolumeLevel	Renvoie le niveau de volume maximum du périphérique sous forme de valeur numérique.
GetNetworkConnectionName	Renvoie le nom de la connexion réseau active ou par défaut.
GetNetworkConnectStatus	Renvoie une valeur qui indique l'état de la connexion réseau actuelle.
GetNetworkGeneration	Renvoie la génération du réseau mobile sans fil actuel (par exemple 2G ou deuxième génération de communications mobiles sans fil).
GetNetworkName	Définit un paramètre reprenant le nom du réseau actif.
GetNetworkRequestStatus	Renvoie une valeur indiquant l'état de la requête HTTP la plus récente.
GetNetworkStatus	Renvoie une valeur indiquant l'état réseau du téléphone (indique si un réseau est enregistré et si le téléphone est en mode mobile).
GetPlatform	Définit un paramètre qui identifie la plate-forme actuelle, ce qui décrit de façon générale la classe du périphérique.
GetPowerSource	Renvoie une valeur qui indique si l'alimentation vient de la batterie ou d'une source externe.
GetSignalLevel	Renvoie la force du signal actuel sous forme de valeur numérique.
GetSoftKeyLocation	Renvoie une valeur qui indique l'emplacement des touches programmables sur le périphérique.
GetTotalPlayerMemory	Renvoie la quantité totale de la mémoire heap, en kilo-octets, affectée à Flash Lite.
GetVolumeLevel	Renvoie le niveau de volume actuel du périphérique sous forme de valeur numérique.
Quit	Entraîne l'arrêt du lecteur Flash Lite et ferme ce programme.
ResetSoftKeys	Rétablit les valeurs d'origine des touches programmables.
SetFocusRectColor	Définit la couleur du rectangle de focus sur une autre couleur.
SetInputTextType	Spécifie le mode d'ouverture du champ texte de saisie.

Commande	Description
SetSoftKeys	Reconfigure les touches programmables d'un périphérique mobile.
StartVibrate	Active la fonctionnalité de vibration du téléphone.
StopVibrate	Arrête la vibration actuelle, si nécessaire.

Commande ExtendBacklightDuration fscommand2

`ExtendBacklightDuration`

Cette fonction prolonge le rétroéclairage pendant la période spécifiée.

Si la durée est supérieure à zéro, cette commande spécifie le nombre de secondes (avec un maximum de 60) pendant lesquelles le rétroéclairage doit rester allumé. Si cette période s'écoule sans appel supplémentaire à cette commande, le comportement de rétroéclairage applique de nouveau la durée par défaut. Si la durée est de zéro, le comportement de rétroéclairage applique immédiatement le comportement par défaut.

Remarque : Cette fonctionnalité dépend du système. Par exemple, certains systèmes limitent la durée totale de prolongement du rétroéclairage.

Remarque : Cette commande n'est pas prise en charge pour les périphériques BREW.

Commande	Paramètres	Valeur renvoyée
ExtendBacklightDuration	`duration` La durée du rétroéclairage, en secondes. Valeur maximum de 60 secondes.	-1 : pas de prise en charge.. 0: Une erreur s'est produite et l'opération n'a pas pu se terminer. 1: Succès.

Disponibilité
Flash Lite 2.0

Exemple
L'exemple suivant prolonge le rétroéclairage de 45 secondes :

```
status = FSCommand2("ExtendBacklightDuration", 45)
```

Commande fscommand2 FullScreen

`FullScreen`

Définit la taille de la zone d'affichage à utiliser pour le rendu.

La taille peut correspondre à une variable définie ou une valeur constante de chaîne incluant l'une des valeurs suivantes : `true`(plein écran) ou `false` (n'occupe pas toute la surface de l'écran). Toute autre valeur est traitée comme la valeur `false`.

Remarque : Cette commande est prise en charge uniquement lorsque Flash Lite est en cours d'exécution en mode autonome. Elle n'est pas prise en charge lorsque le lecteur s'exécute dans le contexte d'une autre application (par exemple, en tant que module externe dans un navigateur).

Commande	Paramètres	Valeur renvoyée
FullScreen	size	-1 : pas de prise en charge. 0 : prise en charge.

Disponibilité
Flash Lite 1.1

Exemple
L'exemple suivant définit la taille de la zone d'affichage en mode plein écran :

```
status = fscommand2("FullScreen", true);
```

Commande fscommand2 GetBatteryLevel

`GetBatteryLevel`

Renvoie le niveau actuel de la batterie. Il s'agit d'une valeur numérique comprise entre 0 et la valeur maximale renvoyée par la variable `GetMaxBatteryLevel`.

Remarque : Cette commande n'est pas prise en charge pour les périphériques BREW.

Commande	Paramètres	Valeur renvoyée
GetBatteryLevel	Aucun.	-1 : pas de prise en charge. Autres valeurs numériques : niveau actuel de la batterie.

Disponibilité
Flash Lite 1.1

Exemple
L'exemple suivant définit la variable `battLevel` sur le niveau actuel de la batterie :

```
battLevel = fscommand2("GetBatteryLevel");
```

Commande fscommand2 GetDevice

`GetDevice`

Définit un paramètre qui identifie le périphérique servant à exécuter Flash Lite. Cet identificateur correspond généralement au nom de modèle.

Commande	Paramètres	Valeur renvoyée
GetDevice	*device* Chaîne devant recevoir l'identificateur du périphérique. Il peut s'agir du nom d'une variable ou d'une chaîne qui contient le nom d'une variable.	-1 : pas de prise en charge. 0 : prise en charge..

Disponibilité
Flash Lite 1.1

Exemple
L'exemple suivant affecte l'identificateur de périphérique à la variable `device` :

```
status = fscommand2("GetDevice", "device");
```

Vous trouverez ci-dessous des exemples de résultats et les périphériques qu'ils désignent :

`D506i`Téléphone Mitsubishi 506i. `DFOMA1`Téléphone Mitsubishi FOMA1. `F506i`Téléphone Fujitsu 506i. `FFOMA1`Téléphone Fujitsu FOMA1. `N506i`Téléphone NEC 506i. `NFOMA1`Téléphone NEC FOMA1. `Nokia3650` Téléphone Nokia 3650. `p506i` Téléphone Panasonic 506i. `PFOMA1`Téléphone Panasonic FOMA1. `SH506i`Téléphone Sharp 506i. `SHFOMA1`Téléphone Sharp FOMA1. `SO506i`Téléphone Sony 506i.

Commande fscommand2 GetDeviceID

`GetDeviceID`

Définit un paramètre qui représente l'identificateur unique du périphérique (par exemple, le numéro de série).

Commande	Paramètres	Valeur renvoyée
GetDeviceID	`id`Une chaîne devant recevoir l'identificateur unique du périphérique. Il peut s'agir du nom d'une variable ou d'une chaîne qui contient le nom d'une variable.	-1 : pas de prise en charge. 0 : prise en charge.

Disponibilité
Flash Lite 1.1

Exemple
L'exemple suivant affecte l'identificateur unique à la variable `deviceID` :

```
status = fscommand2("GetDeviceID", "deviceID");
```

GetFreePlayerMemory Commande fscommand2

`GetFreePlayerMemory`

Renvoie la quantité de mémoire heap, en kilo-octets, disponible actuellement pour Flash Lite.

Commande	Paramètres	Valeur renvoyée
GetFreePlayerMemory	Aucune	-1 : pas de prise en charge. 0 ou valeur positive : kilo-octets disponibles dans la mémoire heap.

Disponibilité
Flash Lite 1.1

Exemple
L'exemple suivant définit l'`état` en fonction de la quantité de mémoire disponible :

```
status = fscommand2("GetFreePlayerMemory");
```

Commande fscommand2 GetMaxBatteryLevel

`GetMaxBatteryLevel`

Renvoie le niveau maximum de la batterie du périphérique. Il s'agit d'une valeur numérique supérieure à 0.

Remarque : Cette commande n'est pas prise en charge pour les périphériques BREW.

Commande	Paramètres	Valeur renvoyée
GetMaxBatteryLevel	Aucune	-1 : pas de prise en charge. Autres valeurs : niveau maximum de la batterie.

Disponibilité
Flash Lite 1.1

Exemple
L'exemple suivant définit la variable `maxBatt` sur le niveau maximal de la batterie :

```
maxBatt = fscommand2("GetMaxBatteryLevel");
```

Commande fscommand2 GetMaxSignalLevel

`GetMaxSignalLevel`

Renvoie la force maximale du signal sous forme de valeur numérique.

Remarque : Cette commande n'est pas prise en charge pour les périphériques BREW.

Commande	Paramètres	Valeur renvoyée
GetMaxSignalLevel	Aucune	-1 : pas de prise en charge. Autres valeurs numériques : niveau maximum du signal.

Disponibilité
Flash Lite 1.1

Exemple
L'exemple suivant affecte l'intensité maximale du signal à la variable `sigStrengthMax` :

```
sigStrengthMax = fscommand2("GetMaxSignalLevel");
```

Commande fscommand2 GetMaxVolumeLevel

`GetMaxVolumeLevel`

Renvoie le niveau de volume maximum du périphérique sous forme de valeur numérique.

Commande	Paramètres	Valeur renvoyée
GetMaxVolumeLevel	Aucune	-1 : pas de prise en charge. Autres valeurs : niveau maximum du volume.

Disponibilité
Flash Lite 1.1

Exemple
L'exemple suivant définit la variable `maxvolume` sur le niveau de volume maximal du périphérique :

```
maxvolume = fscommand2("GetMaxVolumeLevel");
trace (maxvolume); // output: 80
```

Commande fscommand2 GetNetworkConnectionName

`GetNetworkConnectionName`

Renvoie le nom de la connexion réseau active ou par défaut. Pour les périphériques mobiles, cette connexion est également appelée point d'accès.

Remarque : Cette commande n'est pas prise en charge pour les périphériques BREW.

Commande	Paramètres	Valeur renvoyée
GetNetworkConnectionName	Aucune	`-1` : pas de prise en charge. `0` : succès. Renvoie le nom de la connexion réseau active. `1` : succès. Renvoie le nom de la connexion réseau par défaut. `2`: impossible d'extraire le nom de la connexion.

Disponibilité
Flash Lite 2.0

Exemple
L'exemple suivant renvoie le nom de la connexion réseau active ou par défaut dans l'argument `myConnectionName` :

```
status = FSCommand2("GetNetworkConnectionName", "myConnectionName");
```

Commande fscommand2 GetNetworkConnectStatus

`GetNetworkConnectStatus`

Renvoie une valeur qui indique l'état de la connexion réseau actuelle.

Remarque : Cette commande n'est pas prise en charge pour les périphériques BREW.

Commande	Paramètres	Valeur renvoyée
GetNetworkConnectStatus	Aucune	`-1` : pas de prise en charge. `0`: une connexion réseau est actuellement active. `1` : le périphérique tente de se connecter au réseau. `2` : aucune connexion réseau n'est actuellement active. `3` : la connexion réseau a été interrompue. `4` : la connexion réseau ne peut pas être déterminée.

Disponibilité
Flash Lite 1.1

Exemple
L'exemple suivant affecte l'état de la connexion réseau à la variable `connectstatus`, puis utilise une instruction `switch` pour mettre à jour un champ texte avec l'état de la connexion :

```
connectstatus = FSCommand2("GetNetworkConnectStatus");
switch (connectstatus) {
    case -1 :
    _root.myText += "connectstatus not supported" + "\n";
    break;
    case 0 :
    _root.myText += "connectstatus shows active connection" + "\n";
    break;
    case 1 :
    _root.myText += "connectstatus shows attempting connection" + "\n";
    break;
    case 2 :
    _root.myText += "connectstatus shows no connection" + "\n";
    break;
    case 3 :
    _root.myText += "connectstatus shows suspended connection" + "\n";
    break;
    case 4 :
    _root.myText += "connectstatus shows indeterminable state" + "\n";
    break;
}
```

Commande fscommand2 GetNetworkGeneration

`GetNetworkGeneration`

Renvoie la génération du réseau mobile sans fil actuel, par exemple 2G (deuxième génération de communications mobiles sans fil).

Commande	Paramètres	Valeur renvoyée
GetNetworkGeneration	Aucune	`-1` : pas de prise en charge. `0` : génération inconnue de réseau de communications mobiles sans fil `1` : 2G `2` : 2.5G `3` : 3G

Disponibilité
Flash Lite 2.0

Exemple
L'exemple suivant indique la syntaxe à utiliser pour renvoyer la génération du réseau :

```
status = fscommand2("GetNetworkGeneration");
```

Commande fscommand2 GetNetworkName

`GetNetworkName`

Définit un paramètre reprenant le nom du réseau actif.

Remarque : Cette commande n'est pas prise en charge pour les périphériques BREW.

Commande	Paramètres	Valeur renvoyée
GetNetworkName	`networkName` Chaîne représentant le nom du réseau. Il peut s'agir du nom d'une variable ou d'une chaîne qui contient le nom d'une variable. Si le réseau est enregistré et si son nom peut être déterminé, la variable `networkname` est définie sur le nom du réseau. Sinon, cette chaîne reste vide.	-1 : pas de prise en charge. `0` : aucun réseau n'est enregistré. `1` : le réseau est enregistré, mais le nom de réseau est inconnu. `2` : le réseau est enregistré et le nom de réseau est connu.

Disponibilité
Flash Lite 1.1

Exemple
L'exemple suivant affecte le nom du réseau actuel au paramètre `myNetName` et une valeur d'état à la variable `netNameStatus` :

```
netNameStatus = fscommand2("GetNetworkName", myNetName);
```

Commande fscommand2 GetNetworkRequestStatus

`GetNetworkRequestStatus`

Renvoie une valeur indiquant l'état de la requête HTTP la plus récente.

Remarque : Cette commande n'est pas prise en charge pour les périphériques BREW.

Commande	Paramètres	Valeur renvoyée
GetNetworkRequestStatus	Aucune	`-1` : la commande n'est pas prise en charge. `0` : une requête est en attente, une connexion réseau a été établie, le nom d'hôte du serveur a été résolu et une connexion au serveur a été établie. `1` : une requête est en attente et une connexion réseau est en cours d'établissement. `2` : une requête est en attente, mais la connexion réseau n'a pas encore été établie. `3` : une requête est en attente, une connexion réseau a été établie et le nom d'hôte du serveur est en cours de résolution. `4` : la requête a échoué à cause d'une erreur réseau. `5` : la requête a échoué en raison d'un échec de connexion au serveur. `6` : le serveur a renvoyé une erreur HTTP (par exemple, 404). `7` : la requête a échoué en raison de l'échec de l'accès au DNS ou lors de la résolution du nom de serveur. `8` : la requête a été complétée avec succès. `9` : la requête a échoué à cause du dépassement du délai. `10` : la requête n'a pas encore été créée.

Disponibilité
Flash Lite 1.1

Exemple
L'exemple suivant affecte l'état de la requête HTTP la plus récente à la variable `requesttatus`, puis utilise une instruction `switch` pour mettre à jour un champ de texte avec cet état :

```
requeststatus = fscommand2("GetNetworkRequestStatus");
switch (requeststatus) {
    case -1:
    _root.myText += "requeststatus not supported" + "\n";
    break;
    case 0:
    _root.myText += "connection to server has been made" + "\n";
    break;
    case 1:
    _root.myText += "connection is being established" + "\n";
    break;
    case 2:
    _root.myText += "pending request, contacting network" + "\n";
    break;
    case 3:
    _root.myText += "pending request, resolving domain" + "\n";
    break;
    case 4:
    _root.myText += "failed, network error" + "\n";
    break;
    case 5:
    _root.myText += "failed, couldn't reach server" + "\n";
    break;
    case 6:
    _root.myText += "HTTP error" + "\n";
    break;
    case 7:
    _root.myText += "DNS failure" + "\n";
    break;
    case 8:
    _root.myText += "request has been fulfilled" + "\n";
    break;
    case 9:
    _root.myText += "request timedout" + "\n";
    break;
    case 10:
    _root.myText += "no HTTP request has been made" + "\n";
    break;
}
```

Commande fscommand2 GetNetworkStatus

`GetNetworkStatus`

Renvoie une valeur indiquant l'état réseau du téléphone (indique si un réseau est enregistré et si le téléphone est en mode mobile).

Commande	Paramètres	Valeur renvoyée
GetNetworkStatus	Aucune	`-1` : la commande n'est pas prise en charge. `0` : aucun réseau n'est enregistré. `1` : sur réseau domestique. `2` : sur réseau domestique étendu. `3` : mobile (en dehors du réseau domestique).

Disponibilité
Flash Lite 1.1

Exemple
L'exemple suivant affecte l'état de la connexion réseau à la variable `networkstatus`, puis utilise une instruction `switch` pour mettre à jour un champ texte avec cet état :

```
networkstatus = fscommand2("GetNetworkStatus");
switch(networkstatus) {
    case -1:
    _root.myText += "network status not supported" + "\n";
    break;
    case 0:
    _root.myText += "no network registered" + "\n";
    break;
    case 1:
    _root.myText += "on home network" + "\n";
    break;
    case 2:
    _root.myText += "on extended home network" + "\n";
    break;
    case 3:
    _root.myText += "roaming" + "\n";
    break;
}
```

Commande fscommand2 GetPlatform

`GetPlatform`

Définit un paramètre qui identifie la plate-forme actuelle, ce qui décrit de façon générale la classe du périphérique. Pour les périphériques disposant de systèmes d'exploitation ouverts, cet identificateur correspond généralement au nom et à la version du système d'exploitation.

Commande	Paramètres	Valeur renvoyée
GetPlatform	`platform` Chaîne devant recevoir l'identificateur de la plate-forme.	`-1` : pas de prise en charge. `0` : prise en charge.

Disponibilité
Flash Lite 1.1

Exemple
L'exemple suivant définit le paramètre `platform` en fonction de l'identificateur de la plate-forme actuelle :

```
status = fscommand2("GetPlatform", "platform");
```

Vous trouverez ci-dessous des exemples de résultats pour `platform` :

`506i`Téléphone 506i. `FOMA1`Téléphone FOMA1. `Symbian6.1_s60.1`Téléphone Symbian 6.1, Series 60 version 1. `Symbian7.0`Téléphone Symbian 7.0

Commande fscommand2 GetPowerSource

`GetPowerSource`

Renvoie une valeur qui indique si l'alimentation vient de la batterie ou d'une source externe.

Remarque : Cette commande n'est pas prise en charge pour les périphériques BREW.

Commande	Paramètres	Valeur renvoyée
GetPowerSource	Aucune	`-1` : pas de prise en charge. `0` : le périphérique est alimenté par la batterie. `1` : le périphérique est alimenté par une source externe.

Disponibilité
Flash Lite 1.1

Exemple
L'exemple suivant définit la variable `myPower` sur la valeur indiquant la source d'alimentation, ou sur `-1` s'il ne peut le faire :

```
myPower = fscommand2("GetPowerSource");
```

Commande fscommand2 GetSignalLevel

`GetSignalLevel`

Renvoie la force du signal actuel sous forme de valeur numérique.

Remarque : Cette commande n'est pas prise en charge pour les périphériques BREW.

Commande	Paramètres	Valeur renvoyée
GetSignalLevel	Aucune	`-1` : pas de prise en charge. Autres valeurs numériques : le niveau actuel du signal, compris entre `0` et la valeur maximale renvoyée par `GetMaxSignalLevel`.

Disponibilité
Flash Lite 1.1

Exemple
L'exemple suivant affecte la valeur du niveau du signal à la variable `sigLevel` :

```
sigLevel = fscommand2("GetSignalLevel");
```

Commande fscommand2 GetSoftKeyLocation

`GetSoftKeyLocation`

Renvoie une valeur qui indique l'emplacement des touches programmables sur le périphérique.

Commande	Paramètres	Valeur renvoyée
GetSoftKeyLocation	Aucune	-1 : pas de prise en charge. 0: touches programmables en haut. 1: touches programmables à gauche. 2: touches programmables en bas. 3: touches programmables à droite.

Disponibilité
Flash Lite 2.0

Exemple
L'exemple suivant définit la variable `status` pour désigner l'emplacement de la touche programmable ou sur `-1` si les touches programmables ne sont pas prises en charge sur le périphérique :

```
status = fscommand2("GetSoftKeyLocation");
```

Commande fscommand2 GetTotalPlayerMemory

`GetTotalPlayerMemory`

Renvoie la quantité totale de la mémoire heap, en kilo-octets, affectée à Flash Lite.

Commande	Paramètres	Valeur renvoyée
GetTotalPlayerMemory	Aucune	`-1` : pas de prise en charge. `0` ou valeur positive : nombre total de kilo-octets de la mémoire heap.

Disponibilité
Flash Lite 1.1

Exemple
L'exemple suivant définit la variable `status` sur la quantité totale de mémoire heap :

```
status = fscommand2("GetTotalPlayerMemory");
```

Commande fscommand2 GetVolumeLevel

`GetVolumeLevel`

Renvoie le niveau de volume actuel du périphérique sous forme de valeur numérique.

Commande	Paramètres	Valeur renvoyée
GetVolumeLevel	Aucune	`-1` : pas de prise en charge. Autres valeurs numériques : Le niveau de volume actuel, compris entre `0` et la valeur renvoyée par `fscommand2("GetMaxVolumeLevel")`.

Disponibilité
Flash Lite 1.1

Exemple
L'exemple suivant affecte le niveau actuel du volume à la variable volume :

```
volume = fscommand2("GetVolumeLevel");
trace (volume); // output: 50
```

Commande fscommand2 Quit

`Quit`

Entraîne l'arrêt du lecteur Flash Lite et ferme ce programme.

Cette commande est prise en charge uniquement lorsque Flash Lite est en cours d'exécution en mode autonome. Elle n'est pas prise en charge lorsque le lecteur s'exécute dans le contexte d'une autre application (par exemple, en tant que module externe dans un navigateur).

Commande	Paramètres	Valeur renvoyée
Quit	Aucune	`-1` : pas de prise en charge.

Disponibilité
Flash Lite 1.1

Exemple
L'exemple suivant oblige Flash Lite à arrêter la lecture et quitter lorsqu'il s'exécute en mode autonome :

```
status = fscommand2("Quit");
```

Commande fscommand2 ResetSoftKeys

`ResetSoftKeys`

Rétablit les valeurs d'origine des touches programmables.

Cette commande est prise en charge uniquement lorsque Flash Lite est en cours d'exécution en mode autonome. Elle n'est pas prise en charge lorsque le lecteur s'exécute dans le contexte d'une autre application (par exemple, en tant que module externe dans un navigateur).

Commande	Paramètres	Valeur renvoyée
ResetSoftKeys	Aucune	`-1` : pas de prise en charge.

Disponibilité
Flash Lite 1.1

Exemple
L'instruction suivante rétablit les valeurs d'origine des touches programmables.

```
status = fscommand2("ResetSoftKeys");
```

Commande fscommand2 SetFocusRectColor

`SetFocusRectColor`

Définit la couleur du rectangle de focus sur une autre couleur.

La plage acceptable des valeurs pour le rouge, le vert et le bleu vont de 0 à 255. Pour Flash, vous ne pouvez pas modifier la couleur par défaut du rectangle de focus, qui est le jaune.

Commande	Paramètres	Valeur renvoyée
SetFocusRectColor	Aucune	`-1` : pas de prise en charge. `0`: indéterminable. `1` : succès

Disponibilité
Flash Lite 2.0

Exemple
L'exemple suivant redéfinit la couleur du rectangle de focus :

```
status = fscommand2("SetFocusRectColor, <red>, <green>, <blue>);
```

Commande fscommand2 SetInputTextType

`SetInputTextType`

Spécifie le mode d'ouverture du champ texte de saisie.

Flash Lite prend en charge la fonctionnalité de saisie de texte en demandant à l'application hôte d'activer l'interface de saisie de texte propre au périphérique, généralement appelée processeur frontal (FEP - front-end processor). Lorsque la commande `SetInputTextType` n'est pas utilisée, le mode par défaut du FEP est ouvert.

Commande	Paramètres	Valeur renvoyée
SetInputTextType	`variableName` Nom du champ texte de saisie. Il peut s'agir du nom d'une variable ou d'une chaîne qui contient le nom d'une variable. **Remarque :** Le nom d'une variable de champ texte est différent de son nom d'occurrence. Vous pouvez spécifier un nom de variable de champ texte dans la zone de texte Var de l'inspecteur des propriétés ou à l'aide du code ActionScript. Par exemple, le code suivant limite la saisie aux caractères numériques pour l'occurrence de champ texte (`numTxt`) dont le nom de variable est « `numTxt_var` ». `var numTxt:TextField;numTxt.variable = "numTxt_var";fscommand2("SetInputTextType", "numTxt_var", "Numeric");` `type` Une des valeurs `Numeric, Alpha`, `Alphanumeric`, `Latin`, `NonLatin` ou `NoRestriction`.	`0` : échec. `1` : succès

Le tableau suivant affiche les effets des différents modes, ainsi que les modes substitués :

Mode InputTextType	**Définit le processeur frontal sur l'un de ces modes, qui s'excluent mutuellement.**	**Si le mode retenu n'est pas pris en charge sur le périphérique actif, le processeur s'ouvre dans ce mode**
Numeric	Nombres uniquement (0 à 9)	Alphanumérique
Alpha	Caractères alphabétiques uniquement (A à Z, a à z)	Alphanumérique
Alphanumérique	Caractères alphanumériques uniquement (0à 9, A à Z, a à z)	Latin
Latin	Caractères latins uniquement (alphanumérique et ponctuation)	NoRestriction
NonLatin	Caractères non latin uniquement (par exemple, Kanji et Kana)	NoRestriction
NoRestriction	Mode par défaut (ne définit par de restriction sur le processeur frontal)	S/O
REMARQUE : Tous les téléphones mobiles ne prennent pas en charge les types de champ texte de saisie suivants. Pour cette raison, vous devez valider les données du texte saisi.		

Disponibilité

Flash Lite 1.1

Exemple

La ligne de code suivante définit le type de texte de saisie du champ associé à la variable `input1` devant recevoir les données numériques :

```
status = fscommand2("SetInputTextType", "input1", "Numeric");
```

Commande fscommand2 SetSoftKeys

`SetSoftKeys`

Reconfigure les touches programmables d'un périphérique mobile.

Lorsque l'utilisateur appuie sur une touche programmable, tout code ActionScript associé à l'événement softkey est exécuté. Le lecteur Flash Lite exécute cette fonction immédiatement lorsqu'elle est appelée. Cette commande est prise en charge uniquement lorsque Flash Lite est en cours d'exécution en mode autonome. Elle n'est pas prise en charge lorsque le lecteur s'exécute dans le contexte d'une autre application (par exemple, en tant que module externe dans un navigateur).

Pour la compatibilité ascendante avec Flash Lite 1.1, la touche programmable `SOFT1` est toujours associée à la touche gauche du combiné, puis la touche programmable `SOFT2` est toujours associée à la touche droite du combiné. A partir de la touche programmable `SOFT3`, les emplacements dépendent du combiné.

Les arguments de cette commande spécifient le texte à afficher pour les touches programmables correspondantes. Lorsque la commande `SetSoftKeys` s'exécute, le fait d'appuyer sur la touche gauche génère un événement de pression de touche `SOFT1` et le fait d'appuyer sur la touche droite génère un évènement de pression de touche `SOFT2`. Les touches programmables allant de `SOFT3` à `SOFT12` génèrent leurs propres événements.

Remarque : La reconfiguration des touches programmables dépend du périphérique mobile. Consultez la documentation du fabricant du périphérique pour déterminer si la reconfiguration est prise en charge.

Commande	Paramètres	Valeur renvoyée
SetSoftKeys	`soft1` Le texte à associer à la touche programmable `SOFT1`. `soft2` Le texte à associer à la touche programmable `SOFT2`. Ces paramètres peuvent être soit des noms de variable, soit des valeurs constantes de type chaîne (par exemple, "`Previous`").	-1 : pas de prise en charge. 0 : prise en charge.

Disponibilité
Flash Lite 1.1

Exemple
L'exemple suivant nomme la touche programmable `SOFT1` « Previous » et la touche programmable `SOFT2` « Next » :

```
status = fscommand2("SetSoftKeys", "Previous", "Next");
```

Vous pouvez définir des variables ou utiliser des valeurs de chaîne constantes de type chaîne pour chaque touche programmable :

```
status = fscommand2("SetSoftKeys", soft1, soft2, [soft3], [soft4], ..., [softn])
```

Remarque : Vous pouvez définir une touche programmable sans définir les autres. Ces exemples illustrent la syntaxe permettant d'associer un comportement à une touche programmable spécifique sans affecter les autres touches :

- Pour définir la touche programmable de gauche sur "soft1" et celle de droite sur rien :

  ```
  status = fscommand2("SetSoftKeys", "soft1", "")
  ```

- Pour ne pas modifier l'étiquette de la touche programmable de gauche et définir la touche programmable de droite sur « soft2 » :

  ```
  status = fscommand2("SetSoftKeys", undefined, "soft2")
  ```

- Pour ne pas modifier l'étiquette de la touche programmable de gauche et définir la touche programmable de droite sur « soft2 » :

  ```
  status = fscommand2("SetSoftKeys", null, "soft2")
  ```

- Pour définir la touche programmable de gauche sur « soft1 » et conserver la touche de droite telle quelle :

  ```
  status = fscommand2("SetSoftKeys", "soft1")
  ```

Commande fscommand2 StartVibrate

`StartVibrate`

Active la fonctionnalité de vibration du téléphone.

Si une vibration se produit déjà, Flash Lite arrête cette vibration avant de passer à la suivante. Les vibrations s'arrêtent également lorsque la lecture de l'application Flash s'arrête ou est interrompue, et lorsque le lecteur Flash Lite s'arrête.

Commande	Paramètres	Valeur renvoyée
StartVibrate	`time_on` Durée, en millisecondes (5 secondes au maximum), d'activation de la vibration. `time_off` Durée, en millisecondes (5 secondes au maximum), de désactivation de la vibration. `repeat` Nombre de répétitions (3 au maximum) de la vibration.	`-1` : pas de prise en charge. `0` : la vibration a été activée. `1` : une erreur s'est produite et la vibration n'a pas pu être activée.

Disponibilité

Flash Lite 1.1

Exemple

L'exemple suivant tente de lancer une séquence de vibrations de 2,5 secondes à 1 seconde d'intervalle avec deux répétitions. Il affecte une valeur à la variable `status` indiquant le succès ou l'échec de l'opération :

```
fscommand2("StartVibrate", 2500, 1000, 2);
```

Commande fscommand2 StopVibrate

`StopVibrate`

Arrête la vibration actuelle, si nécessaire.

Commande	Paramètres	Valeur renvoyée
StopVibrate	Aucune	`-1` : pas de prise en charge. `0` : la vibration s'est arrêtée.

Disponibilité

Flash Lite 1.1

Exemple

L'exemple suivant appelle `StopVibrate` et enregistre le résultat (non pris en charge ou vibration arrêtée) dans la variable `status` :

```
status = fscommand2("StopVibrate");
```

Chapitre 2 : Classes ActionScript

La documentation relative aux classes ActionScript inclut des informations sur la syntaxe, l'utilisation et des exemples de code concernant les méthodes, les propriétés et les événements appartenant à des classes spécifiques. Les classes sont énumérées par ordre alphabétique. Si vous ne savez pas à quelle classe une méthode, une propriété ou un événement appartient, consultez l'index.

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

Flash Lite 2.0

Voir aussi

`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

constructor (propriété Object.constructor), __proto__ (Object.__proto__, propriété) prototype (propriété Object.prototype), __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

Flash Lite 2.0

Voir aussi

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

Flash Lite 2.0

Voir aussi

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

Flash Lite 2.0

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 avec le type 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

Flash Lite 2.0

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
statique	CASEINSENSITIVE : Number	Représente le tri ne tenant pas compte de la casse.
statique	DESCENDING : Number	Représente un tri par ordre décroissant.
	length : Number	Un entier non négatif spécifiant le nombre d'éléments contenus dans le tableau.
statique	NUMERIC : Number	Représente un tri numérique et non pas en fonction des chaînes.
statique	RETURNINDEXEDARRAY : Number	Permet de renvoyer un tableau indexé suite à l'appel de la méthode `sort()` ou `sortOn()`.
statique	UNIQUESORT : Number	Représente le critère de tri unique.

Propriétés héritées de la classe Object

`constructor (propriété Object.constructor), __proto__ (Object.__proto__, propriété) prototype (propriété Object.prototype), __resolve (Object.__resolve, propriété)`

Récapitulatif des constructeurs

Signature	Description
Array ([valeur : Object])	Permet de créer un tableau.

Résumé de la méthode

Modificateurs	Signature	Description
	concat ([valeur : 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 (valeur : 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], [contrôle en amont : Number]) : Array	Trie les éléments d'un tableau.

Modificateurs	Signature	Description
	sortOn(fieldName:Object, [contrôle en amont:Object]) : Array	Trie les éléments d'un tableau selon un ou plusieurs champs du tableau.
	splice(startIndex:Number, [deleteCount:Number], [valeur: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(valeur:Object) : Number	Ajoute un ou plusieurs éléments au début d'un tableau et renvoie la nouvelle longueur du tableau.

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 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

Flash Lite 2.0

Paramètres

valeur : Object [facultatif] - Soit :

- Un entier spécifiant le nombre d'éléments contenus dans le tableau.
- Une liste de deux valeurs arbitraires au minimum. 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 aussi

Opérateur d'accès au tableau [], length (propriété Array.length)

CASEINSENSITIVE (propriété Array.CASEINSENSITIVE)

```
public static CASEINSENSITIVE : Number
```

Représente le tri ne tenant pas compte de 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

Flash Lite 2.0

Voir aussi

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
Flash Lite 2.0

Paramètres
valeur : 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.

Valeur renvoyée
Tableau - 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 l'indique 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)

```
public static DESCENDING : Number
```

Représente un tri par ordre décroissant. Vous pouvez utiliser cette constante pour le paramètre `options` de la méthode `sort()` ou `sortOn()`. La valeur de cette constante est 2.

Disponibilité
Flash Lite 2.0

Voir aussi
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
Flash Lite 2.0

Paramètres
delimiter : String [facultatif] - Caractère ou 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.

Valeur renvoyée
String - 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 une espace), puis à l'aide d'un tiret (-) et enfin à l'aide 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 aussi
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

Flash Lite 2.0

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,undefined,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)

```
public static NUMERIC : Number
```

Représente un tri numérique et non pas en fonction des chaînes. Le tri sur chaîne, qui constitue le paramètre par défaut, traite les nombres en tant que chaînes lors du tri. Ainsi, 10 vient avant 3. Le tri numérique traite les éléments en fonction de leur valeur numérique, et par conséquent 3 vient alors avant 10. Vous pouvez utiliser cette constante pour le paramètre `options` dans la méthode `sort()` ou `sortOn()`. La valeur de cette constante est 16.

Disponibilité

Flash Lite 2.0

Voir aussi

`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
Flash Lite 2.0

Valeur renvoyée
Object - Valeur du dernier élément dans le tableau spécifié.

Exemple
Le code suivant crée le tableau `myPets_` 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 aussi
`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
Flash Lite 2.0

Paramètres
valeur : Object - Une ou plusieurs valeurs à ajouter au tableau.

Valeur renvoyée
Number - 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 Sortie.

```
var myPets_array:Array = new Array("cat", "dog");
var pushed:Number = myPets_array.push("bird", "fish");
trace(pushed); // Displays 4.
```

Voir aussi
`pop (méthode Array.pop)`, `shift (méthode Array.shift)`, `unshift (méthode Array.unshift)`

RETURNINDEXEDARRAY (propriété Array.RETURNINDEXEDARRAY)

```
public static RETURNINDEXEDARRAY : Number
```

Permet de renvoyer un tableau indexé suite à 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
Flash Lite 2.0

Voir aussi
`sort (méthode Array.sort)`, `sortOn (méthode Array.sortOn)`

reverse (méthode Array.reverse)

```
public reverse() : Void
```

Inverse le tableau.

Disponibilité
Flash Lite 2.0

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
Flash Lite 2.0

Valeur renvoyée
Object - 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 aussi
`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

Flash Lite 2.0

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.

Valeur renvoyée

Array - 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

Flash Lite 2.0

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

contrôle en amont : 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.

Valeur renvoyée

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.
fruits_array.sort(Array.DESCENDING);
trace(fruits_array); // Displays 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:");
trace(passwords_array);
//Displays mom:glam,ana:ring,jay:mag,anne:home,regina:silly.
passwords_array.sort(order);
trace("Sorted:");
trace(passwords_array);
//Displays ana:ring,anne:home,jay:mag,mom:glam,regina:silly.
```

Voir aussi

`|, 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 *field*.

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 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 ».

Vous pouvez utiliser le paramètre `options` pour remplacer 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);
```

Disponibilité

Flash Lite 2.0

Paramètres

fieldName : Object - Chaîne identifiant un champ à utiliser en tant que valeur de tri ou tableau dans lequel le premier élément représente le champ de tri principal, le deuxième le champ de tri secondaire, etc.

contrôle en amont : Object [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. 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).

Valeur renvoyée

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

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 ci-dessous 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 aussi

`|, 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

Flash Lite 2.0

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.

valeur : Object [facultatif] - Spécifie les valeurs à insérer dans le tableau au point d'insertion défini dans le paramètre `startIndex`.

Valeur renvoyée

Array - 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

Flash Lite 2.0

Valeur renvoyée

String - 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 aussi

split (méthode String.split), join (méthode Array.join)

UNIQUESORT (propriété Array.UNIQUESORT)

```
public static UNIQUESORT : Number
```

Représente le critère de tri unique. 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

Flash Lite 2.0

Voir aussi

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

Flash Lite 2.0

Paramètres

valeur : Object - Un ou plusieurs nombres, éléments ou variables à insérer au début du tableau.

Valeur renvoyée

Number - Entier représentant la nouvelle longueur du tableau.

Exemple

L'exemple suivant illustre l'utilisation de la méthode `Array.unshift()` method :

```
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 aussi

pop (méthode Array.pop), push (méthode Array.push), shift (méthode Array.shift)

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 du lecteur Flash Lite. 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 une bitmap entièrement opaque ou entièrement transparente 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 0 à 255) décrivant les valeurs de transparence alpha et les valeurs de rouge, vert et bleu (ARVB) du pixel.

Les quatre canaux (rouge, vert, bleu et alpha) sont représentés sous forme de nombres lorsque vous utilisez la méthode `BitmapData.copyChannel()` ou 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 2 880 pixels.

Disponibilité

Flash Lite 3.1

Voir aussi

attachBitmap (méthode MovieClip.attachBitmap), beginFill (méthode MovieClip.beginFill)

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 : Boolean [lecture seule]	Largeur de l'image bitmap en pixels.

```
constructor (propriété Object.constructor), __proto__ (Object.__proto__, propriété),
prototype (propriété Object.prototype), __resolve (Object.__resolve, propriété)
```

Récapitulatif 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	Prend une image source et un objet filtre et génère l'image filtrée. Flash Lite 3.1 ne prend pas en charge les filtres. Cette méthode n'est donc pas prise en charge.
	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 BitmapData actuel vers un canal de l'objet BitmapData actuel.

Modificateurs	Signature	Description
	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.
	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 Lite.
	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	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. Flash Lite ne prend pas cette méthode en charge.
	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.
statique	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.

Modificateurs	Signature	Description
	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	Remplit une image avec des pixels représentant un bruit aléatoire. Flash Lite ne prend pas cette méthode en charge.
	paletteMap	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 Lite ne prend pas cette méthode en charge.
	perlinNoise	Génère une image de bruit Perlin. Flash Lite ne prend pas cette méthode en charge.
	pixelDissolve	Procède à la dissolution de pixels, soit d'une image source vers une image de destination, soit en utilisant la même image. Flash Lite ne prend pas cette méthode en charge.
	scroll	Fait défiler une image en fonction d'un certain montant en pixels (*x*, *y*). Flash Lite ne prend pas cette méthode en charge.
	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 les valeurs de couleur et transparence alpha d'un pixel unique d'un objet BitmapData.
	threshold	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. Flash Lite ne prend pas cette méthode en charge.

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 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'une bitmap opaque utilise uniquement 24 bits d'informations de canal de couleur. Si vous réglez 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 2 880 pixels. Si vous spécifiez une valeur de largeur ou de hauteur supérieure à 2880, la nouvelle occurrence n'est pas créée.

Disponibilité

Flash Lite 3.1

Paramètres

width : Number - 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 une bitmap entièrement transparente, réglez 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, copie du bitmap cloné. Un clone et l'objet cloné ont des propriétés identiques Néanmoins, un clone n'est pas évalué comme valeur égale de l'objet BitmapData qui a été cloné. En effet, les propriétés de l'objet d'origine sont une valeur transmise au clone, pas une référence transmise. Si vous modifiez les valeurs dans l'objet d'origine après la création du clone, le clone ne reçoit pas les nouvelles valeurs.

Disponibilité
Flash Lite 3.1

Valeur renvoyée
BitmapData - Nouvel objet BitmapData identique à l'original.

Exemple
L'exemple suivant crée trois objets BitmapData et les compare. Le code utilise le constructeur `BitmapData pour créer l'instance bitmap_1`. Il crée l'instance `bitmap_2`en lui attribuant une valeur égale à `bitmap_1`. Il créer l'instance `clonedBitmap` en clonant `bitmap_1`. Veuillez noter que `bitmap_2` est considéré comme égal à `bitmap_1` et que `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 " + (bitmap_1 == bitmap_2)); // true
trace("bitmap_1 == clonedBitmap " + (bitmap_1 == clonedBitmap)); // false

trace("-------------bitmap_1 properties-------------")
for(var i in bitmap_1) {
    trace(">> " + i + ": " + bitmap_1[i]);
}

trace("-------------bitmap_2 properties-------------")
for(var i in bitmap_2) {
    trace(">> " + i + ": " + bitmap_1[i]);
}

trace("-------------clonedBitmap properties-------------")
for(var i in clonedBitmap) {
    trace(">> " + i + ": " + clonedBitmap[i]);
}
```

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 en pixel de (1, 1) change la valeur en pixel pour `bitmap_2`. En effet, `bitmap_2` contient des références à `bitmap_1`. La modification de `bitmap_1` ne change pas clonedBitmap. En effet, `clonedBitmap` n'est pas rattaché aux valeurs dans `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.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 tout entière.

Disponibilité

Flash Lite 3.1

Paramètres

rect : Rectangle - Un objet Rectangle qui définit la zone de l'image dans laquelle l'objet ColorTransform est appliqué.

colorTransform : ColorTransform - 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.

```
fscommand2("SetSoftKeys");
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());

var myListener:Object = new Object ();
myListener.onKeyDown = function () {
    var keyCode = Key.getCode ();
    if (keyCode == ExtendedKey.SOFT1) {
        // Handle left soft key event
        myBitmapData.colorTransform(myBitmapData.rectangle, new ColorTransform(1, 0, 0, 1, 255, 0,
0, 0));
    }
};
```

Voir aussi

`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 BitmapData 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 des canaux source et de destination peut être l'une des valeurs suivantes ou la somme de n'importe laquelle de ces valeurs :

- `1` (rouge)
- `2` (vert)
- `4` (bleu)
- `8` (alpha)

Disponibilité

Flash Lite 3.1

Paramètres

sourceBitmap : 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 : Rectangle - 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 : Point - 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 - Canal source. Utilisez l'une des valeurs de l'ensemble (1,2,4,8), représentant respectivement les canaux rouge, vert, bleu et alpha ou la somme de ces valeurs.

destChannel : Number - Canal de destination. Utilisez l'une des valeurs de l'ensemble (1,2,4,8), représentant respectivement les canaux rouge, vert, bleu et alpha ou la somme de ces valeurs.

Exemple

L'exemple suivant indique comment copier un canal ARVB source à partir d'un objet BitmapData situé à un emplacement différent :

```
fscommand2("SetSoftKeys");
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());

var myListener:Object = new Object ();
myListener.onKeyDown = function () {
    var keyCode = Key.getCode ();
    if (keyCode == ExtendedKey.SOFT1) {
        // Handle left soft key event
        myBitmapData.copyChannel(myBitmapData, new Rectangle(0, 0, 50, 80), new Point(51, 0),
3, 1);
    }
};
```

Voir aussi

Rectangle (flash.geom.Rectangle)

copyPixels (méthode BitmapData.copyPixels)

```
public copyPixels(sourceBitmap:BitmapData, sourceRect:Rectangle, destPoint:Point) : 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.

Disponibilité

Flash Lite 3.1

Paramètres

sourceBitmap : BitmapData - Image bitmap d'entrée à partir de laquelle les pixels sont copiés. L'image source peut être une occurrence de BitmapData différente ou peut faire référence à l'occurrence de BitmapData actuelle.

sourceRect : Rectangle - Un rectangle qui définit la zone de l'image source à utiliser en tant qu'entrée.

destPoint : Point - Le point de destination représentant le coin supérieur gauche de la zone rectangulaire dans laquelle les nouveaux pixels sont placés.

Exemple

L'exemple suivant indique comment copier les pixels d'une occurrence BitmapData vers une autre.

```
fscommand2("SetSoftKeys");

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;

var myListener:Object = new Object ();
myListener.onKeyDown = function () {
    var keyCode = Key.getCode ();
    if (keyCode == ExtendedKey.SOFT1) {
        // Handle left soft key event
        bitmapData_2.copyPixels(bitmapData_1, new Rectangle(0, 0, 50, 80), new Point(51, 0));
    }
    else if (keyCode == ExtendedKey.SOFT2) {
        // Handle right soft key event
        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.

Appelez `myBitmapData.dispose()` pour définir la largeur et la hauteur de l'image sur 0. Après que la mémoire d'un objet BitmapData a été libérée, les appels d'accès à la méthode et la propriété sur l'instance échouent, renvoyant une valeur de -1.

Disponibilité
Flash Lite 3.1

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());

var myListener:Object = new Object ();
myListener.onKeyDown = function () {
    var keyCode = Key.getCode ();
    if (keyCode == ExtendedKey.SOFT1) {
        // Handle left soft key event
        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],
[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 Lite. Vous pouvez utiliser les objets Matrix, ColorTransform, BlendMode et un objet Rectangle de destination pour contrôler la qualité du rendu. Vous pouvez éventuellement indiquer si le bitmap doit être lissé lorsqu'il est redimensionné (cette opération ne fonctionne que 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.

Le paramètre `blendMode` n'est pas pris en charge dans Flash Lite.

Disponibilité

Flash Lite 3.1

Paramètres

source : Object - L'objet BitmapData à dessiner.

matrix : Matrix [facultatif] - 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 : ColorTransform [facultatif] - 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.

clipRect : Rectangle [facultatif] - Un objet Rectangle. Si cette valeur n'est pas fournie, aucun découpage n'est effectué.

smooth : Boolean [facultatif] - 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.

```
fscommand2("SetSoftKeys");

im          p  ort flash.displa y .BitmapData;
import flash.geom . R e c t a n g l  e ;
impor t  fl a sh. g  e  o m.M a tr i x;
import  flash .ge o  m.Colo rTrans form;
v ar myBi tmapDat a:Bit m apData  = new  BitmapDa ta(100,  80, fal  s e, 0x0 0C CCCCC);
 var mc_1: Mov i eCli p =  this .cre ateEmptyMo vieClip("m c ", this .getNextHighestDepth());
 mc_1.a tta chB itmap (my Bit mapDa ta,  th is . ge tNe xtH ig h es tDe pth () ) ;

va r m c _2:MovieClip  =  cr eat eRe  c t angle(50, 40, 0x FF0000);
mc _ 2 ._x = 101;

 va r myMatrix:Mat ri x = new Matrix ( );
myMatrix.ro tate(Math.PI/2) ;
var translate M a trix:Matrix = n ew  Mat rix();
tra  nsl ate Matrix .trans lat e(7 0, 15)  ;
my Matr ix. con cat  (transl at eMat rix );

var  myCol orTr ans  form: C olorTransfo rm  = new C o lorTransf orm (0, 0, 1 , 1, 0,  0 , 255, 0);
v ar blend Mode:String = "nor mal";
var m yRectangle:Rectangle = new Rectangle(0, 0, 100, 80);
var smooth:Boolean = true;
var myListener:Object = new Object ();
myListener.onKeyDown = function () {
    var keyCode = Key.getCode ();
    if (keyCode == ExtendedKey.SOFT1) {
        // Handle left soft key event
        myBitmapData.draw(mc_2, myMatrix, myColorTransform, blendMode, myRectangle, smooth);
    }
};
Key.addListener (myListener);
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
Flash Lite 3.1

Paramètres
rect : Rectangle - Zone rectangulaire à remplir.

color : Number - 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.

```
fscommand2("SetSoftKeys");

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());

var myListener:Object = new Object ();
myListener.onKeyDown = function () {
    var keyCode = Key.getCode ();
    if (keyCode == ExtendedKey.SOFT1) {
        // Handle left soft key event
        myBitmapData.fillRect(new Rectangle(0, 0, 50, 40), 0x00FF0000);
    }
};
```

Voir aussi
`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
Flash Lite 3.1

Paramètres
x : Number - Coordonnée *x* de l'image.

y : Number - Coordonnée *y* de l'image.

color : Number - 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.

```
fscommand2("SetSoftKeys");

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);

var myListener:Object = new Object ();
myListener.onKeyDown = function () {
    var keyCode = Key.getCode ();
    if (keyCode == ExtendedKey.SOFT1) {
        // Handle left soft key event
        myBitmapData.floodFill(_xmouse, _ymouse, 0x000000FF);}
};
```

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 `(valeur et masque) sont recherchées dans l'image toute entière.= color`. Pour déterminer les espaces blancs autour d'une image, utilisez `{mask: 0xFFFFFFFF, color: 0xFFFFFFFF}` pour rechercher les bornes des pixels autres que blanc.

Disponibilité

Flash Lite 3.1

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 limites dans lesquelles cette couleur n'existe pas dans une image. La valeur par défaut est `true`.

Valeur renvoyée

Rectangle - 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 :

```
fscommand2("SetSoftKeys");

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);

var myListener:Object = new Object ();
myListener.onKeyDown = function () {
    var keyCode = Key.getCode ();
    if (keyCode == ExtendedKey.SOFT1) {
        // Handle left soft key event
        var colorBoundsRect:Rectangle = myBitmapData.getColorBoundsRect(0x00FFFFFF,
0x00FF0000, true);
trace(colorBoundsRect); // (x=0, y=0, w=50, h=40)
    }
};
Key.addListener (myListener);
```

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 Lite 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

Flash Lite 3.1

Paramètres

x : Number - Coordonnée *x* du pixel.

y : Number - Coordonnée *y* du pixel.

Valeur renvoyée

Number - Nombre représentant une valeur de pixels RVB. Si les coordonnées (*x*, *y*) se trouvent à l'extérieur des limites de l'image, 0 est renvoyé.

Exemple

L'exemple suivant utilise la méthode `getPixel()` pour récupérer la valeur RVB d'un pixel à des emplacements *x* et *y* spécifiques.

```
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 aussi

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

Flash Lite 3.1

Paramètres

x : Number - Coordonnée *x* du pixel.

y : Number - Coordonnée *y* du pixel.

Valeur renvoyée

Number - Nombre représentant une valeur de pixels ARVB. Si les coordonnées (*x*, *y*) se trouvent à l'extérieur des limites de l'image, 0 est renvoyé. 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 des emplacements *x* et *y* spécifiques :

```
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 aussi

getPixel (méthode BitmapData.getPixel)

height (propriété BitmapData.height)

```
public height : Number [read-only]
```

La hauteur de l'image bitmap en pixels.

Disponibilité

Flash Lite 3.1

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

Flash Lite 3.1

Paramètres

firstPoint : Point - 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 - Objet Rectangle, Point, ou BitmapData.

secondBitmapPoint : 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.

Valeur renvoyée

Boolean - Valeur booléenne. En cas de correspondance, renvoie une valeur `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 x:" + 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)

```
public 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.

Disponibilité

Flash Lite 3.1

Paramètres

id : String - Un ID de liaison d'un symbole dans la bibliothèque.

Valeur renvoyée

BitmapData - 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:Rectangle,
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
Flash Lite 3.1

Paramètres
sourceBitmap : 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 : Rectangle - Un rectangle qui définit la zone de l'image source à utiliser en tant qu'entrée.

destPoint : Point - Le point sur l'image de destination (l'occurrence BitmapData actuelle) correspondant au coin supérieur gauche du rectangle source.

redMult : Number - Nombre par lequel la valeur de canal de rouge doit être multipliée.

greenMult : Number - Nombre par lequel la valeur de canal de vert doit être multipliée.

blueMult : Number - Nombre par lequel la valeur de canal de bleu doit être multipliée.

alphaMult : Number - Nombre par lequel la valeur de transparence alpha doit être multipliée.

Exemple
L'exemple suivant indique comment fusionner deux parties d'une occurrence `BitmapData`.

```
fscommand2("SetSoftKeys");

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;

var myListener:Object = new Object ();
myListener.onKeyDown = function () {
    var keyCode = Key.getCode ();
    if (keyCode == ExtendedKey.SOFT1) {
        // Handle left soft key event
        bitmapData_1.merge(bitmapData_2, new Rectangle(0, 0, 50, 40), new Point(25, 20), 128,
0, 0, 0);
    }
};
Key.addListener (myListener);
```

rectangle (propriété BitmapData.rectangle)

```
public rectangle : Rectangle [read-only]
```

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

Flash Lite 3.1

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)
```

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
Flash Lite 3.1

Paramètres
x : Number - Coordonnée *x* du pixel dont la valeur change.

y : Number - Coordonnée *y* du pixel dont la valeur change.

color : Number - La couleur RVB sur laquelle le pixel va être défini.

Voir aussi
`getPixel (méthode BitmapData.getPixel)`, `setPixel32 (méthode BitmapData.setPixel32)`

setPixel32 (méthode BitmapData.setPixel32)

```
public setPixel32(x:Number, y:Number, color:Number) : Void
```

Définit les valeurs de couleur et 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
Flash Lite 3.1

Paramètres
x : Number - Coordonnée *x* du pixel dont la valeur change.

y : Number - 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.

Voir aussi
`getPixel32 (méthode BitmapData.getPixel32)`, `setPixel (méthode BitmapData.setPixel)`

transparent (propriété BitmapData.transparent)

```
public transparent : Boolean [read-only]
```

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
Flash Lite 3.1

width (propriété BitmapData.width)

```
public width : Number [read-only]
```

Largeur de l'image bitmap en pixels.

Disponibilité

Flash Lite 3.1

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 booléen avant d'appeler ses méthodes.

Disponibilité

Flash Lite 2.0

Résumé des propriétés

Propriétés héritées de la classe Object

constructor (propriété Object.constructor), __proto__ (Object.__proto__, propriété) prototype (propriété Object.prototype), __resolve (Object.__resolve, propriété)

Récapitulatif des constructeurs

Signature	Description
Boolean([valeur: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
Flash Lite 2.0

Paramètres
valeur : 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
Flash Lite 2.0

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
Flash Lite 2.0

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
```

Bouton

```
Object
    |
    +-Button
```

```
public class Button
extends Object
```

Tous les symboles de boutons sont des instances de l'objet Bouton. Vous pouvez donner un nom d'occurrence à un bouton dans l'inspecteur 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.

Disponibilité
Flash Lite 2.0

Voir aussi
Object

Résumé des propriétés

Modificateurs	Propriété	Description
	_alpha : Number	La valeur de transparence alpha du bouton.
	enabled : Boolean	Une valeur booléenne spécifiant si un bouton est activé.
	_focusrect : Boolean	Une valeur booléenne indiquant si un bouton est entouré d'un rectangle jaune lorsqu'il a le focus d'entrée.
	_height : Number	La hauteur du bouton, en pixels.
	_highquality : Number	**Déconseillé depuis Flash Player 7.** Cette propriété est déconseillée au profit de `Button._quality`. Spécifie le niveau d'anti-aliasing appliqué au fichier SWF actuel.

Modificateurs	Propriété	Description
	_name : String	Nom d'instance du bouton.
	_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.
	_soundbuftime : Number	Spécifie le nombre de secondes pendant lequel les sons sont chargés en mémoire tampon avant d'être diffusés en continu.
	tabEnabled : Boolean	Spécifie si un bouton 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]	Renvoie le chemin cible de l'instance de bouton.
	trackAsMenu : Boolean	Valeur booléenne indiquant si d'autres boutons ou clips peuvent recevoir un événement de relâchement de la souris ou du stylet.
	_url : String [lecture seule]	Récupère l'URL du fichier SWF qui a créé le bouton.
	_visible : Boolean	Une valeur booléenne indiquant si le bouton 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]	Renvoie 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

```
constructor (propriété Object.constructor), __proto__ (Object.__proto__,
propriété) prototype (propriété Object.prototype), __resolve (Object.__resolve,
propriété)
```

Résumé des événements

Evénement	Description
onDragOut = fonction() {}	Cet événement est appelé lorsque l'utilisateur clique sur le bouton puis fait glisser le pointeur au dessus de la zone occupée par le bouton.
onDragOver = fonction() {}	Cet événement est appelé lorsque l'utilisateur clique en dehors de la zone occupée par le bouton puis fait glisser le pointeur au-dessus du bouton.
onKeyDown = fonction() {}	Invoqué lorsqu'un bouton reçoit le focus clavier et lorsque l'utilisateur appuie sur une touche.
onKeyUp = fonction() {}	Cet événement est appelé lorsque le bouton a le focus d'entrée et l'utilisateur relâche une touche.
onKillFocus = function(newFocus: Object) {}	Invoqué lorsqu'un bouton perd le focus clavier.
onPress = fonction() {}	Invoqué lorsqu'un bouton est enfoncé.
onRelease = fonction() {}	Invoqué lorsqu'un bouton est relâché.
onReleaseOutside = fonction() {}	Invoqué lorsque l'utilisateur relâche la souris tandis que le pointeur se trouve hors du bouton après avoir appuyé sur le bouton de la souris lorsque le pointeur se trouvait à l'intérieur du bouton.
onRollOut = fonction() {}	Invoqué lorsqu'un bouton perd le focus.
onRollOver = fonction() {}	Invoqué lorsqu'un bouton reprend le focus.
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 (Button._alpha, propriété)

```
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
Flash Lite 2.0

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 aussi
`_alpha (MovieClip._alpha, propriété)`, `_alpha (propriété TextField._alpha)`

enabled (propriété Button.enabled)

```
public enabled : Boolean
```

Une valeur booléenne spécifiant si un bouton est activé. Lorsqu'il est désactivé (la propriété enabled est alors réglée sur `false`), le bouton est visible mais vous ne pouvez pas cliquer dessus. 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
Flash Lite 2.0

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 occurrences de boutons sur la scène. Ensuite, attribuez leur les noms d'occurrence `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 );
};
```

_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 d'entrée. 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 et 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`.

Remarque : Pour le lecteur Flash Lite 2.0, lorsque la propriété `_focusrect` est désactivée (en d'autres termes, `Button.focusRect is false`), le bouton reçoit tous les événements. Ce comportement est différent de celui de Flash Lite Player, car lorsque la propriété `_focusrect` est désactivée, le bouton reçoit les événements `rollOver` et `rollOut`, mais pas `press` et `release`.

D'autre part, pour Flash Lite 2.0, vous pouvez modifier la couleur du rectangle de focus à l'aide de la commande fscommand2 `SetFocusRectColor`. Ce comportement diffère également de Flash Lite Player, où la couleur du rectangle de focus est limitée au jaune.

Disponibilité
Flash Lite 2.0

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;
```

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
Flash Lite 2.0

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 aussi

`getDepth (méthode MovieClip.getDepth)`, `getDepth (méthode TextField.getDepth)`, `getInstanceAtDepth (méthode MovieClip.getInstanceAtDepth)`

_height (propriété Button._height)

```
public _height : Number
```

La hauteur du bouton, en pixels.

Disponibilité

Flash Lite 2.0

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 est déconseillée au profit 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

Flash Lite 2.0

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 aussi

_quality (Button._quality, propriété), _quality, propriété

_name (propriété Button._name)

```
public _name : String
```

Nom d'occurrence du bouton spécifié par `my_btn`.

Disponibilité

Flash Lite 2.0

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() {}
```

Cet événement est appelé lorsque l'utilisateur appuie sur le bouton de la souris, puis fait glisser le pointeur au dessus de la zone occupée par le bouton. Vous devez définir une fonction qui s'exécute lorsque le gestionnaire d'événements est appelé.

Remarque : Le gestionnaire d'événements `onDragOut` n'est pris en charge pour Flash Lite 2.0 que si `System.capabilities.hasMouse a pour valeur true` ou System.capabilities.hasStylus a pour valeur true .

Disponibilité

Flash Lite 2.0

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() {}
```

Cet événement est appelé lorsque l'utilisateur appuie sur le bouton de la souris, puis fait glisser le pointeur en dehors de la zone occupée par le bouton. Vous devez définir une fonction qui s'exécute lorsque le gestionnaire d'événements est appelé.

Remarque : Le gestionnaire d'événements `onDragOver` n'est pris en charge pour Flash Lite 2.0 que si `System.capabilities.hasMouse a pour valeur true` ou System.capabilities.hasStylus a pour valeur true .

Disponibilité

Flash Lite 2.0

Exemple

L'exemple suivant définit une fonction pour le gestionnaire onDragOver qui envoie une instruction `trace()` au panneau 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 Sortie suit vos mouvements.

Voir aussi

`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

Flash Lite 2.0

Exemple

Dans l'exemple suivant, une fonction qui envoie du texte vers le panneau 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;
}
```

Choisissez 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 de tabulation 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 Sortie.

Voir aussi

`onKeyUp (gestionnaire Button.onKeyUp)`, `getAscii (méthode Key.getAscii)`, `getCode (méthode Key.getCode)`

onKeyUp (gestionnaire Button.onKeyUp)

```
onKeyUp = function() {}
```

Cet événement est appelé lorsque le bouton a le focus d'entrée et 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

Flash Lite 2.0

Exemple

Dans l'exemple suivant, une fonction qui envoie du texte vers le panneau 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()+")");
};
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 de tabulation 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 Sortie.

Voir aussi

`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

Flash Lite 2.0

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 texte `output_txt`.

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
Flash Lite 2.0

Exemple
Dans l'exemple suivant, une fonction envoyant une instruction `trace()` vers le panneau 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
Flash Lite 2.0

Exemple
Dans l'exemple suivant, une fonction envoyant une instruction `trace()` vers le panneau 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 le bouton de la souris 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.

Remarque : Le gestionnaire d'événements `onReleaseOutside` n'est pris en charge pour Flash Lite 2.0 que si `System.capabilities.hasMouse a pour valeur true` ou System.capabilities.hasStylus a pour valeur true .

Disponibilité
Flash Lite 2.0

Exemple

Dans l'exemple suivant, une fonction envoyant une instruction `trace()` vers le panneau Sortie est définie pour le gestionnaire `onReleaseOutside` :

```
my_btn.onReleaseOutside = function () {
    trace ("onReleaseOutside called");
};
```

onRollOut (gestionnaire Button.onRollOut)

```
onRollOut = function() {}
```

Invoqué lorsqu'un bouton perd le focus. Cette situation risque de se produire lorsque l'utilisateur clique sur un autre bouton ou en dehors du bouton sélectionné. Vous devez définir une fonction qui s'exécute lorsque le gestionnaire d'événements est appelé.

Disponibilité

Flash Lite 2.0

Exemple

Dans l'exemple suivant, une fonction envoyant une instruction `trace()` vers le panneau Sortie est définie pour le gestionnaire `onRollOut` :

```
my_btn.onRollOut = function () {
    trace ("onRollOut called");
};
```

onRollOver (gestionnaire Button.onRollOver)

```
onRollOver = function() {}
```

Invoqué lorsqu'un bouton reprend le focus. Cette situation risque de se produire lorsque l'utilisateur clique sur un autre bouton, en dehors du bouton sélectionné. 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

Flash Lite 2.0

Exemple

Dans l'exemple suivant, une fonction envoyant une instruction `trace()` vers le panneau 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 de tabulation pour déplacer le focus d'entrée d'un champ texte vers un bouton, le paramètre `oldFocus` contient l'occurrence de champ texte.

Si aucun objet n'avait précédemment reçu le focus, le paramètre `oldFocus` contient une valeur null.

Disponibilité
Flash Lite 2.0

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 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
Flash Lite 2.0

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 Sortie affiche le code suivant :

```
_level0.my_mc
```

Voir aussi
_parent (propriété MovieClip._parent), _target (propriété MovieClip._target), _root, propriété

_quality (Button._quality, propriété)

```
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 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. 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 lissés lorsque l'animation est statique. Il s'agit du paramètre de qualité de rendu par défaut de Flash.

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
Flash Lite 2.0

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
Flash Lite 2.0

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 rond (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;
};
```

Voir aussi

_rotation (propriété MovieClip._rotation), _rotation (propriété TextField._rotation)

_soundbuftime (propriété Button._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 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 section `_soundbuftime`.

Disponibilité

Flash Lite 2.0

Voir aussi

_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

Flash Lite 2.0

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;
```

Voir aussi

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 affiché du fichier SWF contient une propriété `tabIndex`, l'ordre de tabulation automatique est désactivé et calculé à partir des propriétés `tabIndex` des objets du fichier SWF. L'ordre de tabulation personnalisé n'inclut que 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 dont la propriété `tabIndex` est définie sur `1` précède un objet dont la propriété `tabIndex` est définie sur `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 *flat*. 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

Flash Lite 2.0

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;
```

Voir aussi

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 [read-only]
```

Renvoie le chemin cible de l'occurrence de bouton spécifiée par `my_btn`.

Disponibilité

Flash Lite 2.0

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 aussi

_target (propriété MovieClip._target)

trackAsMenu (propriété Button.trackAsMenu)

```
public trackAsMenu : Boolean
```

Valeur booléenne indiquant si d'autres boutons ou clips peuvent recevoir un événement de relâchement de la souris ou du stylet. Si vous faites glisser un stylet ou la souris au-dessus d'un bouton, puis relâchez un deuxième bouton, l'événement `onRelease` est enregistré pour le deuxième bouton. Ceci permet de créer des menus pour le deuxième bouton. Vous pouvez définir la propriété `trackAsMenu` sur n'importe quel bouton ou objet clip. Si vous n'avez pas défini la propriété `trackAsMenu`, le comportement par défaut est `false`.

Vous pouvez modifier la propriété `trackAsMenu` à tout moment ; le bouton modifié accepte immédiatement le nouveau comportement.

Remarque : La propriété `trackAsMenu` n'est prise en charge pour Flash Lite 2.0 que si `System.capabilities.hasMouse` a pour valeur `true` ou `System.capabilities.hasStylus` a pour valeur `true`.

Disponibilité

Flash Lite 2.0

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");
};
```

Pour tester le fichier SWF, cliquez sur `one_btn` dans la scène, maintenez le bouton de la souris enfoncé, puis de relâchez-le 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 aussi

trackAsMenu (propriété MovieClip.trackAsMenu)

_url (propriété Button._url)

```
public _url : String [read-only]
```

Récupère l'URL du fichier SWF qui a créé le bouton.

Disponibilité

Flash Lite 2.0

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 Sortie.

_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
Flash Lite 2.0

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 aussi
`_visible (propriété MovieClip._visible)`, `_visible (propriété TextField._visible)`

_width (propriété Button._width)

```
public _width : Number
```

La largeur du bouton, en pixels.

Disponibilité
Flash Lite 2.0

Exemple
L'exemple suivant augmente la valeur de la propriété width d'un bouton intitulé `my_btn` et affiche la largeur dans le panneau Sortie. Entrez le code ActionScript suivant dans l'image 1 du scénario :

```
my_btn.onRelease = function() {
    trace(this._width);
    this._width ~= 1.1;
};
```

Voir aussi
`_width (propriété MovieClip._width)`

_x (Button._x, propriété)

```
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
Flash Lite 2.0

Exemple
L'exemple suivant définit sur 0 les coordonnées de `my_btn` 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 aussi
`_xscale (propriété Button._xscale)`, `_y (Button._y, propriété)`, `_yscale (propriété Button._yscale)`

_xmouse (propriété Button._xmouse)

```
public _xmouse : Number [read-only]
```

Renvoie la coordonnée *x* de la position de la souris par rapport au bouton.

Remarque : La propriété `_xmouse` n'est prise en charge pour Flash Lite 2.0 que si `System.capabilities.hasMouse` a pour valeur `true` ou `System.capabilities.hasStylus` a pour valeur `true`.

Disponibilité
Flash Lite 2.0

Exemple
L'exemple suivant affiche la coordonnée x 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 aussi

_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 paramétrage 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

Flash Lite 2.0

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 les axes *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 aussi

_x (Button._x, propriété), _y (Button._y, propriété), _yscale (propriété Button._yscale)

_y (Button._y, propriété)

```
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

Flash Lite 2.0

Exemple

L'exemple suivant définit sur 0 les coordonnées de my_btn 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 aussi

_x (Button._x, propriété), _xscale (propriété Button._xscale), _yscale (propriété Button._yscale)

_ymouse (propriété Button._ymouse)

```
public _ymouse : Number [read-only]
```

Renvoie la coordonnée *y* de la position de la souris par rapport au bouton.

Remarque : La propriété `_ymouse` n'est prise en charge pour Flash Lite 2.0 que si `System.capabilities.hasMouse` a pour valeur `true` ou `System.capabilities.hasStylus` a pour valeur `true`.

Disponibilité

Flash Lite 2.0

Exemple

L'exemple suivant affiche la coordonnée x 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 aussi

_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

Flash Lite 2.0

Exemple

L'exemple suivant redemandions un bouton intitulé `my_btn`. Lorsque vous cliquez sur le bouton et le relâchez, sa taille augmente de 10 % sur les axes *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 aussi

_y (Button._y, propriété), _x (Button._x, propriété), _xscale (propriété Button._xscale)

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 périphérique portable est différent de celui d'un ordinateur. 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 demander 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 porte sur une chaîne destinée à un périphérique portable :

- qui indique une orientation d'écran normale
- dont la langue d'exécution n'est pas déterminée
- qui s'exécute sous le système d'exploitation Symbian7.0sSeries60V2
- qui est configuré de façon à empêcher l'utilisateur d'accéder au disque dur, à la caméra ou au microphone
- qui utilise le lecteur Flash Lite sous sa version de publication officielle
- pour lequel le lecteur Flash Lite ne prend pas en charge le développement ou la lecture des applications de diffusion à l'écran par l'intermédiaire de Flash Media Server
- qui ne prend pas en charge l'impression sur le périphérique
- où le lecteur Flash Lite s'exécute sur un périphérique portable qui prend en charge la vidéo intégrée.

```
undefinedScreenOrientation=normal
language=xu
OS=Symbian7.0sSeries60V2
localFileReadDisable=true
avHardwareDisable=true
isDebugger=false
hasScreenBroadcast=false
hasScreenPlayback=false
hasPrinting=false
hasEmbeddedVideo=true
```

La plupart des propriétés de l'objet `System.capabilities` sont en lecture seule.

Disponibilité

Flash Lite 2.0

Résumé des propriétés

Modificateurs	Propriété	Description
statique	audioMIMETypes : Array [lecture seule]	Renvoie un tableau de types MIME pour les codecs audio pris en charge par un périphérique portable.
statique	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`).
statique	has4WayKeyAS : Boolean [lecture seule]	Une valeur booléenne qui est `true` si le lecteur Flash Lite exécute le code ActionScript associé aux gestionnaires d'événements de touche qui sont associés aux touches Gauche, Droite, Haut et Bas.
statique	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 Lite Player et les options d'accessibilité ; sinon définie sur `false`.
statique	hasAudio : Boolean [lecture seule]	Spécifie si le système est doté de fonctionnalités audio.
statique	hasAudioEncoder : Boolean [lecture seule]	Spécifie si Flash Lite Player peut coder un flux audio.
statique	hasCMIDI : Boolean [lecture seule]	Renvoie `true` si le périphérique portable est capable de lire les données audio au format CMIDI.
statique	hasCompoundSound : Boolean [lecture seule]	Renvoie `true` si le lecteur Flash Lite est capable de traiter les données audio composites.
statique	hasDataLoading : Boolean [lecture seule]	Renvoie `true` si le lecteur Flash Lite est capable charger de façon dynamique des données supplémentaires par l'intermédiaire d'appels à des fonctions spécifiques.
statique	hasEmail : Boolean [lecture seule]	Renvoie `true` si le lecteur Flash Lite est capable d'envoyer des messages électroniques avec la commande `GetURL` d'ActionScript.
statique	hasEmbeddedVideo : Boolean [lecture seule]	Une valeur booléenne qui indique si le périphérique portable prend en charge la vidéo intégrée.
statique	hasMappableSoftKeys : Boolean	Renvoie `true` si le périphérique portable permet de réinitialiser ou affecter de nouveau des étiquettes de touches programmables et de gérer les événements provenant de ces touches programmables.
statique	hasMFI : Boolean [lecture seule]	Renvoie `true` si le périphérique portable est capable de lire les données audio au format MFI.
statique	hasMIDI : Boolean [lecture seule]	Renvoie `true` si le périphérique portable est capable de lire les données audio au format MIDI.
statique	hasMMS : Boolean [lecture seule]	Renvoie `true` si le périphérique portable est capable d'envoyer des messages MMS avec la commande `GetURL` d'ActionScript.
statique	hasMouse : Boolean [lecture seule]	Indique si le périphérique portable envoie les événements liés à la souris au lecteur Flash Lite.

Modificateurs	Propriété	Description
statique	hasMP3 : Boolean [lecture seule]	Indique si le périphérique portable est doté d'un décodeur MP3.
statique	hasPrinting : Boolean [lecture seule]	Valeur booléenne définie sur `true` si le lecteur s'exécute sur un périphérique portable qui prend en charge l'impression ; définie sur `false` sinon.
statique	hasQWERTYKeyboard : Boolean [lecture seule]	Renvoie `true` si le lecteur Flash Lite peut traiter le code ActionScript associé à l'ensemble des touches, ce qui inclut la touche RETOUR ARRIERE, figurant sur les claviers standard.
statique	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 Media Server ; définie sur `false` sinon.
statique	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 Media Server ; définie sur `false` sinon.
statique	hasSharedObjects : Boolean [lecture seule]	Renvoie `true` si le contenu Flash Lite lu par une application peut accéder à la version Flash Lite des objets partagés.
statique	hasSMAF : Boolean [lecture seule]	Renvoie `true` si le périphérique portable est capable de lire les données audio au format SMAF.
statique	hasSMS : Number [lecture seule]	Indique si le périphérique portable est capable d'envoyer des messages SMS avec la commande `GetURL` d'ActionScript.
statique	hasStreamingAudio : Boolean [lecture seule]	Valeur booléenne définie sur `true` si le lecteur peut lire les sons en flux continu ; sinon définie sur `false`.
statique	hasStreamingVideo : Boolean [lecture seule]	Valeur booléenne qui indique si le lecteur peut lire les vidéos en flux continu.
statique	hasStylus : Boolean [lecture seule]	Indique si le périphérique portable prend en charge les événements relatifs au stylet.
statique	hasVideoEncoder : Boolean [lecture seule]	Spécifie si Flash Lite Player peut coder un flux vidéo.
statique	hasXMLSocket : Number [lecture seule]	Indique si l'application hôte prend en charge les sockets XML.
statique	imageMIMETypes : Array [lecture seule]	Renvoie un tableau qui contient tous les types MIME que la fonction `loadMovie` et les codecs du périphérique portable prennent en charge pour le traitement des images.
statique	isDebugger : Boolean [lecture seule]	Valeur booléenne indiquant si le lecteur est une version officielle (`false`) ou une version de débogage spéciale (`true`).
statique	language : String [lecture seule]	Indique la langue du système sur lequel s'exécute le lecteur.
statique	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`).
statique	MIMETypes : Array [lecture seule]	Renvoie un tableau qui contient tous les types MIME que la fonction `loadMovie` et les objets Sound et Video prennent en charge.
statique	os : String [lecture seule]	Chaîne indiquant le système d'exploitation actuel.
statique	screenOrientation : String [lecture seule]	Cette variable appartient à l'objet `System.capabilities` qui indique l'orientation actuelle de l'écran.

Modificateurs	Propriété	Description
statique	screenResolutionX : Number [lecture seule]	Entier indiquant la résolution horizontale maximale de l'écran.
statique	screenResolutionY : Number [lecture seule]	Entier indiquant la résolution verticale maximale de l'écran.
statique	softKeyCount : Number [lecture seule]	Indique le nombre de touches programmables pouvant être reconfigurées que le périphérique portable prend en charge.
statique	version : String [lecture seule]	Chaîne contenant la plate-forme Flash Lite Player et les informations sur la version (par exemple, "WIN 7,1,0,0").
statique	videoMIMETypes : Array [lecture seule]	Indique tous les types MIME pour la vidéo que les codecs du périphérique portable prennent en charge.

Propriétés héritées de la classe Object

```
constructor (propriété Object.constructor),__proto__ (Object.__proto__,
propriété)prototype (propriété Object.prototype),__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)
```

audioMIMETypes (propriété capabilities.audioMIMETypes)

```
public static audioMIMETypes : Array [read-only]
```

Renvoie un tableau de types MIME pour les codecs audio pris en charge par un périphérique portable.

Disponibilité

Flash Lite 2.0

Exemple

L'exemple suivant présente la valeur de cette propriété en lecture seule :

```
trace(System.capabilities.audioMIMETypes);
```

avHardwareDisable (propriété capabilities.avHardwareDisable)

```
public static avHardwareDisable : Boolean [read-only]
```

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`.

Remarque : Pour Flash Lite 2.0, la valeur renvoyée est toujours `true`.

Disponibilité
Flash Lite 2.0

Exemple
L'exemple suivant présente la valeur de cette propriété en lecture seule :

```
trace(System.capabilities.avHardwareDisable);
```

has4WayKeyAS (propriété capabilities.has4WayKeyAS)

```
public static has4WayKeyAS : Boolean [read-only]
```

Une valeur booléenne qui est `true` si le lecteur Flash Lite exécute le code ActionScript associé aux gestionnaires d'événements de touche qui sont associés aux touches Gauche, Droite, Haut et Bas. Dans le cas contraire, cette propriété renvoie `false`.

Si la valeur de cette variable est `true`, lorsque l'une des quatre touches de direction est sollicitée, le lecteur recherche d'abord un gestionnaire correspondant. S'il n'en trouve aucun, Flash procède à un contrôle de navigation. Cependant, si un gestionnaire d'événements est trouvé, aucune action de navigation ne se produit pour cette touche. En d'autres termes, la présence d'un gestionnaire de pression de touche pour la touche Bas désactive la fonctionnalité de navigation vers le bas.

Disponibilité
Flash Lite 2.0

Exemple
L'exemple suivant présente la valeur de cette propriété en lecture seule :

```
trace(System.capabilities.has4WayKeyAS);
```

hasAccessibility (propriété capabilities.hasAccessibility)

```
public static hasAccessibility : Boolean [read-only]
```

Valeur booléenne définie sur `true` si le lecteur s'exécute dans un environnement qui prend en charge la communication entre Flash Lite Player et les options d'accessibilité ; sinon définie sur `false`. La chaîne de serveur est `ACC`.

Remarque : Pour Flash Lite 2.0, la valeur renvoyée est toujours `false`.

Disponibilité
Flash Lite 2.0

Exemple
L'exemple suivant présente la valeur de cette propriété en lecture seule :

```
trace(System.capabilities.hasAccessibility);
```

hasAudio (propriété capabilities.hasAudio)

```
public static hasAudio : Boolean [read-only]
```

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
Flash Lite 2.0

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 [read-only]
```

Spécifie si Flash Lite Player peut coder un flux audio. 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`.

Remarque : Pour Flash Lite 2.0, la valeur renvoyée est toujours `false`.

Disponibilité
Flash Lite 2.0

Exemple
L'exemple suivant présente la valeur de cette propriété en lecture seule :

```
trace(System.capabilities.hasAudioEncoder);
```

hasCMIDI (propriété capabilities.hasCMIDI)

```
public static hasCMIDI : Boolean [read-only]
```

Renvoie `true` si le périphérique portable est capable de lire les données audio au format CMIDI. Dans le cas contraire, cette propriété renvoie `false`.

Disponibilité
Flash Lite 2.0

Exemple
L'exemple suivant présente la valeur de cette propriété en lecture seule :

```
trace(System.capabilities.hasCMIDI);
```

hasCompoundSound (propriété capabilities.hasCompoundSound)

```
public static hasCompoundSound : Boolean [read-only]
```

Renvoie `true` si le lecteur Flash Lite est capable de traiter les données audio composites. Sinon, elle renvoie la valeur `false`.

Disponibilité
Flash Lite 2.0

Exemple
L'exemple suivant présente la valeur de cette propriété en lecture seule :

```
trace(System.capabilities.hasCompoundSound);
```

hasDataLoading (propriété capabilities.hasDataLoading)

```
public static hasDataLoading : Boolean [read-only]
```

Renvoie `true` si le lecteur Flash Lite est capable charger de façon dynamique des données supplémentaires par l'intermédiaire d'appels à des fonctions spécifiques.

Vous pouvez appeler les fonctions suivantes :

- `loadMovie()`
- `loadMovieNum()`
- `loadVariables()`
- `loadVariablesNum()`
- `XML.parseXML()`
- `Sound.loadSound()`
- `MovieClip.loadVariables()`
- `MovieClip.loadMovie()`
- `MovieClipLoader.loadClip()`
- `LoadVars.load()`
- `LoadVars.sendAndLoad()`

Dans le cas contraire, cette propriété renvoie `false`.

Disponibilité
Flash Lite 2.0

Exemple
L'exemple suivant présente la valeur de cette propriété en lecture seule :

```
trace(System.capabilities.hasDataLoading);
```

hasEmail (propriété capabilities.hasEmail)

```
public static hasEmail : Boolean [read-only]
```

Renvoie `true` si le lecteur Flash Lite est capable d'envoyer des messages électroniques avec la commande `GetURL` d'ActionScript.

Dans le cas contraire, cette propriété renvoie `false`.

Disponibilité
Flash Lite 2.0

Exemple
L'exemple suivant présente la valeur de cette propriété en lecture seule :

```
trace(System.capabilities.hasEmail);
```

hasEmbeddedVideo (propriété capabilities.hasEmbeddedVideo)

```
public static hasEmbeddedVideo : Boolean [read-only]
```

Une valeur booléenne qui indique si le périphérique portable prend en charge la vidéo intégrée.

Remarque : La propriété `hasEmbeddedVideo` est toujours définie sur `true` dans Flash Lite 2.0 et Flash Lite 2.1 afin d'indiquer si la bibliothèque est prise en charge pour la vidéo de périphérique.

Disponibilité
Flash Lite 2.0

Exemple
L'exemple suivant présente la valeur de cette propriété en lecture seule :

```
trace(System.capabilities.hasEmbeddedVideo);
```

hasMappableSoftKeys (propriété capabilities.hasMappableSoftKeys)

```
public static hasMappableSoftKeys : Boolean
```

Renvoie `true` si le périphérique portable permet de réinitialiser ou affecter de nouveau des étiquettes de touches programmables et de gérer les événements provenant de ces touches programmables. Sinon, `false`.

Disponibilité
Flash Lite 2.0

Exemple
L'exemple suivant présente la valeur de cette propriété en lecture seule :

```
trace(System.capabilities.hasMappableSoftKeys);
```

hasMFI (propriété capabilities.hasMFI)

```
public static hasMFI : Boolean [read-only]
```

Renvoie `true` si le périphérique portable est capable de lire les données audio au format MFI.

Dans le cas contraire, cette propriété renvoie `false`.

Disponibilité
Flash Lite 2.0

Exemple
L'exemple suivant présente la valeur de cette propriété en lecture seule :

```
trace(System.capabilities.hasMFI);
```

hasMIDI (propriété capabilities.hasMIDI)

```
public static hasMIDI : Boolean [read-only]
```

Renvoie `true` si le périphérique portable est capable de lire les données audio au format MIDI.

Dans le cas contraire, cette propriété renvoie `false`.

Disponibilité
Flash Lite 2.0

Exemple
L'exemple suivant présente la valeur de cette propriété en lecture seule :

```
trace(System.capabilities.hasMIDI);
```

hasMMS (propriété capabilities.hasMMS)

```
public static hasMMS : Boolean [read-only]
```

Renvoie `true` si le périphérique portable est capable d'envoyer des messages MMS avec la commande `GetURL` d'ActionScript.

Dans le cas contraire, cette propriété renvoie `false`.

Disponibilité
Flash Lite 2.0

Exemple
L'exemple suivant présente la valeur de cette propriété en lecture seule :

```
trace(System.capabilities.hasMMS);
```

hasMouse (propriété capabilities.hasMouse)

```
public static hasMouse : Boolean [read-only]
```

Indique si le périphérique portable envoie les événements liés à la souris au lecteur Flash Lite.

Cette propriété renvoie `true` si le périphérique portable envoie les événements liés à la souris au lecteur Flash Lite. Sinon, elle renvoie la valeur `false`.

Disponibilité
Flash Lite 2.0

Exemple
L'exemple suivant présente la valeur de cette propriété en lecture seule :

```
trace(System.capabilities.hasMouse);
```

hasMP3 (propriété capabilities.hasMP3)

```
public static hasMP3 : Boolean [read-only]
```

Indique si le périphérique portable 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
Flash Lite 2.0

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 [read-only]
```

Valeur booléenne définie sur `true` si le lecteur s'exécute sur un périphérique portable qui prend en charge l'impression ; définie sur `false` sinon. La chaîne de serveur est `PR`.

Remarque : Pour Flash Lite 2.0, la valeur renvoyée est toujours `false`.

Disponibilité
Flash Lite 2.0

Exemple
L'exemple suivant présente la valeur de cette propriété en lecture seule :

```
trace(System.capabilities.hasPrinting);
```

hasQWERTYKeyboard (propriété capabilities.hasQWERTYKeyboard)

```
public static hasQWERTYKeyboard : Boolean [read-only]
```

Renvoie `true` si le lecteur Flash Lite peut traiter le code ActionScript associé à l'ensemble des touches, ce qui inclut la touche RETOUR ARRIERE, figurant sur les claviers standard.

Dans le cas contraire, cette propriété renvoie `false`.

Disponibilité
Flash Lite 2.0

Exemple
L'exemple suivant présente la valeur de cette propriété en lecture seule :

```
trace(System.capabilities.hasQWERTYKeyboard);
```

hasScreenBroadcast (propriété capabilities.hasScreenBroadcast)

```
public static hasScreenBroadcast : Boolean [read-only]
```

Valeur booléenne définie sur `true` si le lecteur prend en charge le développement des applications de diffusion sur écran devant être exécutées via Flash Media Server ; définie sur `false` sinon. La chaîne de serveur est `SB`.

Remarque : Pour Flash Lite 2.0, la valeur renvoyée est toujours `false`.

Disponibilité
Flash Lite 2.0

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 [read-only]
```

Valeur booléenne définie sur `true` si le lecteur prend en charge la lecture des applications de diffusion sur écran exécutées via Flash Media Server ; définie sur `false` sinon. La chaîne de serveur est `SP`.

Remarque : Pour Flash Lite 2.0, la valeur renvoyée est toujours `false`.

Disponibilité
Flash Lite 2.0

Exemple
L'exemple suivant présente la valeur de cette propriété en lecture seule :

```
trace(System.capabilities.hasScreenPlayback);
```

hasSharedObjects (propriété capabilities.hasSharedObjects)

```
public static hasSharedObjects : Boolean [read-only]
```

Renvoie `true` si le contenu Flash Lite lu par une application peut accéder à la version Flash Lite des objets partagés.

Dans le cas contraire, cette propriété renvoie `false`.

Disponibilité
Flash Lite 2.0

Exemple
L'exemple suivant présente la valeur de cette propriété en lecture seule :

```
trace(System.capabilities.hasSharedObjects);
```

hasSMAF (propriété capabilities.hasSMAF)

```
public static hasSMAF : Boolean [read-only]
```

Renvoie `true` si le périphérique portable est capable de lire les données audio au format SMAF.

Dans le cas contraire, cette propriété renvoie `false`.

Disponibilité
Flash Lite 2.0

Exemple
L'exemple suivant présente la valeur de cette propriété en lecture seule :

```
trace(System.capabilities.hasSMAF);
```

hasSMS (propriété capabilities.hasSMS)

```
public static hasSMS : Number [read-only]
```

Indique si le périphérique portable est capable d'envoyer des messages SMS avec la commande `GetURL` d'ActionScript.

Si Flash Lite peut envoyer des messages SMS, cette variable est définie et prend la valeur 1. Sinon, cette variable n'est pas définie.

Disponibilité
Flash Lite 2.0

Exemple
L'exemple suivant présente la valeur de cette propriété en lecture seule :

```
trace(System.capabilities.hasSMS);
```

hasStreamingAudio (propriété capabilities.hasStreamingAudio)

```
public static hasStreamingAudio : Boolean [read-only]
```

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
Flash Lite 2.0

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 [read-only]
```

Valeur booléenne qui indique si le lecteur peut lire les vidéos en flux continu.

Remarque : La propriété `hasStreamingVideo` est toujours définie sur `false` dans Flash Lite 2.0 et Flash Lite 2.1 afin d'indiquer que la diffusion en continu des FLV n'est pas prise en charge.

Disponibilité
Flash Lite 2.0

Exemple
L'exemple suivant présente la valeur de cette propriété en lecture seule :

```
trace(System.capabilities.hasStreamingVideo);
```

hasStylus (propriété capabilities.hasStylus)

```
public static hasStylus : Boolean [read-only]
```

Indique si le périphérique portable prend en charge les événements relatifs au stylet.

Cette propriété renvoie `true` si la plate-forme du périphérique portable ne prend pas en charge les événements relatifs au stylet. Dans le cas contraire, cette propriété renvoie `false`.

Le stylet ne prend pas en charge l'événement onMouseMove. Cet indicateur de fonctionnalité permet au contenu Flash de vérifier si la plate-forme du périphérique portable prend en charge cet événement.

Disponibilité
Flash Lite 2.0

Exemple
L'exemple suivant présente la valeur de cette propriété en lecture seule :

```
trace(System.capabilities.hasStylus);
```

hasVideoEncoder (propriété capabilities.hasVideoEncoder)

```
public static hasVideoEncoder : Boolean [read-only]
```

Spécifie si Flash Lite 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 ; sinon définie sur `false`. La chaîne de serveur est `VE`.

Remarque : Pour Flash Lite 2.0, la valeur renvoyée est toujours `false`.

Disponibilité
Flash Lite 2.0

Exemple
L'exemple suivant présente la valeur de cette propriété en lecture seule :

```
trace(System.capabilities.hasVideoEncoder);
```

hasXMLSocket (propriété capabilities.hasXMLSocket)

```
public static hasXMLSocket : Number [read-only]
```

Indique si l'application hôte prend en charge les sockets XML.

Si l'application hôte prend en charge les sockets XML, cette variable est définie et a la valeur 1. Sinon, cette variable n'est pas définie.

imageMIMETypes (propriété capabilities.imageMIMETypes)

```
public static imageMIMETypes : Array [read-only]
```

Renvoie un tableau qui contient tous les types MIME que la fonction `loadMovie` et les codecs du périphérique portable prennent en charge pour le traitement des images.

Disponibilité
Flash Lite 2.0

Exemple
L'exemple suivant présente la valeur de cette propriété en lecture seule :

```
trace(System.capabilities.imageMIMETypes);
```

isDebugger (propriété capabilities.isDebugger)

```
public static isDebugger : Boolean [read-only]
```

Valeur booléenne indiquant si le lecteur est une version officielle (`false`) ou une version de débogage spéciale (`true`). La chaîne de serveur est `DEB`.

Remarque : Pour Flash Lite 2.0, la valeur renvoyée est toujours `false`.

Disponibilité
Flash Lite 2.0

Exemple
L'exemple suivant présente la valeur de cette propriété en lecture seule :

```
trace(System.capabilities.isDebugger);


***note about space instead of tab used for indents in code
```

language (propriété capabilities.language)

```
public static language : String [read-only]
```

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.

Langue	Balise
Tchèque	cs
Danois	da
Néerlandais	nl
Anglais	en
Finnois	fi
Français	fr
Allemand	de
Hongrois	hu
Italien	it
Japonais	ja
Coréen	ko

Langue	Balise
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
Flash Lite 2.0

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 [read-only]
```

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 Lite Player ne peut pas lire de fichiers (y compris le premier fichier SWF de démarrage de Flash Lite 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`.

Remarque : Pour Flash Lite 2.0, la valeur renvoyée est toujours `true`.

Disponibilité
Flash Lite 2.0

Exemple
L'exemple suivant présente la valeur de cette propriété en lecture seule :

```
trace(System.capabilities.localFileReadDisable);
```

MIMETypes (propriété capabilities.MIMETypes)

```
public static MIMETypes : Array [read-only]
```

Renvoie un tableau qui contient tous les types MIME que la fonction `loadMovie` et les objets Sound et Video prennent en charge.

Disponibilité
Flash Lite 2.0

Exemple
L'exemple suivant présente la valeur de cette propriété en lecture seule :

```
trace(System.capabilities.MIMETypes);
```

os (propriété capabilities.os)

```
public static os : String [read-only]
```

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 serveur est `OS`.

Disponibilité
Flash Lite 2.0

Exemple
L'exemple suivant présente la valeur de cette propriété en lecture seule :

```
trace(System.capabilities.os);
```

screenOrientation (propriété capabilities.screenOrientation)

```
public static screenOrientation : String [read-only]
```

Cette variable appartient à l'objet `System.capabilities` qui indique l'orientation actuelle de l'écran.

Valeurs possibles pour la propriété `screenOrientation` :

- `normal`, l'orientation de l'écran est normale
- `rotated90`, l'écran a pivoté sur 90 degrés
- `rotated180`, l'écran a pivoté sur 180 degrés
- `rotated270`, l'écran a pivoté sur 270 degrés

Disponibilité
Flash Lite 2.0

Exemple
L'exemple suivant présente la valeur de cette propriété en lecture seule :

```
trace(System.capabilities.screenOrientation);
```

screenResolutionX (propriété capabilities.screenResolutionX)

```
public static screenResolutionX : Number [read-only]
```

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
Flash Lite 2.0

Exemple
L'exemple suivant présente la valeur de cette propriété en lecture seule :

```
trace(System.capabilities.screenResolutionX);
```

screenResolutionY (propriété capabilities.screenResolutionY)

```
public static screenResolutionY : Number [read-only]
```

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
Flash Lite 2.0

Exemple
L'exemple suivant présente la valeur de cette propriété en lecture seule :

```
trace(System.capabilities.screenResolutionY);
```

softKeyCount (propriété capabilities.softKeyCount)

```
public static softKeyCount : Number [read-only]
```

Indique le nombre de touches programmables pouvant être reconfigurées que le périphérique portable prend en charge.

Disponibilité
Flash Lite 2.0

Exemple
L'exemple suivant présente la valeur de cette propriété en lecture seule :

```
    trace(System.capabilities.softKeyCount);
```

version (propriété capabilities.version)

```
public static version : String [read-only]
```

Chaîne contenant la plate-forme Flash Lite Player et les informations sur la version (par exemple, `"WIN 7,1,0,0"`). La chaîne de serveur est V.

Disponibilité
Flash Lite 2.0

Exemple
L'exemple suivant présente la valeur de cette propriété en lecture seule :

```
trace(System.capabilities.version);
```

videoMIMETypes (propriété capabilities.videoMIMETypes)

```
public static videoMIMETypes : Array [read-only]
```

Indique tous les types MIME pour la vidéo que les codecs du périphérique portable prennent en charge.

Cette propriété renvoie un tableau de tous les types MIME pour la vidéo que les codecs du périphérique portable prennent en charge.

Disponibilité
Flash Lite 2.0

Exemple
L'exemple suivant présente la valeur de cette propriété en lecture seule :

```
trace(System.capabilities.videoMIMETypes);
```

Color

```
Object
    |
    +-Color
```

```
public class Color
extends Object
```

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
Flash Lite 2.0

Résumé des propriétés

Propriétés héritées de la classe Object

constructor (propriété Object.constructor), __proto__ (Object.__proto__, propriété) prototype (propriété Object.prototype), __resolve (Object.__resolve, propriété)

Récapitulatif des constructeurs

Signature	Description
Color (target : Object)	Crée un objet Color pour le clip spécifié par le paramètre target_mc.

Résumé de la méthode

Modificateurs	Signature	Description
	getRGB () : Number	Renvoie la combinaison R+V+B actuellement utilisée par l'objet Color.
	getTransform () : Object	Renvoie la valeur de transformation définie par le dernier appel Color.setTransform().
	setRGB (offset : Number) : Void	Spécifie une couleur RVB pour un objet Color.
	setTransform (transformObject : Object) : Void	Définit les informations relatives à la transformation de couleurs pour un objet Color.

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 Color

```
public Color(target:Object)
```

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

Flash Lite 2.0

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
```

Renvoie la combinaison R+V+B actuellement utilisée par l'objet Color.

Disponibilité
Flash Lite 2.0

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 aussi
setRGB (méthode Color.setRGB)

getTransform (méthode Color.getTransform)

```
public getTransform() : Object
```

Renvoie la valeur de transformation définie par le dernier appel `Color.setTransform()`.

Disponibilité
Flash Lite 2.0

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 aussi

setTransform (méthode Color.setTransform)

setRGB (méthode Color.setRGB)

```
public setRGB(offset:Number) : Void
```

Spécifie une couleur RVB pour un objet Color. L'appel de cette méthode remplace tout paramètre `Color.setTransform()` précédent.

Disponibilité

Flash Lite 2.0

Paramètres

offset : Number - `0xRRVVBB` Valeur hexadécimale ou 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 un clip 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);
my_color.setRGB(0xFF0000); // my_mc turns red
```

Voir aussi

setTransform (méthode Color.setTransform)

setTransform (méthode Color.setTransform)

```
public setTransform(transformObject:Object) : Void
```

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

Flash Lite 2.0

Paramètres

transformObject : Object - Objet créé avec le 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 *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 aussi

Object

ColorTransform (flash.geom.ColorTransform)

```
Object
    |
    +-flash.geom.ColorTransform

public class ColorTransform
extends Object
```

La classe ColorTransform permet de régler de façon mathématique l'ensemble des valeurs de couleur dans un clip. La fonction de réglage des couleurs ou de *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`

Toute valeur de canal de couleur supérieure à 255 après le calcul est ramenée à 255. Si elle est inférieure à 0, elle est réglée sur 0.

Vous devez utiliser le constructeur `new ColorTransform()` pour créer un objet ColorTransform avant 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

Flash Lite 3.1

Voir aussi

`colorTransform (propriété Transform.colorTransform)`

Résumé des propriétés

Modificateurs	Propriété	Description
	alphaMultiplier : Number	Une valeur décimale multipliée par la valeur du canal de transparence alpha.
	alphaOffset : Number	Nombre, compris entre -255 et 255, qui est ajouté à la valeur du canal de transparence alpha après sa multiplication par la valeur `alphaMultiplier`.
	blueMultiplier : Number	Une valeur décimale multipliée par la valeur du canal de bleu.
	blueOffset : Number	Nombre, compris entre -255 et 255, qui est ajouté à la valeur du canal de bleu après sa multiplication par la valeur `blueMultiplier`.
	greenMultiplier : Number	Une valeur décimale multipliée par la valeur du canal de vert.
	greenOffset : Number	Nombre, compris entre -255 et 255, qui est ajouté à la valeur du canal de vert après sa multiplication par la valeur `greenMultiplier`.
	redMultiplier : Number	Une valeur décimale multipliée par la valeur du canal de rouge.
	redOffset : Number	Nombre, compris entre -255 et 255, qui est ajouté à la valeur du canal de rouge après sa multiplication par la valeur `redMultiplier`.
	rgb : Number	Valeur de couleur RVB d'un objet ColorTransform.

« constructor (propriété Object.constructor) » à la page 507, __proto__ (Object.__proto__, propriété), prototype (propriété Object.prototype), « __resolve (Object.__resolve, propriété) » à la page 511

Récapitulatif 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.

« addProperty (méthode Object.addProperty) » à la page 504, « hasOwnProperty (méthode Object.hasOwnProperty) » à la page 507, « isPropertyEnumerable (méthode Object.isPropertyEnumerable) » à la page 507, « isPrototypeOf (méthode Object.isPrototypeOf) » à la page 508, « registerClass (méthode Object.registerClass) » à la page 510, « toString (méthode Object.toString) » à la page 514 « unwatch (méthode Object.unwatch) » à la page 515, valueOf (méthode Object.valueOf), « watch (méthode Object.watch) » à la page 517

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 en utilisant la propriété MovieClip._alpha, cela affecte la valeur de la propriété alphaMultiplier de l'objet ColorTransform de ce clip.

Disponibilité

Flash Lite 3.1

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 aussi

_alpha (MovieClip._alpha, propriété)

alphaOffset (propriété ColorTransform.alphaOffset)

public alphaOffset : Number

Nombre, compris entre -255 et 255, qui est ajouté à la valeur du canal de transparence alpha après sa multiplication par la valeur `alphaMultiplier`.

Disponibilité

Flash Lite 3.1

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

Flash Lite 3.1

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
```

Nombre, compris entre -255 et 255, qui est ajouté à la valeur du canal de bleu après sa multiplication par la valeur `blueMultiplier`.

Disponibilité

Flash Lite 3.1

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

Flash Lite 3.1

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 la 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

Flash Lite 3.1

Paramètres

second : ColorTransform - Second 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

Flash Lite 3.1

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), this;
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
```

Nombre, compris entre -255 et 255, qui est ajouté à la valeur du canal de vert après sa multiplication par la valeur `greenMultiplier`.

Disponibilité

Flash Lite 2.0

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

Flash Lite 2.0

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
```

Nombre, compris entre -255 et 255, qui est ajouté à la valeur du canal de rouge après sa multiplication par la valeur `redMultiplier`.

Disponibilité

Flash Lite 2.0

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
```

Valeur de couleur RVB d'un objet ColorTransform.

La définition de cette propriété entraîne la modification des trois valeurs de décalage de couleur (`redOffset`, `greenOffset` et `blueOffset`) et le réglage sur zéro des trois valeurs de multiplicateur de couleur (`redMultiplier`, `greenMultiplier` et `blueMultiplier`). Les valeurs de multiplicateur et de décalage de la transparence alpha ne changent pas.

Transmettez une valeur à cette propriété au format : 0x*RRVVBB*. 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
Flash Lite 2.0

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;
}
```

toString (méthode ColorTransform.toString)

```
public toString() : String
```

Formate et renvoie une chaîne qui décrit l'ensemble des propriétés de l'objet ColorTransform.

Disponibilité

Flash Lite 2.0

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 crée 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)
```

Date

```
Object
    |
    +-Date
```

```
public class Date
extends Object
```

La classe Date 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 sous lequel Flash Lite 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 à 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 avec 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

Flash Lite 2.0

Résumé des propriétés

Propriétés héritées de la classe Object

constructor (propriété Object.constructor), __proto__ (Object.__proto__, propriété) prototype (propriété Object.prototype), __resolve (Object.__resolve, propriété)

Récapitulatif 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.
	getLocaleLongDate () : String	Renvoie une chaîne représentant la date actuelle, sous forme longue, formatée selon les paramètres régionaux en vigueur.
	getLocaleShortDate () : String	Renvoie une chaîne représentant la date du jour, sous forme courte et formatée selon les paramètres régionaux en vigueur.
	getLocaleTime () : String	Renvoie une chaîne représentant l'heure actuelle et formatée selon les paramètres régionaux en vigueur.
	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.

Modificateurs	Signature	Description
	getUTCMilliseconds() : Number	Renvoie les millisecondes (entier de 0 à 999) de l'objet Date spécifié, conformément à l'heure universelle.
	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.
	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.

Modificateurs	Signature	Description
	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.
statique	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 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()`.

Le code suivant 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

Flash Lite 2.0

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 [facultatif] - 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 aussi

getMonth (méthode Date.getMonth), 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 sous lequel Flash Lite Player s'exécute.

Disponibilité

Flash Lite 2.0

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 aussi

getMonth (méthode Date.getMonth), 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 sous lequel Flash Lite Player s'exécute.

Disponibilité

Flash Lite 2.0

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 sous lequel Flash Lite Player s'exécute.

Disponibilité
Flash Lite 2.0

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 sous lequel Flash Lite Player s'exécute.

Disponibilité
Flash Lite 2.0

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;
}
```

getLocaleLongDate (méthode Date.getLocaleLongDate)

```
public getLocaleLongDate() : String
```

Renvoie une chaîne représentant la date actuelle, sous forme longue, formatée selon les paramètres régionaux en vigueur.

Remarque : Le format de la date dépend du périphérique portable et des paramètres régionaux.

Disponibilité
Flash Lite 2.0

Valeur renvoyée
String - Chaîne représentant la date du jour, sous forme longue, formatée selon les paramètres régionaux en vigueur.

Exemple
L'exemple suivant utilise le constructeur pour créer un objet Date en fonction de l'heure actuelle. Il applique également la méthode `getLocaleLongDate()` pour renvoyer la date du jour, sous forme longue et formatée en fonction des paramètres régionaux sélectionnés, comme suit :

```
var my_date:Date = new Date();
trace(my_date.getLocaleLongDate());
```

Vous trouverez ci-dessous des exemples de valeurs renvoyées par `getLocaleLongDate()` :

```
October 16, 2005
16 October 2005
```

getLocaleShortDate (méthode Date.getLocaleShortDate)

```
public getLocaleShortDate() : String
```

Renvoie une chaîne représentant la date du jour, sous forme courte et formatée selon les paramètres régionaux en vigueur.

Remarque : Le format de la date dépend du périphérique portable et des paramètres régionaux.

Disponibilité
Flash Lite 2.0

Valeur renvoyée
String - Chaîne représentant la date du jour, sous forme courte, formatée selon les paramètres régionaux en vigueur.

Exemple
L'exemple suivant utilise le constructeur pour créer un objet Date en fonction de l'heure actuelle. Il applique également la méthode `getLocaleShortDate()` pour renvoyer la date du jour, sous forme courte et formatée en fonction des paramètres régionaux sélectionnés, comme suit :

```
var my_date:Date = new Date();
trace(my_date.getLocaleShortDate());
```

Vous trouverez ci-dessous des exemples de valeurs renvoyées par `getLocaleLongDate()` :

```
10/16/2005
16-10-2005
```

getLocaleTime (méthode Date.getLocaleTime)

```
public getLocaleTime() : String
```

Renvoie une chaîne représentant l'heure actuelle et formatée selon les paramètres régionaux en vigueur.

Remarque : Le format de la date dépend du périphérique portable et des paramètres régionaux.

Disponibilité
Flash Lite 2.0

Valeur renvoyée
String - Chaîne représentant l'heure actuelle et formatée selon les paramètres régionaux en vigueur.

Exemple
L'exemple suivant utilise le constructeur pour créer un objet Date en fonction de l'heure actuelle. Il utilise également la méthode `getLocaleTime()` pour renvoyer l'heure en fonction des paramètres régionaux actifs, comme suit :

```
var my_date:Date = new Date();
trace(my_date.getLocaleTime());
```

Vous trouverez ci-dessous des exemples de valeurs renvoyées par `getLocaleTime()` :

```
6:10:44 PM
18:10:44
```

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 sous lequel Flash Lite Player s'exécute.

Disponibilité
Flash Lite 2.0

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 sous lequel Flash Lite Player s'exécute.

Disponibilité
Flash Lite 2.0

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 en minutes de cet objet :

```
var my_date:Date = new Date();
trace(my_date.getMinutes());
```

getMonth (méthode Date.getMonth)

```
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 sous lequel Flash Lite Player s'exécute.

Disponibilité
Flash Lite 2.0

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 sous lequel Flash Lite Player s'exécute.

Disponibilité
Flash Lite 2.0

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 secondes 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
Flash Lite 2.0

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
Flash Lite 2.0

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
Flash Lite 2.0

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 aussi
`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
Flash Lite 2.0

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 aussi
`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
Flash Lite 2.0

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
```

Voir aussi
`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
Flash Lite 2.0

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 aussi

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

Flash Lite 2.0

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

Flash Lite 2.0

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
Flash Lite 2.0

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 aussi
getMonth (méthode Date.getMonth)

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
Flash Lite 2.0

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 par 100.

Disponibilité
Flash Lite 2.0

Valeur renvoyée
Number -

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 sous lequel Flash Lite Player s'exécute. L'année est l'année entière moins 1900. Par exemple, l'année 2000 est représentée par 100.

Disponibilité
Flash Lite 2.0

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 aussi
`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 sous lequel Flash Lite Player s'exécute.

Disponibilité
Flash Lite 2.0

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 25.05.04 :

```
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 réglés sur l'heure locale. L'heure locale est déterminée par le système d'exploitation sous lequel Flash Lite 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

Flash Lite 2.0

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 correspond pas à l'année 1999, mais à l'année 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 15.05.02 :

```
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 aussi

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 sous lequel Flash Lite Player s'exécute.

Disponibilité
Flash Lite 2.0

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 sous lequel Flash Lite Player s'exécute.

Disponibilité
Flash Lite 2.0

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 sous lequel Flash Lite Player s'exécute.

Disponibilité

Flash Lite 2.0

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 sous lequel Flash Lite Player s'exécute.

Disponibilité

Flash Lite 2.0

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 sous lequel Flash Lite Player s'exécute.

Disponibilité
Flash Lite 2.0

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 08:00:00, puis utilise la méthode `Date.setSeconds()` pour modifier l'heure et la définir sur 08:00:45 :

```
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 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.

Disponibilité
Flash Lite 2.0

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

Flash Lite 2.0

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 aussi

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

Flash Lite 2.0

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 aussi

`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

Flash Lite 2.0

Paramètres

hour : Number - Nombre ; entier compris entre 0 (minuit) et 23 (23h00).

minute : Number [facultatif] – Numéro ; 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] – Numéro ; 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] – Numéro ; 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

Flash Lite 2.0

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

Flash Lite 2.0

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

Flash Lite 2.0

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 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 aussi

`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
Flash Lite 2.0

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 08:00:00, 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 sous lequel Flash Lite Player s'exécute.

Disponibilité
Flash Lite 2.0

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
Flash Lite 2.0

Valeur renvoyée
String - Chaîne.

Exemple
L'exemple suivant renvoie les informations dans l'objet Date `dateOfBirth_date` sous forme de chaîne. Le résultat obtenu des instructions trace est en heure locale et varie en conséquence. 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 par le biais du constructeur de l'objet Date, et non au moyen d'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
Flash Lite 2.0

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. Le résultat obtenu est en heure locale et varie en conséquence. 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
Flash Lite 2.0

Valeur renvoyée
Number - Nombre de millisecondes.

Error

```
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
Flash Lite 2.0

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

```
constructor (propriété Object.constructor), __proto__ (Object.__proto__,
propriété) prototype (propriété Object.prototype), __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 vous transmettez un paramètre *message*, sa valeur est affectée à la propriété `Error.message`.

Disponibilité

Flash Lite 2.0

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 aussi

throw, instruction, try..catch..dernière instruction

message (propriété Error.message)

```
public message : String
```

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

Flash Lite 2.0

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 Sortie car vous essayez de diviser par 0.

Voir aussi

throw, instruction, try..catch..dernière instruction

name (propriété Error.name)

```
public name : String
```

Contient le nom de l'objet Error. Par défaut, la valeur de cette propriété est « `Error` ».

Disponibilité
Flash Lite 2.0

Exemple
Dans l'exemple suivant, une fonction renvoie une erreur spécifiée en fonction des deux nombres que vous essayez de diviser. Ajoutez l'ActionScript suivant à l'Image 1 du scénario :

```
function divideNumber(numerator:Number, denominator:Number):Number {
    if (isNaN(numerator) || isNaN(denominator)) {
    throw new Error("divideNumber() 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());
}
```

Ajoutez 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 aussi
throw, instruction, try..catch..dernière instruction

toString (méthode Error.toString)

```
public toString() : String
```

Renvoie la chaîne "Error" ou la valeur contenue dans Error.message, s'il est défini.

Disponibilité
Flash Lite 2.0

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 aussi

message (propriété Error.message), throw, instruction, try..catch..dernière instruction

ExtendedKey

```
Object
    |
    +-ExtendedKey
```

```
public class ExtendedKey
extends Object
```

Fournit les codes de touche étendus qui peuvent être renvoyés par la méthode `Key.getCode()`.

Disponibilité

Flash Lite 2.0

Exemple

L'exemple suivant crée un écouteur qui est appelé lorsque l'utilisateur appuie sur une touche. Il applique la méthode `Key.getCode()` pour obtenir le code de la touche qui a été sollicitée :

```
var myListener = new Object();

myListener.onKeyDown = function() {
    var code = Key.getCode();
    trace(code + " down");
}

myListener.onKeyUp = function() {
    trace("onKeyUp called");
}

Key.addListener(myListener);
```

Voir aussi

getCode (méthode Key.getCode)

Résumé des propriétés

Modificateurs	Propriété	Description
statique	SOFT1 : String	La valeur du code de la touche programmable SOFT1.
statique	SOFT3 : String	La valeur du code de la touche programmable SOFT3.
statique	SOFT4 : String	La valeur du code de la touche programmable SOFT4.
statique	SOFT5 : String	La valeur du code de la touche programmable SOFT5.
statique	SOFT6 : String	La valeur du code de la touche programmable SOFT6.
statique	SOFT7 : String	La valeur du code de la touche programmable SOFT7.
statique	SOFT8 : String	La valeur du code de la touche programmable SOFT8.
statique	SOFT9 : String	La valeur du code de la touche programmable SOFT9.
statique	SOFT10 : String	La valeur du code de la touche programmable SOFT10.
statique	SOFT11 : String	La valeur du code de la touche programmable SOFT11.
statique	SOFT12 : String	La valeur du code de la touche programmable SOFT12.
statique	SOFT2 : String	La valeur du code de la touche programmable SOFT2.

Propriétés héritées de la classe Object

```
constructor (propriété Object.constructor),__proto__ (Object.__proto__,
propriété)prototype (propriété Object.prototype),__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)
```

SOFT1 (propriété ExtendedKey.SOFT1)

```
public static SOFT1 : String
```

La valeur du code de la touche programmable SOFT1. Le code de touche SOFT1 correspond toujours à la touche programmable de gauche ; SOFT2 correspond toujours à la touche programmable de droite.

Disponibilité

Flash Lite 2.0

Exemple

L'exemple suivant crée un écouteur qui traite les touches programmables gauche et droite :

```
var myListener:Object = new Object();
myListener.onKeyDown = function () {
var keyCode = Key.getCode();
switch (keyCode) {
    case ExtendedKey.SOFT1:
    // Handle left soft key.
    break;
    case ExtendedKey.SOFT2:
    // Handle right soft key
    break;
    }
}
Key.addListener(myListener);
```

SOFT2 (propriété ExtendedKey.SOFT2)

```
public static SOFT2 : String
```

La valeur du code de la touche programmable SOFT2. Le code de touche SOFT2 correspond toujours à la touche programmable de droite ; SOFT1 correspond toujours à la touche programmable de gauche.

Disponibilité
Flash Lite 2.0

Voir aussi
SOFT1 (propriété ExtendedKey.SOFT1)

SOFT3 (propriété ExtendedKey.SOFT3)

```
public static SOFT3 : String
```

La valeur du code de la touche programmable SOFT3.

Disponibilité
Flash Lite 2.0

SOFT4 (propriété ExtendedKey.SOFT4)

```
public static SOFT4 : String
```

La valeur du code de la touche programmable SOFT4.

Disponibilité
Flash Lite 2.0

SOFT5 (propriété ExtendedKey.SOFT5)

```
public static SOFT5 : String
```

La valeur du code de la touche programmable SOFT5.

Disponibilité
Flash Lite 2.0

SOFT6 (propriété ExtendedKey.SOFT6)

```
public static SOFT6 : String
```

La valeur du code de la touche programmable SOFT6.

Disponibilité
Flash Lite 2.0

SOFT7 (propriété ExtendedKey.SOFT7)

```
public static SOFT7 : String
```

La valeur du code de la touche programmable SOFT7.

Disponibilité
Flash Lite 2.0

SOFT8 (propriété ExtendedKey.SOFT8)

```
public static SOFT8 : String
```

La valeur du code de la touche programmable SOFT8.

Disponibilité
Flash Lite 2.0

SOFT9 (propriété ExtendedKey.SOFT9)

```
public static SOFT9 : String
```

La valeur du code de la touche programmable SOFT9.

SOFT10 (propriété ExtendedKey.SOFT10)

```
public static SOFT10 : String
```

La valeur du code de la touche programmable SOFT10.

Disponibilité
Flash Lite 2.0

SOFT11 (propriété ExtendedKey.SOFT11)

```
public static SOFT11 : String
```

La valeur du code de la touche programmable SOFT11.

Disponibilité
Flash Lite 2.0

SOFT12 (propriété ExtendedKey.SOFT12)

```
public static SOFT12 : String
```

La valeur du code de la touche programmable SOFT12.

Disponibilité
Flash Lite 2.0

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
Flash Lite 2.0

Résumé des propriétés
Propriétés héritées de la classe Object

constructor (propriété Object.constructor), __proto__ (Object.__proto__, propriété) prototype (propriété Object.prototype), __resolve (Object.__resolve, propriété)

Résumé de la méthode

Modificateurs	Signature	Description
	apply(thisObject:Object, [argArray:Array])	Spécifie la valeur thisObject à utiliser dans toute fonction appelée par ActionScript.
	call(thisObject:Object, [parameter1: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])
```

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
Flash Lite 2.0

Paramètres
thisObject : Object - Objet auquel myFunction s'applique.

argArray : Array [facultatif] - Tableau dont les éléments sont transmis à myFunction en tant que paramètres.

Valeur renvoyée
Toute valeur spécifiée par la fonction appelée.

Exemple
Les invocations de fonction suivantes sont équivalentes :

```
Math.atan2(1, 0)
Math.atan2.apply(null, [1, 0])
```

L'exemple 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 aussi

call (méthode Function.call)

call (méthode Function.call)

```
public call(thisObject:Object, [parameter1: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'appel 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 suit :

```
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
Flash Lite 2.0

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.

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 aussi
apply (méthode Function.apply)

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 des interfaces. 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. Utilisez les propriétés System.capabilities pour déterminer les touches prises en charge par un périphérique.

Certains périphériques et types de contenu Flash Lite ne prennent pas en charge toutes les touches. Par exemple, les périphériques qui prennent en charge la navigation bidirectionnelle ne prennent pas en charge les touches de navigation gauche et droite. En outre, seuls certains périphériques permettent d'accéder aux touches programmables. Pour plus d'informations, consultez la section *Développement d'applications Flash Lite 2.x et 3.x*.

Disponibilité
Flash Lite 2.0

Voir aussi

ExtendedKey

« has4WayKeyAS (propriété capabilities.has4WayKeyAS) » à la page 250

« hasMappableSoftKeys (propriété capabilities.hasMappableSoftKeys) » à la page 253

« hasQWERTYKeyboard (propriété capabilities.hasQWERTYKeyboard) » à la page 256

« softKeyCount (propriété capabilities.softKeyCount) » à la page 263

Résumé des propriétés

Modificateurs	Propriété	Description
statique	BACKSPACE : Number	La valeur de code correspondant à la touche Retour arrière (8).
statique	CAPSLOCK : Number	La valeur de code correspondant à la touche Verr Maj (20).
statique	CONTROL : Number	La valeur de code correspondant à la touche Ctrl (17).
statique	DELETEKEY : Number	La valeur de code correspondant à la touche Suppr (46).
statique	DOWN : Number	La valeur de code correspondant à la flèche Bas (40).
statique	END : Number	La valeur de code correspondant à la touche Fin (35).
statique	ENTER : Number	La valeur de code correspondant à la touche Entrée (13).
statique	ESCAPE : Number	La valeur de code correspondant à la touche Echap (27).
statique	HOME : Number	La valeur de code correspondant à la touche Origine (36).
statique	INSERT : Number	La valeur de code correspondant à la touche Inser (45).
statique	LEFT : Number	La valeur de code correspondant à la flèche Gauche (37).
statique	_listeners : Array [lecture seule]	Une liste de références à tous les objets écouteurs enregistrés auprès de l'objet Key.
statique	PGDN : Number	La valeur de code correspondant à la touche Pg. Suiv. (34).
statique	PGUP : Number	La valeur de code correspondant à la touche Pg. Préc. (33).
statique	RIGHT : Number	La valeur de code correspondant à la flèche Droite (39).
statique	SHIFT : Number	La valeur de code correspondant à la touche Maj (16).
statique	SPACE : Number	La valeur de code correspondant à la barre d'espace (32).
statique	TAB : Number	La valeur de code correspondant à la touche Tab (9).
statique	UP : Number	La valeur de code correspondant à la flèche Haut (38).

Propriétés héritées de la classe Object

constructor (propriété Object.constructor), __proto__ (Object.__proto__, propriété), prototype (propriété Object.prototype), __resolve (Object.__resolve, propriété)

Résumé des événements

Evénement	Description
onKeyDown = fonction() {}	Notifié lorsqu'une touche est enfoncée.
onKeyUp = fonction() {}	Notifié lorsqu'une touche est relâchée.

Résumé de la méthode

Modificateurs	Signature	Description
statique	addListener(listener: Object) : Void	Enregistre un objet pour qu'il reçoive la notification onKeyDown et onKeyUp.
statique	getAscii() : Number	Renvoie le code ASCII de la dernière touche enfoncée ou relâchée.
statique	getCode() : Number	Renvoie la valeur de code de touche de la dernière touche enfoncée.
statique	isDown(code:Number) : Boolean	Renvoie true si la touche spécifiée dans code est enfoncée ; false sinon.
statique	removeListener(listener:Object) : Boolean	Supprime un objet précédemment enregistré avec 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)

```
public static addListener(listener:Object) : Void
```

Enregistre un objet pour qu'il reçoive les notifications onKeyDown et onKeyUp. Lorsqu'une touche est enfoncée ou relâchée, quel que soit le focus d'entrée, tous les objets d'écoute enregistrés avec addListener() font l'objet d'un appel de leur méthode onKeyDown ou onKeyUp. Plusieurs objets peuvent écouter les notifications de clavier.

Disponibilité

Flash Lite 2.0

Paramètres

listener : Object - Objet avec les méthodes onKeyDown et onKeyUp.

Exemple

L'exemple suivant crée un nouvel objet écouteur et définit des fonctions pour onKeyDown et onKeyUp. La dernière ligne appelle 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);
```

Voir aussi

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)

```
public static BACKSPACE : Number
```

La valeur de code correspondant à la touche Retour arrière (8).

Disponibilité

Flash Lite 2.0

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);
```

CAPSLOCK (propriété Key.CAPSLOCK)

```
public static CAPSLOCK : Number
```

La valeur de code correspondant à la touche Verr Maj (20).

Disponibilité

Flash Lite 2.0

CONTROL (propriété Key.CONTROL)

```
public static CONTROL : Number
```

La valeur de code correspondant à la touche Ctrl (17).

Disponibilité
Flash Lite 2.0

DELETEKEY (propriété Key.DELETEKEY)

```
public static DELETEKEY : Number
```

La valeur de code correspondant à la touche Suppr (46).

Disponibilité
Flash Lite 2.0

DOWN (propriété Key.DOWN)

```
public static DOWN : Number
```

La valeur de code correspondant à la flèche Bas (40).

Disponibilité
Flash Lite 2.0

Exemple
L'exemple suivant déplace un clip intitulé `car_mc` selon une distance constante (10) lorsque vous appuyez sur les touches fléchées. Placez un clip sur la scène et nommez-le car_mc.

```
var DISTANCE:Number = 10;
var keyListener_obj:Object = new Object();
keyListener_obj.onKeyDown = function() {
    switch (Key.getCode()) {
    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)

```
public static END : Number
```

La valeur de code correspondant à la touche Fin (35).

Disponibilité
Flash Lite 2.0

ENTER (propriété Key.ENTER)

```
public static ENTER : Number
```

La valeur de code correspondant à la touche Entrée (13).

Disponibilité

Flash Lite 2.0

Exemple

L'exemple suivant déplace un clip lorsque vous appuyez sur les touches fléchées. La lecture du clip s'arrête lorsque vous appuyez sur la touche de sélection et supprimez l'événement `onEnterFrame`. Placez un clip sur la scène et nommez-le car_mc.

```
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);
```

ESCAPE (propriété Key.ESCAPE)

```
public static ESCAPE : Number
```

La valeur de code correspondant à la touche Echap (27).

Disponibilité

Flash Lite 2.0

Exemple

L'exemple suivant définit un compteur. Lorsque vous appuyez sur la touche de sélection, le panneau Sortie affiche le temps qu'il vous a fallu pour appuyer sur la touche.

```
var keyListener:Object = new Object();
keyListener.onKeyDown = function() {
    if (Key.isDown(Key.ENTER)) {
    // 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 Select key after: "+getTimer()+"ms ("+timer+"s)");
    }
};
Key.addListener(keyListener);
```

getAscii (méthode Key.getAscii)

```
public static 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 sur un clavier japonais ou anglais, `Key.getAscii()` renvoie `@`.

Disponibilité

Flash Lite 2.0

Valeur renvoyée

Number - La valeur ASCII de la dernière touche enfoncée. Cette méthode renvoie 0 si aucune touche n'a été enfoncée ou relâchée ou si la valeur ASCII 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);
```

L'exemple suivant ajoute un appel de la méthode `Key.getAscii()` pour illustrer la façon dont `getAscii()` et `getCode()` 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);
```

getCode (méthode Key.getCode)

```
public static getCode() : Number
```

Renvoie la valeur de code de touche de la dernière touche enfoncée.

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.

Disponibilité

Flash Lite 2.0

Valeur renvoyée

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 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);
```

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);
```

Disponibilité

Flash Lite 2.0

Voir aussi

getAscii (méthode Key.getAscii)

HOME (propriété Key.HOME)

```
public static HOME : Number
```

La valeur de code correspondant à la touche Origine (36).

Disponibilité
Flash Lite 2.0

INSERT (propriété Key.INSERT)

```
public static INSERT : Number
```

La valeur de code correspondant à la touche Inser (45).

Disponibilité
Flash Lite 2.0

isDown (méthode Key.isDown)

```
public static isDown(code:Number) : Boolean
```

Renvoie `true` si la touche spécifiée dans `code` est enfoncée ; `false` sinon.

Disponibilité
Flash Lite 2.0

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.

Valeur renvoyée
Boolean - La valeur `true` si la touche spécifiée dans `code` est enfoncée ; `false` sinon.

Exemple
Le script suivant permet d'utiliser les touches Gauche et Droite pour contrôler l'emplacement sur la scène d'un clip intitulé `car_mc` :

```
car_mc.onEnterFrame = function() {
    if (Key.isDown(Key.RIGHT)) {
    this._x += 10;
    } else if (Key.isDown(Key.LEFT)) {
    this._x -= 10;
    }
};
```

LEFT (propriété Key.LEFT)

```
public static LEFT : Number
```

La valeur de code correspondant à la flèche Gauche (37).

Disponibilité
Flash Lite 2.0

_listeners (propriété Key._listeners)

```
public static _listeners : Array [read-only]
```

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
Flash Lite 2.0

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 = function() {}
```

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.

Disponibilité
Flash Lite 2.0

Voir aussi
addListener (méthode Key.addListener)

onKeyUp (écouteur d'événement Key.onKeyUp)

```
onKeyUp = function() {}
```

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.

Disponibilité
Flash Lite 2.0

Voir aussi
addListener (méthode Key.addListener)

PGDN (propriété Key.PGDN)

```
public static PGDN : Number
```

La valeur de code correspondant à la touche Pg. Suiv. (34).

Disponibilité
Flash Lite 2.0

PGUP (propriété Key.PGUP)

```
public static PGUP : Number
```

La valeur de code correspondant à la touche Pg. Préc. (33).

Disponibilité
Flash Lite 2.0

removeListener (méthode Key.removeListener)

```
public static removeListener(listener:Object) : Boolean
```

Supprime un objet précédemment enregistré avec `Key.addListener()`.

Disponibilité
Flash Lite 2.0

Paramètres
listener : Object - Un objet.

Valeur renvoyée
Boolean - Si l'objet *listener* été supprimé, la méthode renvoie `true`. Si l'objet *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`.

RIGHT (propriété Key.RIGHT)

```
public static RIGHT : Number
```

La valeur de code correspondant à la flèche Droite (39).

Disponibilité
Flash Lite 2.0

Exemple
L'exemple suivant déplace sur la scène un clip intitulé `car_mc` lorsque vous appuyez sur les flèches.

```
var DISTANCE:Number = 10;
var keyListener_obj:Object = new Object();
keyListener_obj.onKeyDown = function() {
    switch (Key.getCode()) {
    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)

```
public static SHIFT : Number
```

La valeur de code correspondant à la touche Maj (16).

Disponibilité
Flash Lite 2.0

SPACE (propriété Key.SPACE)

```
public static SPACE : Number
```

La valeur de code correspondant à la barre d'espace (32).

Disponibilité
Flash Lite 2.0

TAB (propriété Key.TAB)

```
public static TAB : Number
```

La valeur de code correspondant à la touche Tab (9).

Disponibilité
Flash Lite 2.0

UP (propriété Key.UP)

```
public static UP : Number
```

La valeur de code correspondant à la flèche Haut (38).

Disponibilité
Flash Lite 2.0

Exemple
L'exemple suivant déplace sur la scène un clip intitulé `car_mc` selon une distance constante (10) lorsque vous appuyez sur les flèches.

```
var DISTANCE:Number = 10;
var keyListener_obj:Object = new Object();
keyListener_obj.onKeyDown = function() {
    switch (Key.getCode()) {
    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
```

La classe LoadVars constitue une alternative à la fonction loadVariables() pour le transfert des variables entre Flash Lite et un serveur Web via HTTP. Utilisez 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 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. Utilisez 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

Flash Lite 2.0

Voir aussi

`loadVariables, fonction`, `onLoad (gestionnaire LoadVars.onLoad)`, `hasXMLSocket (propriété capabilities.hasXMLSocket)`

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

`constructor (propriété Object.constructor), __proto__ (Object.__proto__, propriété) prototype (propriété Object.prototype), __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.
onLoad = function(success:Boolean) {}	Invoqué lorsqu'une opération LoadVars.load() ou LoadVars.sendAndLoad() s'est terminée.

Récapitulatif 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 des en-têtes de requête HTTP (tels que Content-Type ou SOAPAction) envoyés avec des 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 l'objet *LoadVars*.
	send(url:String, target:String, [method:String]) : Boolean	Envoie les variables de l'objet *LoadVars* vers l'URL spécifiée.
	sendAndLoad(url:String, target:Object, [method:String]) : Boolean	Publie les variables de l'objet *LoadVars* vers l'URL spécifiée.
	toString() : String	Renvoie une chaîne contenant toutes les variables énumérables dans l'objet *LoadVars* 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 des en-têtes de requête HTTP (tels que `Content-Type` ou `SOAPAction`) envoyés avec des 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

Flash Lite 2.0

Paramètres

header : Object - Chaîne ou tableau de chaînes 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 de requête HTTP appelé `SOAPAction` avec 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 aussi

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
Flash Lite 2.0

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 aussi
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
Flash Lite 2.0

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 aussi
onData (gestionnaire LoadVars.onData), parseXML (méthode XML.parseXML)

getBytesLoaded (méthode LoadVars.getBytesLoaded)

```
public getBytesLoaded() : Number
```

Renvoie le nombre d'octets téléchargés en appelant `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.

Remarque : *Vous ne pouvez pas utiliser cette méthode pour renvoyer des informations concernant un fichier local résidant sur votre disque dur.*

Disponibilité
Flash Lite 2.0

Valeur renvoyée
Number - Entier.

Voir aussi
`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).

Remarque : *Vous ne pouvez pas utiliser cette méthode pour renvoyer des informations concernant un fichier local résidant sur votre disque dur.*

Disponibilité
Flash Lite 2.0

Valeur renvoyée
Number - Entier.

Voir aussi
`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 un objet *LoadVars*. Toutes les propriétés contenues dans l'objet LoadVars ayant les mêmes noms que les variables téléchargées sont écrasées. Toutes les propriétés contenues dans l'objet LoadVars 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()`.

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 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.

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 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 inter-domaines* sur le serveur hébergeant le fichier SWF.

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()`.

Disponibilité
Flash Lite 2.0

Paramètres
url : String - L'URL permettant de télécharger les variables. Si le fichier SWF qui émet cet appel s'exécute sur un navigateur Web, l'`url` doit appartenir au même domaine que le fichier SWF. Pour plus de détails, consultez la section Description.

Valeur renvoyée
Boolean - Si aucun paramètre (null) n'est transmis, `false` , sinon `true`. Utilisez le gestionnaire d'événements `onLoad()` pour vérifier que les données ont bien été téléchargées.

Exemple
Le code suivant définit un gestionnaire `onLoad` indiquant à quel moment les données sont renvoyées à l'application à partir d'un fichier texte, puis charge les données de ce fichier texte et les envoie dans le panneau Sortie.

```
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");
```

Voir aussi
`load (méthode XML.load)`, `loaded (propriété LoadVars.loaded)`, `onLoad (gestionnaire LoadVars.onLoad)`

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 propriétér `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
Flash Lite 2.0

Exemple
L'exemple suivant charge un fichier texte et affiche les informations dans le panneau 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 aussi
load (méthode LoadVars.load), sendAndLoad (méthode LoadVars.sendAndLoad), load (méthode XML.load)

constructeur LoadVars

```
public LoadVars()
```

Crée un objet LoadVars. Appelle les méthodes de cet objet LoadVars pour envoyer et charger des données.

Disponibilité
Flash Lite 2.0

Exemple
L'exemple suivant crée un objet LoadVars intitulé `my_lv` :

```
var my_lv:LoadVars = new LoadVars();
```

onData (gestionnaire LoadVars.onData)

```
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. 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 Lite. 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
Flash Lite 2.0

Paramètres
src : String - Chaîne ou `undefined` ; 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 TextField intitulée `content_txt` lorsque l'opération se termine. Si une erreur se produit, les informations s'affichent dans le panneau Sortie.

```
var my_lv:LoadVars = new LoadVars();
my_lv.onData = function(src:String) {
    if (src == undefined) {
        trace("Error loading content.");
        return;
    }
    content_txt.text = src;
};
my_lv.load("http://www.helpexamples.com/flash/params.txt", my_lv, "GET");
```

Voir aussi

`onLoad (gestionnaire LoadVars.onLoad)`, `load (méthode LoadVars.load)`, `sendAndLoad (méthode LoadVars.sendAndLoad)`

onLoad (gestionnaire LoadVars.onLoad)

```
onLoad = function(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

Flash Lite 2.0

Paramètres

success : Boolean - Indique si l'opération de chargement s'est terminée avec succès (`true`) ou a échoué (`false`).

Exemple

Consultez l'exemple de méthode LoadVars.sendAndLoad().

Voir aussi

`onLoad (gestionnaire XML.onLoad)`, `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 *LoadVars* vers l'URL spécifiée. Les variables sont concaténées dans une chaîne au format *application/x-www-form-urlencoded* ou dans la valeur de LoadVars.contentType. La méthode `POST` est utilisée sauf si `GET` est spécifié.

Vous devez spécifier le paramètre `target` pour exécuter le script ou l'application à l'URL spécifiée. Si vous omettez le paramètre `target`, la fonction renvoie `true`, mais le script ou l'application n'est 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, utilisez la méthode `LoadVars.sendAndLoad()`

Cette méthode est similaire à `XML.send()`.

Disponibilité
Flash Lite 2.0

Paramètres
url : String - URL vers laquelle les variables doivent être transférées.

target : String - Fenêtre de navigateur ou image dans laquelle la réponse s'affiche. Vous pouvez entrer le nom d'une fenêtre spécifique ou le sélectionner à partir des noms cibles réservés suivants :

- `"_self"` indique le cadre qui est actif dans la fenêtre ouverte.
- `"_blank"` indique une nouvelle fenêtre.
- `"_parent"` indique le parent du cadre actif.
- `"_top"` désigne le cadre de plus haut niveau dans la fenêtre ouverte.

method : String (facultatif) Méthode `GET` ou `POST` du protocole HTTP. La valeur par défaut est`POST`.

Valeur renvoyée
Boolean - Si aucun paramètre n'est spécifié, false, sinon true.

Exemple
L'exemple suivant copie deux valeurs à partir de champs 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 aussi
`sendAndLoad (méthode LoadVars.sendAndLoad)`, `send (méthode XML.send)`

sendAndLoad (méthode LoadVars.sendAndLoad)

```
public sendAndLoad(url:String, target:Object, [method:String]) : Boolean
```

Publie les variables de l'objet *LoadVars* vers l'URL spécifiée. La réponse du serveur est téléchargée et analysée, puis 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 `target` de la même manière que `LoadVars.load()`.

Pour les fichiers SWF lus par une version antérieure à Flash Player 7 (Flash Lite 1.x par exemple), 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 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.

Dans les fichiers SWF d'une version exécutée dans Flash Player 7 ou versions ultérieures (Flash Lite 2.x et 3.x par exemple), l'`url` doit se trouver exactement dans le même domaine. 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 inter-domaines* sur le serveur hébergeant le fichier SWF.

Cette méthode est similaire à `XML.sendAndLoad()`.

Disponibilité
Flash Lite 2.0

Paramètres
url : String - 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) Méthode `GET` ou `POST` du protocole HTTP. La valeur par défaut est`POST`.

Valeur renvoyée
Boolean

Exemple
Pour l'exemple suivant, ajoutez sur la scène un champ texte de saisie intitulé `name_txt`, un champ texte dynamique intitulé `result_txt` et un bouton intitulé `submit_btn`. Lorsque l'utilisateur clique sur le bouton, deux objets LoadVars sont créés : `send_lv` et `result_lv`. L'objet send_lv copie le nom de l'occurrence `name_txt` 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 le champ result_txt. Ajoutez l'ActionScript suivant à l'Image 1 du scénario :

```
var send_lv:LoadVars = new LoadVars();
var result_lv:LoadVars = new LoadVars();
result_lv.onLoad = function(success:Boolean) {
    if (success) {
        result_txt.text = result_lv.welcomeMessage;
    } else {
        result_txt.text = "Error connecting to server.";
    }
};

submit_btn.onRelease = function(){
    send_lv.name = name_txt.text;
    send_lv.sendAndLoad("http://www.flash-mx.com/mm/greeting.cfm", result_lv);
}
```

Voir aussi

send (méthode LoadVars.send), load (méthode LoadVars.load), sendAndLoad (méthode XML.sendAndLoad)

toString (méthode LoadVars.toString)

```
public toString() : String
```

Renvoie une chaîne contenant toutes les variables énumérables dans l'objet LoadVars au format de codage du contenu MIME *application/x-www-form-urlencoded*.

Disponibilité

Flash Lite 2.0

Valeur renvoyée

String

Exemple

L'exemple suivant crée une occurrence du 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 uniquement communiquer entre des fichiers SWF exécutés sur le même périphérique client, mais ils peuvent concerner différentes applications. 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.

Utilisez les méthodes `LocalConnection.send()` et `LocalConnection.connect()` pour envoyer et recevoir des données. Remarque : 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 mettre en œuvre vos mesures de sécurité. Pour plus d'informations, consultez la section consacrée au paramètre `connectionName` dans `LocalConnection.send()` ainsi que les entrées `LocalConnection.allowDomain` et`LocalConnection.domain()`.

Disponibilité

Flash Lite 3.1

Résumé des propriétés

constructor (propriété Object.constructor), __proto__ (Object.__proto__, propriété), prototype (propriété Object.prototype), __resolve (Object.__resolve, propriété)

Résumé des événements

Evénement	Description
allowDomain = function([sendingDomain:]) {} String	Invoqué chaque fois qu'un objet LocalConnection reçoit une requête pour appeler une méthode à partir d'un autre objet LocalConnection.
allowInsecureDomain = function([sendingDomain:]) {} String	Invoqué à chaque fois qu'un objet LocalConnection de réception, 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.

Récapitulatif 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).

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 = 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. 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 Lite 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 ou une version antérieure (Flash Lite 1.x par exemple), le paramètre `sendingDomain` contient le superdomaine de l'appelant. Dans les fichiers créés pour Flash Player 7 ou versions ultérieures (Flash Lite 2.x et 3.x par exemple), le paramètre `sendingDomain` contient le superdomaine 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 versions ultérieures (Flash Lite 2.x et 3.x par exemple), 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 `MovieClip.La propriété _url` dans votre implémentation de 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 la fin du chargement d'un fichier SWF enfant consiste à utiliser `MovieClipLoader.onLoadComplete`.

Le cas contraire peut aussi se présenter : 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

Flash Lite 3.1

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);
```

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");
```

Lors de la publication pour Flash Player 7 ou versions ultérieures (Flash Lite 2.x et 3.x), la correspondance exacte de domaine 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 aussi

`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 = 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. 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 versions ultérieures (Flash Lite 2.x et 3.x par exemple) 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 versions ultérieures.

Disponibilité

Flash Lite 3.1

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");
```

Voir aussi

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

Flash Lite 3.1

Voir aussi

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 qui appelle cette commande est nommé 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 Lite 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 dans l'ordinateur client, la valeur affectée au `superdomain` est `"localhost"`.)

De même, par défaut, Flash Lite ne permet à l'objet LocalConnection de réception de n'accepter que les commandes provenant d'objets LocalConnection d'envoi dont le nom de connexion correspond également à une valeur `"`*superdomain*`:connectionName"`. Ainsi, Flash Lite facilite la communication entre les fichiers SWF situés dans le même domaine.

Si vous implémentez une communication uniquement entre des fichiers SWF appartenant au même domaine, spécifiez pour `connectionName` une chaîne qui ne commence pas par un caractère de soulignement (_) et qui ne spécifie pas un nom de domaine (par exemple, `"myDomain:connectionName"`). Utilisez la même chaÃ(r)ne 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. Les cas de figure possibles sont les suivants :

- Si la chaîne dédiée à *connectionName* ne commence pas par un caractère de soulignement (_), Flash Lite ajoute un préfixe au superdomaine et deux points (par exemple, `"myDomain:connectionName"`). Vous avez ainsi la garantie que votre connexion n'entrera pas en conflit avec les connexions de même nom dans d'autres domaines. Cependant, 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 dans un autre domaine, Flash Player 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 caractère de soulignement (par exemple, `"_connectionName"`), Flash Lite ne lui ajoute pas de préfixe. 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 section consacrée à `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`. Toute chaîne associée à `connectionName` contenant deux-points n'est pas valide.*

Disponibilité
Flash Lite 3.1

Paramètres
connectionName : String - Chaîne correspondant au nom de connexion spécifié dans la commande `LocalConnection.send()` qui souhaite communiquer avec *receiving_lc*.

Valeur renvoyée
Boolean - Valeur booléenne `true` si aucun autre processus en cours d'exécution sur le même ordinateur client n'a déjà appelé cette commande en 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.

Voir aussi

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 ou versions antérieures (Flash Lite 1.x par exemple), la chaîne renvoyée est le superdomaine du fichier SWF actuel. Par exemple, si le fichier SWF se trouve à l'adresse www.adobe.com, cette commande renvoie `"adobe.com"`.

Dans les fichiers SWF publiés pour Flash Player 7 ou versions ultérieures (Flash Lite 2.x et 3.x par exemple), la chaîne renvoyée est le domaine exact du fichier SWF actuel. Par exemple, si le fichier SWF se trouve à l'adresse www.adobe.com, cette commande renvoie `"www.adobe.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

Flash Lite 3.1

Valeur renvoyée

String - 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 example.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=="example.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.example.com will be accepted, but those from one posted at
// a different subdomain, e.g. test.example.com, will not.
var my_lc:LocalConnection = new LocalConnection();
my_lc.allowDomain = function(sendingDomain):String{
    return (sendingDomain==this.domain() || sendingDomain=="www.example.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 Lite 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 devant recevoir la réponse (`"result"`) et les valeurs que `aSum` doit utiliser (123 et 456).
- La méthode `aSum` (ligne 6) est invoquée avec les valeurs suivantes : sender = `"mydomain.com:result"`, `replyMethod` = `"aResult"`, `n1` = 123 et `n2` = 456. Elle exécute ensuite 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 aussi

allowDomain (gestionnaire LocalConnection.allowDomain), connect (méthode LocalConnection.connect)

constructeur LocalConnection

```
public LocalConnection()
```

Crée un objet LocalConnection.

Disponibilité

Flash Lite 3.1

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 aussi

connect (méthode LocalConnection.connect), send (méthode LocalConnection.send)

onStatus (gestionnaire LocalConnection.onStatus)

```
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. 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.

En plus de ce gestionnaire `onStatus`, Flash propose également 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.

Dans la plupart des cas, vous implémentez ce gestionnaire uniquement pour répondre à des conditions d'erreur, comme indiqué dans l'exemple suivant.

Disponibilité

Flash Lite 3.1

Paramètres

infoObject : Object - Paramètre défini selon le message de statut. 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 aussi

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 par un objet LocalConnection de réception. L'objet qui appelle cette méthode est l'objet LocalConnection d'envoi. Les fichiers SWF qui contiennent les objets d'envoi et de réception doivent s'exécuter sur le même périphérique 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 Lite 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 caractère 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 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.*

Lorsque vous employez cette méthode, tenez compte du modèle de sécurité de Flash Lite. 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 les sections suivantes :

- Chapitre 17, « Fonctionnement de la sécurité » du guide *Formation à ActionScript 2.0 dans Flash*
- Le livre blanc Sécurité de Flash Player 8 à l'adresse http://www.macromedia.com/go/fp8_security
- La présentation technique des API relatives à la sécurité de Flash Player 8 à l'adresse http://www.macromedia.com/go/fp8_security_apis

Disponibilité

Flash Lite 3.1

Paramètres

connectionName : String - Chaîne correspondant au nom de connexion spécifié dans la commande `LocalConnection.connect()` qui souhaite communiquer avec *sending_lc*.

methodName : String - 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.

Valeur renvoyée

Boolean - Valeur booléenne `true` si Flash peut exécuter la requête ; sinon `false`.

Remarque : *Une valeur true renvoyée n'indique pas forcément que Flash Lite s'est connecté correctement à un objet LocalConnection de réception. Cela indique uniquement que la syntaxe de la commande est correcte. 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 des domaines spécifiés, consultez `LocalConnection.domain()`.

Voir aussi

allowDomain (gestionnaire LocalConnection.allowDomain), connect (méthode LocalConnection.connect), domain (méthode LocalConnection.domain), onStatus (gestionnaire LocalConnection.onStatus)

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 méthodes de la classe Math sont statiques et doivent être appelées à l'aide de la syntaxe `Math.method()` 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 en 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 (l'angle étant exprimé en radians plutôt qu'en `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

Flash Lite 2.0

Résumé des propriétés

Modificateurs	Propriété	Description
`statique`	E : Number	Constante mathématique pour la base des logarithmes népériens, exprimée en *e*.
`statique`	LN10 : Number	Constante mathématique pour le logarithme népérien de 10, exprimée sous la forme de loge10, d'une valeur approximative de 2,302585092994046.
`statique`	LN2 : Number	Constante mathématique pour le logarithme népérien de 2, exprimée sous la forme de loge2, d'une valeur approximative de 0,6931471805599453.
`statique`	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.

Modificateurs	Propriété	Description
statique	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.
statique	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.
statique	SQRT1_2 : Number	Constante mathématique pour la racine carrée de un demi, d'une valeur approximative de 0,7071067811865476.
statique	SQRT2 : Number	Constante mathématique pour la racine carrée de 2, d'une valeur approximative de 1,4142135623730951.

Propriétés héritées de la classe Object

constructor (propriété Object.constructor), __proto__ (Object.__proto__, propriété). prototype (propriété Object.prototype), __resolve (Object.__resolve, propriété)

Résumé de la méthode

Modificateurs	Signature	Description
statique	abs (x : Number) : Number	Calcule et renvoie la valeur absolue du nombre spécifié par le paramètre x.
statique	acos (x : Number) : Number	Calcule et renvoie l'arc cosinus du nombre spécifié dans le paramètre x, en radians.
statique	asin (x : Number) : Number	Calcule et renvoie l'arc sinus du nombre spécifié dans le paramètre x, en radians.
statique	atan (tangent : Number) : Number	Calcule et renvoie la valeur, en radians, de l'angle dont la tangente est spécifiée dans le paramètre tangent.
statique	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).
statique	ceil (x : Number) : Number	Renvoie la valeur maximale de l'expression ou du nombre spécifié.
statique	cos (x : Number) : Number	Calcule et renvoie le cosinus de l'angle spécifié en radians.
statique	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.
statique	floor (x : Number) : Number	Renvoie la valeur minimale de l'expression ou du nombre spécifié dans le paramètre x.
statique	log (x : Number) : Number	Renvoie le logarithme népérien du paramètre x.
statique	max (x : Number, y : Number) : Number	Evalue x et y, puis renvoie la valeur la plus élevée.
statique	min (x : Number, y : Number) : Number	Evalue x et y, puis renvoie la valeur la plus faible.
statique	pow (x : Number, y : Number) : Number	Calcule et renvoie x à la puissance de y.

Modificateurs	Signature	Description
statique	random() : Number	Renvoie un nombre pseudo-aléatoire n, où 0 <= n < 1.
statique	round(x:Number) : Number	Arrondit la valeur du paramètre x à l'entier immédiatement supérieur ou inférieur et la renvoie.
statique	sin(x:Number) : Number	Calcule et renvoie le sinus de l'angle spécifié en radians.
statique	sqrt(x:Number) : Number	Calcule et renvoie la racine carrée du nombre spécifié.
statique	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)

```
public static abs(x:Number) : Number
```

Calcule et renvoie la valeur absolue du nombre spécifié par le paramètre x.

Disponibilité
Flash Lite 2.0

Paramètres
x : Number - Un nombre.

Valeur renvoyée
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)

```
public static acos(x:Number) : Number
```

Calcule et renvoie l'arc cosinus du nombre spécifié dans le paramètre x, en radians.

Disponibilité
Flash Lite 2.0

Paramètres

x : Number - Un nombre compris entre -1,0 et 1,0.

Valeur renvoyée

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 aussi

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)

```
public static asin(x:Number) : Number
```

Calcule et renvoie l'arc sinus du nombre spécifié dans le paramètre x, en radians.

Disponibilité

Flash Lite 2.0

Paramètres

x : Number - Un nombre compris entre -1,0 et 1,0.

Valeur renvoyée

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 aussi

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)

```
public static 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
Flash Lite 2.0

Paramètres
tangent : Number - Un nombre représentant la tangente d'un angle.

Valeur renvoyée
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 aussi
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)

```
public 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). La valeur renvoyée est comprise entre pi positif et pi négatif.

Disponibilité
Flash Lite 2.0

Paramètres
y : Number - Un nombre spécifiant la coordonnée *y* du point.

x : Number - Un nombre spécifiant la coordonnée *x* du point.

Valeur renvoyée
Number - Un nombre.

Exemple
L'exemple suivant renvoie l'angle, en radians, du point spécifié par les coordonnées (0, 10), sachant 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 aussi
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)

```
public static ceil(x:Number) : Number
```

Renvoie la valeur maximale de l'expression ou du nombre spécifié. La valeur maximale d'un nombre est l'entier le plus proche supérieur ou égal au nombre.

Disponibilité
Flash Lite 2.0

Paramètres
x : Number - Nombre ou expression.

Valeur renvoyée
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 aussi
`floor (méthode Math.floor)`, `round (méthode Math.round)`

cos (méthode Math.cos)

```
public static 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
Flash Lite 2.0

Paramètres
x : Number - Un nombre représentant un angle mesuré en radians.

Valeur renvoyée
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 Lite Player renvoie un nombre le plus proche de zéro, mais pas égal à zéro.

Voir aussi

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)

```
public static E : Number
```

Constante mathématique pour la base des logarithmes népériens, exprimée en *e*. La valeur approximative de *e* est 2,71828182845905.

Disponibilité

Flash Lite 2.0

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)

```
public 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`. La constante `Math.E` peut renvoyer la valeur *e*.

Disponibilité

Flash Lite 2.0

Paramètres

x : Number - Exposant ; un nombre ou une expression.

Valeur renvoyée

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 aussi

E (propriété Math.E)

floor (méthode Math.floor)

```
public static floor(x:Number) : Number
```

Renvoie la valeur minimale de l'expression ou du nombre spécifié dans le paramètre `x`. La valeur minimale est l'entier le plus proche inférieur ou égal à l'expression ou au nombre spécifié.

Disponibilité

Flash Lite 2.0

Paramètres

x : Number - Nombre ou expression.

Valeur renvoyée

Number - Un 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)

```
public static LN10 : Number
```

Constante mathématique pour le logarithme népérien de 10, exprimée sous la forme de loge10, d'une valeur approximative de 2,302585092994046.

Disponibilité

Flash Lite 2.0

Exemple

Cet exemple présente la valeur de `Math.LN10`.

```
trace(Math.LN10);
// output: 2.30258509299405
```

LN2 (propriété Math.LN2)

```
public static LN2 : Number
```

Constante mathématique pour le logarithme népérien de 2, exprimée sous la forme de loge2, d'une valeur approximative de 0,6931471805599453.

Disponibilité
Flash Lite 2.0

log (méthode Math.log)

```
public static log(x:Number) : Number
```

Renvoie le logarithme népérien du paramètre `x`.

Disponibilité
Flash Lite 2.0

Paramètres
x : Number - Un nombre ou une expression d'une valeur supérieure à 0.

Valeur renvoyée
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)

```
public 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.

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
Flash Lite 2.0

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)

```
public 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.

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
Flash Lite 2.0

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)

```
public static max(x:Number, y:Number) : Number
```

Evalue `x` et `y`, puis renvoie la valeur la plus élevée.

Disponibilité
Flash Lite 2.0

Paramètres
x : Number - Nombre ou expression.

y : Number - Nombre ou expression.

Valeur renvoyée
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 aussi
`min (méthode Math.min)`

min (méthode Math.min)

```
public static min(x:Number, y:Number) : Number
```

Evalue `x` et `y`, puis renvoie la valeur la plus faible.

Disponibilité
Flash Lite 2.0

Paramètres
x : Number - Nombre ou expression.

y : Number - Nombre ou expression.

Valeur renvoyée

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 aussi

max (méthode Math.max)

PI (propriété Math.PI)

```
public 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.

Disponibilité

Flash Lite 2.0

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)

```
public static pow(x:Number, y:Number) : Number
```

Calcule et renvoie `x` à la puissance de `y`.

Disponibilité
Flash Lite 2.0

Paramètres
x : Number - Un nombre à élever à une puissance.

y : Number - Un nombre spécifiant la puissance à laquelle le paramètre x est élevé.

Valeur renvoyée
Number - Un nombre.

random (méthode Math.random)

```
public static random() : Number
```

Renvoie un nombre pseudo-aléatoire n, sachant que 0 <= n < 1. Le nombre renvoyé est un nombre pseudo-aléatoire car il n'est pas produit par un phénomène naturel parfaitement aléatoire, telle qu'une désintégration radioactive.

Disponibilité
Flash Lite 2.0

Valeur renvoyée
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)

```
public static round(x:Number) : Number
```

Arrondit la valeur du paramètre x à l'entier immédiatement supérieur ou inférieur et la renvoie. 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
Flash Lite 2.0

Paramètres
x : Number - Un nombre.

Valeur renvoyée
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 aussi

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

sin (méthode Math.sin)

```
public static 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

Flash Lite 2.0

Paramètres

x : Number - Un nombre représentant un angle mesuré en radians.

Valeur renvoyée

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 aussi

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)

```
public static sqrt(x:Number) : Number
```

Calcule et renvoie la racine carrée du nombre spécifié.

Disponibilité

Flash Lite 2.0

Paramètres

x : Number - Une expression ou un nombre supérieur ou égal à 0.

Valeur renvoyée

Number - Un nombre si le paramètre *x* est supérieur ou égal à zéro ; NaN (pas un nombre) sinon.

SQRT1_2 (propriété Math.SQRT1_2)

```
public static SQRT1_2 : Number
```

Constante mathématique pour la racine carrée de un demi, d'une valeur approximative de 0,7071067811865476.

Disponibilité

Flash Lite 2.0

Exemple

Cet exemple présente la valeur de `Math.SQRT1_2`.

```
trace(Math.SQRT1_2);
// Output: 0.707106781186548
```

SQRT2 (propriété Math.SQRT2)

```
public static SQRT2 : Number
```

Constante mathématique pour la racine carrée de 2, d'une valeur approximative de 1,4142135623730951.

Disponibilité

Flash Lite 2.0

Exemple

Cet exemple présente la valeur de `Math.SQRT2`.

```
trace(Math.SQRT2);
// Output: 1.4142135623731
```

tan (méthode Math.tan)

```
public static 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
Flash Lite 2.0

Paramètres
x : Number - Un nombre représentant un angle mesuré en radians.

Valeur renvoyée
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 aussi
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 transformations 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 suffit de créer l'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 :

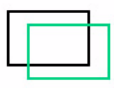
$$\begin{bmatrix} a & b & t_x \\ c & d & t_y \\ u & v & w \end{bmatrix}$$

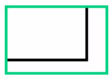
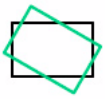
Dans le cas des matrices de transformation classiques, les propriétés `u`, `v` et `w` sont dotées de fonctionnalités supplémentaires. La classe Matrix fonctionne uniquement dans un espace bidimensionnel ; ainsi, elle suppose toujours que les valeurs des propriétés `u` et `v` sont 0,0, et que la valeur de la propriété `w` est 1,0. En d'autres termes, les valeurs réelles de la matrice sont les suivantes :

$$\begin{bmatrix} a & b & t_x \\ c & d & t_y \\ 0 & 0 & 1 \end{bmatrix}$$

Vous pouvez obtenir et définir les valeurs des six autres propriétés d'un objet Matrix : `a`, `b`, `c`, `d`, `tx` et `ty`.

La classe Matrix prend en charge les quatre principaux types de fonctions de transformation : translation, redimensionnement, rotation et 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.

Transformation	Méthode	Valeurs de matrice	Résultat affiché	Description
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*.
Rotation	`rotate(q)`	$\begin{bmatrix} \cos(q) & \sin(q) & \\ -\sin(q) & \cos(q) & \\ 0 & 0 & \end{bmatrix}$		Fait pivoter l'image selon un angle `q`, mesuré en radians
Inclinaison ou cisaillement	Aucun ; il est nécessaire de 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 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

Flash Lite 3.1

Voir aussi

`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	Valeur, dans la première ligne et la première colonne de l'objet Matrix, affectant le positionnement des pixels sur l'axe des *x* lors du redimensionnement ou de la rotation d'une image.
	b:Number	Valeur, dans la première ligne et la deuxième colonne de l'objet Matrix, affectant le positionnement des pixels sur l'axe des *y* lors de la rotation ou de l'inclinaison d'une image.
	c :Number	Valeur, dans la deuxième ligne et la première colonne de l'objet Matrix, affectant le positionnement des pixels sur l'axe des *x* lors de la rotation ou de l'inclinaison d'une image.
	d: Number	Valeur, dans la deuxième ligne et la deuxième colonne de l'objet Matrix, affectant le positionnement des pixels sur l'axe des *y* lors du redimensionnement ou de la rotation d'une image.
	tx: Number	Distance de translation de chaque point sur l'axe des *x*.
	ty: Number	Distance de translation de chaque point sur l'axe *y*.

`constructor (propriété Object.constructor), __proto__ (Object.__proto__, propriété), prototype (propriété Object.prototype), __resolve (Object.__resolve, propriété)`

Récapitulatif des constructeurs

Signature	Description
Matrix([a:Number, [b:Number], [c:Number], [d:Number], [tx:Number], [ty:Number])	Crée un 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 et 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().

Modificateurs	Signature	Description
	deltaTransformPoint(pt:Point) :Point	Etant donné un point dans l'espace de coordonnées de prétransformation, cette méthode renvoie les coordonnées de ce point après 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
```

Valeur, dans la première ligne et la première colonne de l'objet Matrix, affectant le positionnement des pixels sur l'axe des *x* lors du redimensionnement ou de la rotation d'une image.

Disponibilité

Flash Lite 3.1

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
```

Valeur, dans la première ligne et la deuxième colonne de l'objet Matrix, affectant le positionnement des pixels sur l'axe des *y* lors de la rotation ou de l'inclinaison d'une image.

Disponibilité
Flash Lite 3.1

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
```

Valeur, dans la deuxième ligne et la première colonne de l'objet Matrix, affectant le positionnement des pixels sur l'axe des *x* lors de la rotation ou de l'inclinaison d'une image.

Disponibilité
Flash Lite 3.1

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
Flash Lite 3.1

Valeur renvoyée
Matrix - 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 et 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

Flash Lite 3.1

Paramètres

m : Matrix - 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`. La troisième matrice est créée à l'aide de la méthode `concat()` sur `rotate45Matrix` et `doubleScaleMatrix` pour créer `scaleAndRotateMatrix`. Cette matrice est alors 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()` 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
Flash Lite 3.1

Paramètres
scaleX : Number - Facteur à appliquer au redimensionnement horizontal.

scaleY : Number - 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, ainsi que les emplacements *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;
```

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
Flash Lite 3.1

Paramètres
width : Number - Largeur de la zone de dégradé.

height : Number - Hauteur de la zone de dégradé.

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 aussi

`beginGradientFill (méthode MovieClip.beginGradientFill)`

d (propriété Matrix.d)

`public d : Number`

Valeur, dans la deuxième ligne et la deuxième colonne de l'objet Matrix, affectant le positionnement des pixels sur l'axe des *y* lors du redimensionnement ou de la rotation d'une image.

Disponibilité

Flash Lite 3.1

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
```

Etant donné un point dans l'espace de coordonnées de prétransformation, cette méthode renvoie les coordonnées de ce point après 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

Flash Lite 3.1

Paramètres

pt : Point - Un objet Point.

Valeur renvoyée

Point - 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.

Faisant suite à l'appel de la méthode `identity()`, la matrice obtenue présente les 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 :

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Disponibilité
Flash Lite 3.1

Exemple

L'exemple suivant démontre que l'appel de la méthode `identity()` convertit l'objet Matrix appelant 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 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

Flash Lite 3.1

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 `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])
```

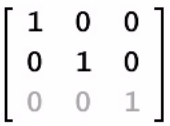
Crée un objet Matrix avec les paramètres spécifiés. Dans la notation des matrices, les propriétés sont organisées comme suit :

$$\begin{bmatrix} a & b & t_x \\ c & d & t_y \\ 0 & 0 & 1 \end{bmatrix}$$

Si vous ne transmettez aucun paramètre au nouveau constructeur `Matrix()`, celui-ci crée une matrice d'identité dotée des 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
Flash Lite 3.1

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
Flash Lite 3.1

Paramètres
angle : Number - 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 manuellement.

```
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 l'emplacement horizontal 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
Flash Lite 3.1

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
Flash Lite 3.1

Valeur renvoyée
« String » à la page 598 - Chaîne répertoriant 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
Flash Lite 3.1

Paramètres
pt : Point - Le point (*x*,*y*) à transformer.

Valeur renvoyée
Point (flash.geom.Point) - 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` de 300, ce qui permet d'obtenir une valeur totale de 450.

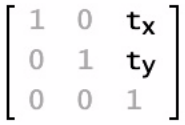
```
import flash.geom.Matrix;
import flash.geom.Point;

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*.

La méthode `translate()` modifie les propriétés `tx` et `ty` de l'objet Matrix. Dans la notation des matrices, ceci est illustré comme suit :



Disponibilité
Flash Lite 3.1

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` avec les valeurs x:100 et y:50. La méthode `translate()` affecte les propriétés de translation `tx` et `ty`, mais pas les propriétés `a`, `b`, `c` et `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
```

Distance de translation de chaque point sur l'axe des *x*. Elle représente la valeur dans la troisième ligne et la première colonne de l'objet Matrix.

Disponibilité
Flash Lite 3.1

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
```

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

Flash Lite 3.1

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
```

Mouse

```
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 ajouter et supprimer des écouteurs et pour traiter les événements de la souris.

Les membres de cette classe ne sont pris en charge par Flash Lite que si `System.capabilities.hasMouse` a pour valeur `true` ou si `System.capabilities.hasStylus` a pour valeur `true`.

Disponibilité

Flash Lite 2.0

Résumé des propriétés

Propriétés héritées de la classe Object

constructor (propriété Object.constructor), __proto__ (Object.__proto__, propriété), prototype (propriété Object.prototype), __resolve (Object.__resolve, propriété)

Résumé des événements

Evénement	Description
onMouseDown = fonction() {}	Signalé lorsque l'utilisateur appuie sur le bouton de la souris.
onMouseMove = fonction() {}	Signalé lorsque la souris bouge.
onMouseUp = fonction() {}	Signalé lorsque le bouton de la souris est relâché.

Résumé de la méthode

Modificateurs	Signature	Description
statique	addListener(listener: Object) : Void	Enregistre un objet afin de recevoir les notifications des écouteurs onMouseDown, onMouseMove et onMouseUp.
statique	removeListener(listener:Object) : Boolean	Supprime un objet précédemment enregistré 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 Mouse.addListener)

```
public static addListener(listener:Object) : Void
```

Enregistre un objet afin de recevoir les notifications des écouteurs onMouseDown, onMouseMove et onMouseUp.

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 ou onMouseUp de tous les objets écouteur enregistrés auprès de cette méthode est appelée. Plusieurs objets peuvent écouter les notifications de souris. Si l'écouteur est déjà enregistré, aucun changement ne se produit.

Remarque : Cette méthode n'est prise en charge par Flash Lite que si System.capabilities.hasMouse est défini sur true ou si System.capabilities.hasStylus est défini sur true.

Disponibilité
Flash Lite 2.0

Paramètres
listener : Object

Exemple
Cet exemple envoie la position du curseur vers le panneau Sortie.

```
// Create a mouse listener object.
var mouseListener:Object = new Object();

mouseListener.onMouseMove = function() {
    trace(_xmouse);
    trace(_ymouse);
};
Mouse.addListener(mouseListener);
```

Voir aussi

onMouseDown (écouteur d'événement Mouse.onMouseDown), onMouseMove (écouteur d'événement Mouse.onMouseMove), onMouseUp (écouteur d'événement Mouse.onMouseUp)

onMouseDown (écouteur d'événement Mouse.onMouseDown)

```
onMouseDown = function() {}
```

Signalé lorsque l'utilisateur appuie sur le bouton de la souris. Pour utiliser l'écouteur `onMouseDown`, vous devez créer un objet écouteur. Définissez une fonction pour `onMouseDown` et appelez `addListener()` pour enregistrer l'écouteur auprès de l'objet Mouse, comme indiqué dans l'exemple de 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.

Cet écouteur d'événement n'est pris en charge par Flash Lite que si `System.capabilities.hasMouse` a pour valeur `true` ou si `System.capabilities.hasStylus` a pour valeur `true`.

Disponibilité

Flash Lite 2.0

Exemple

L'exemple suivant utilise l'API de dessin pour dessiner un rectangle lorsque l'utilisateur appuie sur le bouton de la souris, déplace la souris, puis relâche son bouton lors de l'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);
```

Voir aussi

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 écouteur. Vous pouvez ensuite définir une fonction pour `onMouseMove` et utiliser `addListener()` pour enregistrer l'écouteur auprès de l'objet Mouse, comme indiqué dans l'exemple de 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.

Cet écouteur d'événement n'est pris en charge par Flash Lite que si `System.capabilities.hasMouse` a pour valeur `true`.

Disponibilité

Flash Lite 2.0

Exemple

L'exemple suivant utilise le pointeur de la souris comme outil pour dessiner des lignes avec `onMouseMove` et l'API de dessin. L'utilisateur dessine une ligne en déplaçant le pointeur.

```
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);
```

L'exemple suivant définit les positions `x` et `y` de l'occurrence de clip `pointer_mc` sur les coordonnées `x` et `y` du pointeur. Le périphérique doit prendre en charge les stylets ou les souris pour que l'exemple fonctionne. Dans cet exemple, vous créez un clip et définissez son identificateur Linkage sur `pointer_id`. Ajoutez ensuite le code ActionScript suivant à l'image 1 du scénario :

```
this.attachMovie("pointer_id", "pointer_mc", this.getNextHighestDepth());
var mouseListener:Object = new Object();
mouseListener.onMouseMove = function() {
    pointer_mc._x = _xmouse;
    pointer_mc._y = _ymouse;
};
Mouse.addListener(mouseListener);
```

Voir aussi

`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 écouteur. Vous pouvez ensuite définir une fonction pour `onMouseUp` et utiliser `addListener()` pour enregistrer l'écouteur auprès de l'objet Mouse, comme indiqué dans l'exemple de 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.

Cet écouteur d'événement n'est pris en charge par Flash Lite que si `System.capabilities.hasMouse` a pour valeur `true` ou si `System.capabilities.hasStylus` a pour valeur `true`.

Disponibilité

Flash Lite 2.0

Exemple

L'exemple suivant utilise le pointeur de la souris comme outil pour dessiner des lignes avec `onMouseMove` et l'API de dessin. L'utilisateur dessine une ligne en déplaçant le pointeur et cesse de dessiner 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);
```

Voir aussi

`addListener (méthode Mouse.addListener)`

removeListener (méthode Mouse.removeListener)

```
public static removeListener(listener:Object) : Boolean
```

Supprime un objet précédemment enregistré avec `addListener()`.

Remarque : Cette méthode n'est prise en charge par Flash Lite que si `System.capabilities.hasMouse` est défini sur `true` ou si `System.capabilities.hasStylus` est défini sur `true`.

Disponibilité

Flash Lite 2.0

Paramètres

listener : Object

Valeur renvoyée

Boolean - Si l'objet écouteur est supprimé avec succès, la méthode renvoie `true` ; si l'objet écouteur n'a pas été supprimé (par exemple, s'il 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 avec le pointeur de la souris. Un bouton efface toutes les 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. Ajoutez l'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);
```

MovieClip

```
Object
    |
    +-MovieClip

public dynamic class MovieClip
extends Object
```

Utilisez la classe MovieClip pour manipuler les clips avec ActionScript. Aucun constructeur n'existe pour la classe MovieClip. Pour créer une nouvelle occurrence de clip, effectuez l'une des opérations suivantes :

- Dessinez un clip sur la scène dans l'outil de programmation Flash et donnez-lui un nom d'occurrence dans l'inspecteur Propriétés.
- Appelez la méthode `attachMovie()` pour créer une occurrence de clip en fonction d'un symbole de clip provenant de la bibliothèque.
- Appelez la méthode `createEmptyMovieClip()` pour créer une nouvelle occurrence de clip vide en tant qu'enfant reposant sur un autre clip.
- Appelez la méthode `duplicateMovieClip()` pour créer une occurrence de clip à partir d'un autre clip.

Disponibilité

Flash Lite 2.0

Résumé des propriétés

Modificateurs	Propriété	Description
	_alpha : Number	La valeur de transparence alpha 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.
	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 d'entrée.
	_framesloaded, : Number [lecture seule]	Le nombre d'images à charger à partir d'un fichier SWF en diffusion continue.
	_height : Number	Hauteur du clip, en pixels.
	_highquality : Number	**Déconseillé depuis Flash Player 7.** Cette propriété est déconseillée au profit de `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	Une valeur booléenne qui spécifie ce à quoi `_root` se réfère lorsqu'un fichier SWF est chargé dans un clip.
	_name : String	Le nom d'occurrence du clip.
	_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.
	_soundbuftime : Number	Spécifie le nombre de secondes pendant lequel les sons sont chargés en mémoire tampon avant d'être diffusés en continu.

Modificateurs	Propriété	Description
	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 un événement de relâchement de la souris ou du stylet.
	_url : String [lecture seule]	Récupère l'URL du fichier SWF, JPEG, GIF ou PNG ayant servi à télécharger le clip.
	_visible : Boolean	Valeur booléenne indiquant si le clip est visible.
	_width : Number	Largeur du clip, en pixels.
	_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

`constructor (propriété Object.constructor), __proto__ (Object.__proto__, propriété), prototype (propriété Object.prototype), __resolve (Object.__resolve, propriété)`

Résumé des événements

Evénement	Description
onData = `fonction() {}`	Appelé lorsqu'un clip reçoit des données provenant d'un appel `MovieClip.loadVariables()` ou `MovieClip.loadMovie()`.
onDragOut = `fonction() {}`	Appelé lorsque l'utilisateur appuie sur le bouton de la souris et si le pointeur se déplace hors de l'objet.
onDragOver = `fonction() {}`	Appelé lorsque l'utilisateur fait glisser le pointeur hors du clip, puis sur le clip.

Evénement	Description
onEnterFrame = fonction() {}	Appelé à plusieurs reprises à la cadence du fichier SWF.
onKeyDown = fonction() {}	Appelé lorsqu'un clip reçoit le focus d'entrée et que l'utilisateur appuie sur une touche.
onKeyUp = fonction() {}	Appelé lorsqu'une touche est relâchée.
onKillFocus = function(newFocus: Object) {}	Appelé lorsqu'un clip perd le focus d'entrée.
onLoad = fonction() {}	Appelé lorsque le clip est instancié et apparaît dans le scénario.
onMouseDown = fonction() {}	Appelé lorsque vous appuyez sur le bouton de la souris.
onMouseMove = fonction() {}	Appelé lorsque la souris bouge.
onMouseUp = fonction() {}	Appelé lorsque vous relâchez le bouton de la souris.
onPress = fonction() {}	Appelé lorsque l'utilisateur clique sur le bouton de la souris quand le pointeur est placé sur un clip.
onRelease = fonction() {}	Appelé lorsque l'utilisateur relâche le bouton de la souris au-dessus d'un clip.
onReleaseOutside = fonction() {}	Appelé lorsque l'utilisateur a appuyé sur le bouton de la souris dans la zone occupée par un clip, puis l'a relâché en dehors de cette zone.
onRollOut = fonction() {}	Appelé lorsque le pointeur se déplace hors de la zone du clip.
onRollOver = fonction() {}	Appelé lorsque le pointeur se déplace au-dessus de la zone du clip.
onSetFocus = function(oldFocus: Object) {}	Appelé lorsqu'un clip reçoit le focus d'entrée.
onUnload = fonction() {}	Appelé dans la première image une fois la suppression du clip dans le scénario effectuée.

Résumé de la méthode

Modificateurs	Signature	Description
	attachMovie(id:String, name:String, depth:Number, [initObject:Object]) : MovieClip	Sélectionne un symbole dans la bibliothèque et l'associe au clip.
	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) : 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().
	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 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 à partir 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 rappel à beginFill() ou beginGradientFill().
	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.
	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.

Modificateurs	Signature	Description
	getInstanceAtDepth(depth:Number) : MovieClip	Permet de déterminer si une profondeur spécifique est déjà occupée par un clip.
	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 Lite rende le clip devant tous les autres objets sur les mêmes niveau et calque dans le clip actuel.
	getSWFVersion() : Number	Renvoie un entier qui indique la version de publication de Flash Lite pour le clip.
	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(bloc:Object) : Void	Commence la lecture du fichier SWF sur l'image spécifiée.
	gotoAndStop(bloc: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 `target` ou les paramètres de coordonnées `x` et `y`.
	lineStyle(thickness:Number, rgb:Number, alpha:Number, pixelHinting:Boolean, noScale:String, capsStyle:String, jointStyle:String, miterLimit:Number) : Void	Spécifie un style de trait pour les appels suivants de `lineTo()` et `curveTo()`, jusqu'à ce que vous appeliez `lineStyle()` avec des paramètres différents.
	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*).
	loadMovie(url:String, [method:String]) : Void	Charge un fichier SWF ou JPEG dans un clip Flash Lite 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], [gauche:Number], [haut:Number], [droite:Number], [bas:Number]) : Void	Permet à l'utilisateur de faire glisser le clip spécifié.
	stop() : Void	Arrête le clip en cours de lecture.
	stopDrag() : Void	Termine un appel de la 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 par 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 (MovieClip._alpha, propriété)

```
public _alpha : Number
```

Valeur de transparence alpha d'un 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.

Disponibilité

Flash Lite 2.0

Exemple

Le code suivant définit sur 50 % la propriété _alpha d'un clip intitulé rect_mc lorsque vous appuyez sur un bouton intitulé my_btn.

```
my_btn.onPress = function(){
     rect_mc._alpha  =   50 ;
}
my_btn.onRelease = function(){
    rect_mc._alpha = 100;
}
```

Voir aussi

_alpha (Button._alpha, propriété), _alpha (propriété TextField._alpha), _visible (propriété MovieClip._visible)

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 bitmaps 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, vous pouvez appeler 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.

Paramètres

bmp : `flash.display.BitmapData` - Image de 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'alignement des pixels sont `auto`, `always` et `never`. Le mode par défaut est `auto`.

smoothing : `Boolean` [facultatif] - Le mode de lissage est `true` si activé ou `false` si désactivé. Le mode par défaut est désactivé.

Disponibilité

Flash Lite 3.1

Exemple

Le code suivant crée un objet BitmapData et l'associe à un clip :

```
import flash.display.*;
this.createEmptyMovieClip("bmp1", 99);
var bmpData1:BitmapData = new BitmapData(200, 200, false, 0xaa3344);
bmp1.attachBitmap(bmpData1, 2, "auto", true);
```

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 symbole lié à `attachMovie()`.

Disponibilité

Flash Lite 2.0

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 les 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 deux occurrences d'un symbole portant l'identifiant de liaison « circle » à une occurrence de clip sur la scène :

```
this.attachMovie("circle", "circle1_mc", this.getNextHighestDepth());
this.attachMovie("circle", "circle2_mc", this.getNextHighestDepth(), {_x:50, _y:50});
```

Voir aussi

removeMovieClip (méthode MovieClip.removeMovieClip), unloadMovie (méthode MovieClip.unloadMovie) removeMovieClip, fonction

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 lorsque `MovieClip.endFill()` est appelé.

Disponibilité

Flash Lite 2.0

Paramètres

rgb : Number - Valeur colorimétrique hexadécimale ; par exemple, rouge correspond à 0xFF0000 et bleu à 0x0000FF, etc. 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();
```

Voir aussi

moveTo (méthode MovieClip.moveTo), endFill (méthode MovieClip.endFill), beginGradientFill (méthode MovieClip.beginGradientFill)

beginGradientFill (méthode MovieClip.beginGradientFill)

```
public beginGradientFill(fillType:String, colors:Array, alphas:Array, ratios:Array,
matrix:Object) : 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 « `linéaire` » or « `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

Flash Lite 2.0

Paramètres

fillType : String - Soit la chaîne « `linear` », soit la chaîne « `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.).

alphas : Array - Tableau de valeurs alpha pour les couleurs correspondantes dans le tableau `colors` ; les valeurs valides vont de 0 à 100. Si cette valeur est inférieure à 0, Flash utilise 0. Si elle est supérieure à 100, Flash applique 100.

ratios : Array - Tableau de rapports 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 sur 100 %.

matrix : Object - Matrice de transformation qui est un objet comportant l'un des deux jeux de propriétés suivants :

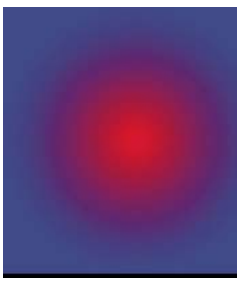
- `a`, `b`, `c`, `d`, `e`, `f`, `g`, `h`, `i`, qui peuvent servir à décrire une matrice 3 x 3 de la forme suivante :

  ```
  a b c
  d e f
  g h i
  ```

  L'exemple suivant utilise la méthode `beginGradientFill()` avec un paramètre `matrix` du type suivant :

  ```
  this.createEmptyMovieClip("gradient_mc", this.getNextHighestDepth());

  gradient_mc._x = -100;
  gradient_mc._y = -100;

  with (gradient_mc)
  {
      colors = [0xFF0000, 0x0000FF];
      fillType = "radial"
      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};
      beginGradientFill(fillType, colors, alphas, ratios, matrix);
      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 :

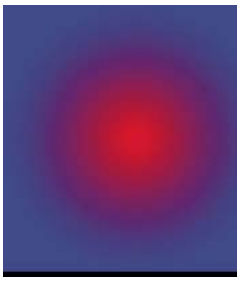
- `matrixType`, `x`, `y`, `w`, `h`, `r`.

  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 `beginGradientFill()` avec un paramètre `matrix` du type suivant :

```
this.createEmptyMovieClip("gradient_mc", this.getNextHighestDepth());

gradient_mc._x = -100;
gradient_mc._y = -100;

with (gradient_mc)
{
    colors = [0xFF0000, 0x0000FF];
    fillType = "radial"
    alphas = [100, 100];
    ratios = [0, 0xFF];
    matrix = {matrixType:"box", x:100, y:100, w:200, h:200,
        r:(45/180)*Math.PI};
    beginGradientFill(fillType, colors, alphas, ratios, matrix);
    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 :



Voir aussi

beginFill (méthode MovieClip.beginFill), endFill (méthode MovieClip.endFill), lineStyle (méthode MovieClip.lineStyle) lineTo (méthode MovieClip.lineTo), moveTo (méthode MovieClip.moveTo)

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

Flash Lite 2.0

Exemple

L'exemple suivant trace un cadre sur la scène. Lorsque l'utilisateur clique sur un bouton intitulé removeBox_btn, le graphique est supprimé.

```
this.createEmptyMovieClip("box_mc", 1);
drawBox(box_mc, 10, 10, 100, 100);
function drawBox(mc:MovieClip, x:Number, y:Number, w:Number, h:Number):Void {
    mc.lineStyle(5);
    mc.beginFill(0x009999);
    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();
}
removeBox_btn.onRelease = function(){
    box_mc.clear();
}
```

Voir aussi

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.

Disponibilité

Flash Lite 2.0

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";
```

Voir aussi

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 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 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 texte dans le clip. Chaque niveau de profondeur peut contenir uniquement un objet. Si vous créez un nouveau champ texte sur une profondeur disposant déjà d'un champ texte, le nouveau champ texte remplace le champ texte existant. Pour éviter d'écraser des champs texte existants, utilisez `MovieClip.getInstanceAtDepth()` afin de déterminer si une profondeur spécifique est déjà occupée, ou `MovieClip.getNextHighestDepth()` afin de déterminer la profondeur inoccupée la plus élevée. Le champ 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 texte. Les paramètres `width` et `height` correspondent aux propriétés `_width` et `_height`du champ texte.

Les propriétés par défaut d'un champ 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 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
Flash Lite 2.0

Paramètres
instanceName : String - Chaîne qui identifie le nom d'occurrence du nouveau champ texte.

depth : Number - Entier positif qui spécifie la profondeur du nouveau champ texte.

x : Number - Entier qui spécifie la coordonnée *x* du nouveau champ texte.

y : Number - Entier qui spécifie la coordonnée *y* du nouveau champ texte.

width : Number - Entier positif qui spécifie la largeur du nouveau champ texte.

height : Number - Entier positif qui spécifie la hauteur du nouveau champ texte.

Valeur renvoyée
TextField

Exemple
L'exemple suivant crée un champ 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 animations.fla du dossier d'exemples ActionScript disponible à l'adresse www.adobe.com/go/learn_fl_samples_fr. Téléchargez le fichier .zip, puis décompressez-le pour afficher le dossier correspondant à votre version d'ActionScript et consulter l'exemple.

Voir aussi
getInstanceAtDepth (méthode MovieClip.getInstanceAtDepth), getNextHighestDepth (méthode MovieClip.getNextHighestDepth) getNewTextFormat (méthode TextField.getNewTextFormat)

_currentframe (propriété MovieClip._currentframe)

```
public _currentframe : Number [read-only]
```

Renvoie le numéro de l'image dans laquelle se trouve la tête de lecture dans le scénario du clip.

Disponibilité
Flash Lite 2.0

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 tracez contient du contenu créé à l'aide des outils de dessin Flash, les appels de `curveTo()` sont tracés sous le 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

Flash Lite 2.0

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.

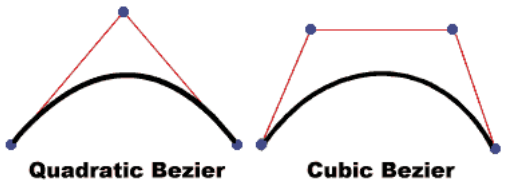
anchorY : Number - Entier qui spécifie la position verticale du point d'ancrage suivant par rapport au point d'alignement du clip parent.

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, '+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 d'exemples ActionScript disponible à l'adresse www.adobe.com/go/learn_fl_samples_fr. Téléchargez le fichier .zip, puis décompressez-le pour afficher le dossier correspondant à votre version d'ActionScript et consulter l'exemple.

Voir aussi

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 (MovieClip._droptarget, propriété)

```
public _droptarget : String [read-only]
```

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. 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 (ActionScript 2.0 ne prend pas en charge la syntaxe à barre oblique).

Remarque : Cette propriété n'est prise en charge par Flash Lite que si `System.capabilities.hasMouse` est définie sur `true` ou si `System.capabilities.hasStylus` est définie sur `true`.

Disponibilité

Flash Lite 2.0

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 aussi

startDrag (méthode MovieClip.startDrag), stopDrag (méthode MovieClip.stopDrag), eval, fonction

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

Flash Lite 2.0

Paramètres

name : String - Un identificateur 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;
}
```

Voir aussi

loadMovie (méthode MovieClip.loadMovie), removeMovieClip (méthode MovieClip.removeMovieClip), duplicateMovieClip, fonction

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

Flash Lite 2.0

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

Flash Lite 2.0

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();
```

Vous trouverez également un exemple dans le fichier drawingapi.fla du dossier d'exemples ActionScript disponible à l'adresse www.adobe.com/go/learn_fl_samples_fr. Téléchargez le fichier .zip, puis décompressez-le pour afficher le dossier correspondant à votre version d'ActionScript et consulter l'exemple.

Voir aussi

`beginFill (méthode MovieClip.beginFill)`, `beginGradientFill (méthode MovieClip.beginGradientFill)` `moveTo (méthode MovieClip.moveTo)`

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

Flash Lite 2.0

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 d'entrée. 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.

Remarque : Pour Flash Lite 2.0, lorsque la propriété `_focusrect` est désactivée (en d'autres termes, quand `MovieClip._focusrect` est défini sur `false`), le clip reçoit toujours les événements de pression de touche ou de souris.

D'autre part, pour Flash Lite 2.0, vous pouvez modifier la couleur du rectangle de focus à l'aide de la commande fscommand2 `SetFocusRectColor`. Ce comportement diffère de Flash Lite Player, où la couleur du rectangle de focus est limitée au jaune.

Disponibilité
Flash Lite 2.0

Exemple
Cet exemple démontre comment masquer le rectangle jaune qui entoure une occurrence de clip donnée dans un fichier SWF lorsque l'occurrence 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);
}
```

Pour tester le fichier SWF dans la fenêtre d'un navigateur, sélectionnez Fichier > Aperçu avant publication > HTML. Pour donner le focus au fichier SWF, cliquez 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.

Vous pouvez tester votre fichier SWF dans l'environnement de test. Sélectionnez Contrôle > Désactivez les raccourcis clavier dans l'environnement de test. Ceci permet d'afficher le rectangle de focus entourant les occurrences dans le fichier SWF.

Voir aussi
`_focusrect, propriété`, `_focusrect (propriété Button._focusrect)`

_framesloaded (propriété MovieClip._framesloaded)

```
public _framesloaded : Number [read-only]
```

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
Flash Lite 2.0

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 aussi
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 `MovieClip.lcalToGlobal()` 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
Flash Lite 2.0

Paramètres
bounds : Object - Chemin cible du scénario dont vous souhaitez utiliser le système de coordonnées comme 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 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 Sortie :

```
yMax --> 110
yMin --> 10
xMax --> 110
xMin --> 10
```

Voir aussi

`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

Flash Lite 2.0

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 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 aussi

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

Flash Lite 2.0

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 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 aussi

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 la profondeur est la plus importante s'affichent au premier plan.

Vous pouvez étendre les méthodes et les gestionnaires d'événements de la classe MovieClip en créant une sous-classe.

Disponibilité

Flash Lite 2.0

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 aussi

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

Flash Lite 2.0

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 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 aussi

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 supérieure 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.

Disponibilité

Flash Lite 2.0

Valeur renvoyée

Number - Entier reflétant le prochain index de profondeur disponible dont le rendu se situe au-dessus de tous les autres objets de même niveau et de 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 aussi

getDepth (méthode MovieClip.getDepth), getInstanceAtDepth (méthode MovieClip.getInstanceAtDepth) swapDepths (méthode MovieClip.swapDepths), attachMovie (méthode MovieClip.attachMovie), duplicateMovieClip (méthode MovieClip.duplicateMovieClip), createEmptyMovieClip (méthode MovieClip.createEmptyMovieClip)

getSWFVersion (méthode MovieClip.getSWFVersion)

```
public getSWFVersion() : Number
```

Renvoie un entier indiquant la version de Flash Lite 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 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

Flash Lite 2.0

Valeur renvoyée

Number - Entier spécifiant la version de Flash Lite 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);
```

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 une animation Flash doivent définir de façon explicite l'attribut `allowScriptAccess` pour autoriser ou bloquer la programmation de Flash Lite 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 hôte.
- Si `allowScriptAccess` n'est pas spécifié par une page HTML, la valeur par défaut `"sameDomain"` s'applique aux fichiers SWF de la version 8, et la valeur par défaut est `"always"` pour les fichiers SWF des versions antérieures.

Vous pouvez étendre les méthodes et les gestionnaires d'événements de la classe MovieClip en créant une sous-classe.

Disponibilité

Flash Lite 2.0

Paramètres

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

window : String [facultatif] - Paramètre spécifiant le nom, le cadre ou l'expression qui indique 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`, une nouvelle fenêtre ; `_parent`, le parent de l'image actuelle ; et `_top`, l'image de premier 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`. `GET` ajoute les variables à la fin de l'URL et est utilisée lorsque les variables sont peu nombreuses. `POST` place les variables dans un en-tête HTTP distinct et s'applique aux variables longues de type chaîne.

Exemple

L'ActionScript suivant crée une nouvelle instance de clip et ouvre le site Web Adobe dans une nouvelle fenêtre du navigateur :

```
this.createEmptyMovieClip("loader_mc", this.getNextHighestDepth());
loader_mc.getURL("http://www.adobe.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");
```

Voir aussi

getURL, fonction, 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 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 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 local 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 les clips. 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

Flash Lite 2.0

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);
```

Voir aussi

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
Flash Lite 2.0

Paramètres
bloc : Object - Nombre représentant le numéro d'image ou 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 la totalité des images n'est pas chargée, la propriété `_xscale` de l'occurrence de clip `loader` 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 aussi
gotoAndPlay, fonction, play, fonction

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 `gotoAndStop()`.

Vous pouvez étendre les méthodes et les gestionnaires d'événements de la classe MovieClip en créant une sous-classe.

Disponibilité
Flash Lite 2.0

Paramètres
bloc : 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 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 aussi

gotoAndStop, fonction, stop, fonction

_height (propriété MovieClip._height)

```
public _height : Number
```

Hauteur du clip, en pixels.

Disponibilité

Flash Lite 2.0

Exemple

L'exemple de code suivant affiche la hauteur et la largeur d'un clip dans le panneau 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);
```

Voir aussi

_width (propriété MovieClip._width)

_highquality (propriété MovieClip._highquality)

```
public _highquality : Number
```

Déconseillé depuis Flash Player 7. Cette propriété est déconseillée au profit de `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 permet de lisser 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
Flash Lite 2.0

Exemple
Le code ActionScript suivant permet d'appliquer la meilleure qualité possible d'anti-aliasing au fichier SWF.

```
my_mc._highquality = 2;
```

Voir aussi
`_quality (MovieClip._quality, propriété)`, `_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. La propriété `hitArea` peut être extraite d'un objet prototype.

Disponibilité
Flash Lite 2.0

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 régler la propriété `visible` du clip `circle_mc` sur `false` pour masquer la zone réactive de `square_mc`.

```
circle_mc._visible = false;
```

Voir aussi
`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 `target` ou les paramètres de coordonnées `x` et `y`.

Utilisation 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.

Utilisation 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` Coordonnée x de la zone active de 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

Flash Lite 2.0

Valeur renvoyée

Boolean - Valeur booléenne `true` si le clip recouvre la zone active spécifiée, sinon `false`.

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 aussi

getBounds (méthode MovieClip.etBounds),globalToLocal (méthode MovieClip.globalToLocal), localToGlobal (méthode MovieClip.localToGlobal)

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 appels suivants de `lineTo()` et `curveTo()`, jusqu'à ce que vous appeliez `lineStyle()` avec des paramètres différents. Vous pouvez appeler `lineStyle()` au cours du dessin afin de spécifier différents styles pour divers segments de ligne dans un tracé.

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

Flash Lite 2.0

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 dessinée. Si vous transmettez une valeur négative, Flash applique 0. La valeur 0 correspond à un filet ; l'épaisseur maximum est de 255. Si vous transmettez une valeur supérieure à 255, l'interprète de Flash applique une valeur de 255.

rgb : Number - Valeur colorimétrique hexadécimale 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 la couleur de la ligne ; les valeurs valides sont comprises entre 0 et 100. En l'absence de valeur, Flash applique 100 (uni). Si cette valeur est négative, Flash applique 0. Si elle est supérieure à 100, Flash applique 100.

pixelHinting : Boolean - 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. Si pixelHinting est défini sur true, Flash Lite Player propose des indices de largeur des pixels. Si pixelHinting est défini sur false, les courbes et les lignes droites risquent de ne pas être continues.

noScale : String - Chaîne indiquant comment redimensionner un trait. Les valeurs valides sont les suivantes :

"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 - Chaîne qui spécifie le type d'extrémité au bout des lignes. Les valeurs correctes sont : "round", "square" et "none". En l'absence de valeur, Flash utilise des extrémités rondes.

jointStyle : String - Chaîne qui indique le type d'aspect des liaisons utilisé aux angles. Les valeurs correctes sont : "round", "miter" et "bevel". En l'absence de valeurs, Flash utilise des liaisons rondes.

miterLimit : Number - Nombre qui indique la limite à laquelle une pointe est coupée. Les valeurs gérées sont comprises entre 1 et 255 (et les valeurs qui excèdent cette plage sont arrondies à 1 ou 255). Cette valeur n'est utilisée que si jointStyle est défini sur "miter". En l'absence de valeur, Flash utilise 3. La valeur miterLimit représente la longueur maximale d'une pointe au-delà du point où les lignes se rencontrent pour former une liaison. La valeur exprime un facteur du paramètre 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.

Exemple

Le code suivant dessine un triangle dont la ligne fait 5 pixels et est en magenta uni.

```
this.createEmptyMovieClip("triangle_mc", 1);
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);
```

Voir aussi

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 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

Flash Lite 2.0

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 aussi

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 les fichiers SWF ou JPEG dans le clip d'un fichier SWF lu par Flash Lite.

Conseil : Pour suivre la progression du téléchargement, appliquez la méthode `MovieClipLoader.loadClip()` en lieu et place de la méthode `loadMovie()`.

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.

Appelez `loadMovie()` pour charger n'importe quel format d'image pris en charge par le périphérique. Par exemple, si le périphérique cible prend en charge les fichiers PNG, le code suivant charge et affiche un fichier PNG se trouvant sur un serveur Web :

```
loadMovie("http://www.adobe.com/image.png", "image_target");
```

Pour déterminer les formats d'image pris en charge par le périphérique cible, utilisez la propriété `System.capabilities.imageMIMETypes` qui contient un tableau des types d'image MIME pris en charge. L'index des éléments du tableau correspond aux différents types MIME. Par exemple, le code suivant détermine si un périphérique prend en charge les images PNG avant que le périphérique ne tente de charger un fichier PNG externe :

```
if (System.capabilities.imageMIMETypes["image/png"]) {
    loadMovie("images/image.png", "mc_myPngImage");
}
```

Flash Lite ne permet à une application d'effectuer que cinq opérations `loadMovie()` pour une image donnée. Flash Lite ne permet que dix opérations `loadMovie()` à la fois. Supposez par exemple que votre application contient du code à l'image 1 qui charge six images JPEG externes :

```
image1.loadMovie("image1.jpg");
image2.loadMovie("image2.jpg");
image3.loadMovie("image3.jpg");
image4.loadMovie("image4.jpg");
image5.loadMovie("image5.jpg");
image6.loadMovie("image6.jpg"); // Won't load
```

Dans ce cas, seules les cinq premières images (de image1.jpg à image5.jpg) sont chargées. La dernière image (image6.jpg) n'est pas chargée car la limite de cinq connexions est atteinte. Vous pouvez alors fractionner les appels `loadMovie()` sur plusieurs images afin que chaque image contienne un maximum de cinq appels `loadMovie()`.

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 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.

Disponibilité

Flash Lite 2.0

Paramètres

url : String - 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:///`.

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 est utilisée pour des 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");
```

Voir aussi

`_lockroot (propriété MovieClip._lockroot)`, `unloadMovie (méthode MovieClip.unloadMovie)`, `loadVariables (méthode MovieClip.loadVariables)`, `loadMovie (méthode MovieClip.loadMovie)`, `onPress (gestionnaire MovieClip.onPress)`, `MovieClipLoader`, `onClipEvent, gestionnaire`, `constantes`, `loadMovieNum, fonction`, `unloadMovie, fonction`, `unloadMovieNum, fonction`

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 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.

La méthode `loadVariables` permet également de mettre à jour les variables du clip actif en fonction des 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.someDomain.com peut charger des données provenant uniquement de sources situées également à l'adresse www.someDomain.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 dans 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

Flash Lite 2.0

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, `url` doit appartenir au même domaine que le fichier SWF. Pour plus de détails, consultez la section « Description », ci-dessous.

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 est utilisée pour des 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());
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"
```

Voir aussi

loadMovie (méthode MovieClip.loadMovie), loadVariablesNum, fonction, 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 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 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 clips. 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

Flash Lite 2.0

Paramètres

pt : Object - Nom ou identifiant d'un objet créé avec la classe Object et qui spécifie 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();
};
```

Voir aussi

`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. Par défaut, la propriété `_lockroot` a pour valeur `undefined`. 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 (et non pas à `_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 `loadMovie()`, définissez la propriété `MovieClip._lockroot` sur `true` dans l'animation de chargeur, comme le montre le 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
Flash Lite 2.0

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 FLA, nolockroot, place le code suivant sur 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 purge 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 aussi

_root, propriété, _lockroot (propriété MovieClip._lockroot), attachMovie (méthode MovieClip.attachMovie) loadMovie (méthode MovieClip.loadMovie), onLoadInit (écouteur d'événement MovieClipLoader.onLoadInit)

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
Flash Lite 2.0

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 aussi
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
Flash Lite 2.0

Voir aussi
_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
Flash Lite 2.0

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 aussi

nextFrame, fonction, prevFrame, fonction, prevFrame (méthode MovieClip.prevFrame)

onData (gestionnaire MovieClip.onData)

```
onData = function() {}
```

Appelé lorsqu'un clip reçoit des données provenant d'un appel MovieClip.loadVariables() ou MovieClip.loadMovie(). Vous devez définir une fonction qui s'exécute lorsque le gestionnaire d'événements est appelé. Vous pouvez définir cette fonction sur le scénario ou dans un fichier de classe qui étend la classe MovieClip ou 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 devez appeler un gestionnaire d'événements lorsqu'un clip reçoit des données, vous devez utiliser onClipEvent() à la place de ce gestionnaire. Ce dernier gestionnaire est appelé lorsqu'un clip reçoit des données.

Disponibilité

Flash Lite 2.0

Exemple

L'exemple suivant illustre l'utilisation correcte de `MovieClip.onData()` et `onClipEvent(data)`.

Le clip symbol_mc est un symbole dans la bibliothèque. Il est lié à la classe MovieClip. La première fonction ci-dessous est déclenchée pour chaque occurrence de symbol_mc lorsqu'elle reçoit des données.

Le clip dynamic_mc est chargé avec MovieClip.loadMovie(). Le code utilisant dynamic_mc ci-dessous tente d'appeler une fonction pendant le chargement du clip, mais ne fonctionne pas. Le fichier SWF chargé doit être un symbole associé à la classe MovieClip dans la bibliothèque.

La dernière fonction utilise onClipEvent(data). Le gestionnaire d'événements onClipEvent() est appelé pour tout clip recevant des données, que le clip soit dans la bibliothèque ou non. Par conséquent, la dernière fonction de cet exemple est appelée lorsqu'une occurrence de symbol_mc est créée et lorsque replacement.swf est chargé.

```
// The following function is triggered for each instance of symbol_mc
// when it receives data.
symbol_mc.onData = function() {
    trace("The movie clip has received data");
}

// This code attempts to call a function when the clip is loaded,
// but it will not work, because the loaded SWF is not a symbol
// in the library associated with the MovieClip class.
function output()
{
    trace("Will never be called.");
}
dynamic_mc.onData = output;
dynamic_mc.loadMovie("replacement.swf");
// The following function is invoked for any movie clip that
// receives data, whether it is in the library or not.
onClipEvent( data ) {
    trace("The movie clip has received data");
}
```

Voir aussi

onClipEvent, gestionnaire

onDragOut (gestionnaire MovieClip.onDragOut)

```
onDragOut = function() {}
```

Appelé lorsque l'utilisateur appuie sur le bouton de la souris et si 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 cette fonction sur le scénario ou dans un fichier de classe qui étend la classe MovieClip ou lié à un symbole dans la bibliothèque.

Remarque : Ce gestionnaire d'événement n'est pris en charge par Flash Lite que si `System.capabilities.hasMouse` a pour valeur `true` ou si `System.capabilities.hasStylus` a pour valeur `true`.

Disponibilité

Flash Lite 2.0

Exemple

L'exemple suivant définit une fonction pour la méthode `onDragOut` qui transmet une instruction `trace()` au panneau Sortie :

```
my_mc.onDragOut = function () {
    trace ("onDragOut called");
}
```

Voir aussi

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 cette fonction sur le scénario ou dans un fichier de classe qui étend la classe MovieClip ou lié à un symbole dans la bibliothèque.

Remarque : Ce gestionnaire d'événement n'est pris en charge par Flash Lite que si `System.capabilities.hasMouse` a pour valeur `true` ou si `System.capabilities.hasStylus` a pour valeur `true`.

Disponibilité
Flash Lite 2.0

Exemple
L'exemple suivant définit une fonction pour la méthode `onDragOver` qui transmet une instruction `trace()` au panneau Sortie :

```
my_mc.onDragOver = function () {
    trace ("onDragOver called");
}
```

Voir aussi
`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
Flash Lite 2.0

Exemple
L'exemple suivant définit une fonction pour le gestionnaire d'événements `onEnterFrame` qui transmet une instruction `trace()` au panneau 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()` 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. Vous pouvez définir cette fonction sur le scénario ou dans un fichier de classe qui étend la classe MovieClip ou 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 `Selection.setFocus()` est utilisé, le chemin du clip doit être transmis à `Selection.setFocus()`. Les autres éléments peuvent aisément reprendre le focus lorsque l'utilisateur déplace la souris.

Disponibilité

Flash Lite 2.0

Exemple

L'exemple suivant définit une fonction pour la méthode `onKeyDown()` qui transmet une instruction `trace()` au panneau 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 vous sélectionnez le clip avec la touche de tabulation et appuyez sur une touche, `key was pressed` s'affiche dans le panneau 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 aussi

`getAscii (méthode Key.getAscii)`, `getCode (méthode Key.getCode)`, `focusEnabled (propriété MovieClip.focusEnabled)` `setFocus (méthode Selection.setFocus)`, `onKeyDown (écouteur d'événement Key.onKeyDown)` `onKeyUp (gestionnaire MovieClip.onKeyUp)`

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()` 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. Vous pouvez définir cette fonction sur le scénario ou dans un fichier de classe qui étend la classe MovieClip ou 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 `Selection.setFocus()` est utilisé, le chemin du clip doit être transmis à `Selection.setFocus()`. Les autres éléments peuvent aisément reprendre le focus lorsque l'utilisateur déplace la souris.

Disponibilité

Flash Lite 2.0

Exemple

L'exemple suivant définit une fonction pour la méthode `onKeyUp` qui transmet une instruction `trace()` au panneau 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 aussi

getAscii (méthode Key.getAscii), getCode (méthode Key.getCode), focusEnabled (propriété MovieClip.focusEnabled) setFocus (méthode Selection.setFocus), onKeyDown (écouteur d'événement Key.onKeyDown) onKeyDown (gestionnaire MovieClip.onKeyDown)

onKillFocus (gestionnaire MovieClip.onKillFocus)

```
onKillFocus = function(newFocus:Object) {}
```

Appelé lorsqu'un clip perd le focus d'entrée. La méthode `onKillFocus` reçoit un paramètre `newFocus` 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 cette fonction sur le scénario ou dans un fichier de classe qui étend la classe MovieClip ou lié à un symbole dans la bibliothèque.

Disponibilité

Flash Lite 2.0

Paramètres

newFocus : Object - Objet recevant le focus d'entrée.

Exemple

L'exemple suivant affiche 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);
};
```

Lorsque vous appuyez sur la touche de tabulation pour basculer entre deux instances, les informations s'affichent dans le panneau Sortie.

Voir aussi

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 cette fonction sur le scénario ou dans un fichier de classe qui étend la classe MovieClip ou 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 `MovieClip.loadMovie()` pour charger un fichier SWF de manière dynamique, vous devez utiliser la classe onClipEvent(load) ou MovieClipLoader à la place de ce gestionnaire. Contrairement à `MovieClip.onLoad`, les autres gestionnaires sont appelés lors du chargement d'un clip.

Disponibilité

Flash Lite 2.0

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 le fichier Oval.as et définissez une méthode de classe appelée `onLoad()` et assurez-vous que le chemin du fichier de classe est correct :

```
// 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 « Exporter pour ActionScript » et complétez les champs « Identificateur » et « Classe ActionScript 2.0 » avec le mot « Oval » (sans guillemets). Assurez-vous que « Exporter dans la première image » reste activé et 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 called ».

Voir aussi

loadMovie (méthode MovieClip.loadMovie), onClipEvent, gestionnaire, 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 cette fonction sur le scénario ou dans un fichier de classe qui étend la classe MovieClip ou lié à un symbole dans la bibliothèque.

Remarque : Ce gestionnaire d'événement n'est pris en charge par Flash Lite que si `System.capabilities.hasMouse` a pour valeur `true` ou si `System.capabilities.hasStylus` a pour valeur `true`.

Disponibilité

Flash Lite 2.0

Exemple
L'exemple suivant définit une fonction pour la méthode `onMouseDown` qui transmet une instruction `trace()` au panneau 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 cette fonction sur le scénario ou dans un fichier de classe qui étend la classe MovieClip ou lié à un symbole dans la bibliothèque.

Remarque : Ce gestionnaire d'événements est pris en charge dans Flash Lite seulement si `System.capabilities.hasMouse` a pour valeur `true`.

Disponibilité
Flash Lite 2.0

Exemple
L'exemple suivant définit une fonction pour la méthode `onMouseMove` qui transmet une instruction `trace()` au panneau 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 cette fonction sur le scénario ou dans un fichier de classe qui étend la classe MovieClip ou lié à un symbole dans la bibliothèque.

Remarque : Ce gestionnaire d'événement n'est pris en charge par Flash Lite que si `System.capabilities.hasMouse` a pour valeur `true` ou si `System.capabilities.hasStylus` a pour valeur `true`.

Disponibilité
Flash Lite 2.0

Exemple
L'exemple suivant définit une fonction pour la méthode `onMouseUp` qui transmet une instruction `trace()` au panneau 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
Flash Lite 2.0

Exemple
L'exemple suivant définit une fonction pour la méthode `onPress` qui transmet une instruction `trace()` au panneau Sortie :

```
my_mc.onPress = function () {
    trace ("onPress called");
}
```

onRelease (gestionnaire MovieClip.onRelease)

```
onRelease = function() {}
```

Appelé lorsque l'utilisateur relâche le bouton de la souris au-dessus d'un clip. Vous devez définir une fonction qui s'exécute lorsque le gestionnaire d'événements est appelé. Vous pouvez définir cette fonction sur le scénario ou dans un fichier de classe qui étend la classe MovieClip ou lié à un symbole dans la bibliothèque.

Disponibilité
Flash Lite 2.0

Exemple
L'exemple suivant définit une fonction pour la méthode `onRelease` qui transmet une instruction `trace()` au panneau Sortie :

```
my_mc.onRelease = function () {
    trace ("onRelease called");
}
```

onReleaseOutside (gestionnaire MovieClip.onReleaseOutside)

```
onReleaseOutside = function() {}
```

Appelé lorsque l'utilisateur a appuyé sur le bouton de la souris dans la zone occupée par un clip, puis l'a relâché en dehors de cette zone.

Vous devez définir une fonction qui s'exécute lorsque le gestionnaire d'événements est appelé. Vous pouvez définir cette fonction sur le scénario ou dans un fichier de classe qui étend la classe MovieClip ou lié à un symbole dans la bibliothèque.

Remarque : Ce gestionnaire d'événement n'est pris en charge par Flash Lite que si `System.capabilities.hasMouse` a pour valeur `true` ou si `System.capabilities.hasStylus` a pour valeur `true`.

Disponibilité
Flash Lite 2.0

Exemple
L'exemple suivant définit une fonction pour la méthode `onReleaseOutside` qui transmet une instruction `trace()` au panneau Sortie :

```
my_mc.onReleaseOutside = function () {
    trace ("onReleaseOutside called");
}
```

onRollOut (gestionnaire MovieClip.onRollOut)

```
onRollOut = function() {}
```

Appelé lorsque le pointeur se déplace hors de la zone du clip.

Vous devez définir une fonction qui s'exécute lorsque le gestionnaire d'événements est appelé. Vous pouvez définir cette fonction sur le scénario ou dans un fichier de classe qui étend la classe MovieClip ou lié à un symbole dans la bibliothèque.

Disponibilité
Flash Lite 2.0

Exemple
L'exemple suivant définit une fonction pour la méthode `onRollOut` qui transmet une instruction `trace()` au panneau Sortie :

```
my_mc.onRollOut = function () {
    trace ("onRollOut called");
}
```

onRollOver (gestionnaire MovieClip.onRollOver)

```
onRollOver = function() {}
```

Appelé lorsque le pointeur se déplace au-dessus de la zone du clip.

Vous devez définir une fonction qui s'exécute lorsque le gestionnaire d'événements est appelé. Vous pouvez définir cette fonction sur le scénario ou dans un fichier de classe qui étend la classe MovieClip ou lié à un symbole dans la bibliothèque.

Disponibilité
Flash Lite 2.0

Exemple
L'exemple suivant définit une fonction pour la méthode `onRollOver` qui transmet une instruction `trace()` au panneau Sortie :

```
my_mc.onRollOver = function () {
    trace ("onRollOver called");
}
```

onSetFocus (gestionnaire MovieClip.onSetFocus)

```
onSetFocus = function(oldFocus:Object) {}
```

Appelé lorsqu'un clip reçoit le focus d'entrée. Le paramètre `oldFocus` est l'objet qui perd le focus. Par exemple, si l'utilisateur appuie sur la touche de tabulation pour déplacer le focus d'entrée d'un clip vers un champ texte, `oldFocus` contient l'occurrence de clip.

Si aucun objet n'avait précédemment reçu le focus, le paramètre `oldFocus` contient une valeur null.

Vous devez définir une fonction qui s'exécute lorsque le gestionnaire d'événements est appelé. Vous pouvez définir cette fonction sur le scénario ou dans un fichier de classe qui étend la classe MovieClip ou lié à un symbole dans la bibliothèque.

Disponibilité

Flash Lite 2.0

Paramètres

oldFocus : Object - Objet perdant le focus.

Exemple

L'exemple suivant affiche des informations sur le clip qui reçoit le focus d'entrée 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);
}
```

Lorsque vous appuyez sur la touche de tabulation entre deux occurrences, les informations s'affichent dans le panneau Sortie.

Voir aussi

`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 cette fonction sur le scénario ou dans un fichier de classe qui étend la classe MovieClip ou lié à un symbole dans la bibliothèque.

Disponibilité

Flash Lite 2.0

Exemple

L'exemple suivant définit une fonction pour la méthode `MovieClip.onUnload` qui transmet une instruction `trace()` au panneau Sortie :

```
my_mc.onUnload = function () {
    trace ("onUnload called");
}
```

_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

Flash Lite 2.0

Exemple

L'exemple ci-dessous suit la référence à un clip et sa relation au scénario principal. 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 Sortie :

```
You clicked the movie clip: _level0.my_mc
The parent of my_mc is: _level0
```

Voir aussi

_parent (propriété Button._parent), _root, propriété, targetPath, fonction, _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

Flash Lite 2.0

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 aussi

play, fonction, gotoAndPlay (méthode MovieClip.gotoAndPlay), gotoAndPlay, fonction

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

Flash Lite 2.0

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);
    }
};
```

Voir aussi

prevFrame, fonction

_quality (MovieClip._quality, propriété)

```
public _quality : String
```

Définit ou extrait la qualité du rendu appliqué à 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 :

Valeur	Description	Anti-aliasing des graphiques
"LOW"	Qualité de rendu inférieure.	Les graphiques ne sont pas anti-aliasés.
"MEDIUM"	Qualité de rendu moyenne. Ce niveau de qualité convient aux animations qui ne contiennent pas de texte.	Les graphiques sont anti-aliasés en utilisant une grille de 2 x 2 pixels.
"HIGH"	Qualité de rendu supérieure. Il s'agit du paramètre de qualité de rendu par défaut de Flash.	Les graphiques sont anti-aliasés en utilisant une grille de 4 x 4 pixels.
"BEST"	Très haute qualité de rendu.	Les graphiques sont anti-aliasés en utilisant une grille de 4 x 4 pixels.

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
Flash Lite 2.0

Exemple
Cet exemple définit la qualité de rendu d'un clip appelé `my_mc` sur `LOW` :

```
my_mc._quality = "LOW";
```

Voir aussi
`_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 `MovieClip.swapDepths()` pour convertir cette valeur en valeur positive.

Vous pouvez étendre les méthodes et les gestionnaires d'événements de la classe MovieClip en créant une sous-classe.

Disponibilité
Flash Lite 2.0

Exemple
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 la supprimez 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 aussi

duplicateMovieClip, fonction, 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 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_mc._rotation = 450` est identique à `my_mc._rotation = 90`.

Disponibilité

Flash Lite 2.0

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;
};
```

Voir aussi

_rotation (propriété Button._rotation), _rotation (propriété TextField._rotation)

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 et cette action est irréversible. Par exemple, vous pouvez disposer d'un clip dans d'un calque de masque appelé `UIMask` qui dissimule 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
Flash Lite 2.0

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 tous les sons 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
Flash Lite 2.0

Voir aussi
`_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.

Remarque : Cette méthode n'est prise en charge par Flash Lite que si `System.capabilities.hasMouse` est défini sur `true` ou si `System.capabilities.hasStylus` est défini sur `true`.

Disponibilité

Flash Lite 2.0

Paramètres

lockCenter : Boolean [facultatif] - Valeur booléenne spécifiant si le clip à déplacer est verrouillé au centre de la position de la souris (`true`) ou 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 spécifient le rectangle de délimitation du clip.

haut : Number [facultatif] - Valeur relative aux coordonnées du parent du clip, qui spécifient le rectangle de délimitation du clip.

droite : Number [facultatif] - Valeur relative aux coordonnées du parent du clip, qui spécifient le rectangle de délimitation du clip.

bas : Number [facultatif] - Valeur relative aux coordonnées du parent du clip, qui spécifient le rectangle de délimitation du clip.

Exemple

L'exemple suivant crée une occurrence de clip déplaçable 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 aussi

_droptarget (MovieClip._droptarget, propriété), startDrag, fonction, 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

Flash Lite 2.0

Exemple

L'exemple suivant indique comment arrêter un clip appelé `aMovieClip` :

```
aMovieClip.stop();
```

Voir aussi

stop, fonction

stopDrag (méthode MovieClip.stopDrag)

```
public stopDrag() : Void
```

Arrête une méthode `MovieClip.startDrag()`. Un clip rendu déplaçable via cette méthode le reste 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.

Remarque : Cette méthode n'est prise en charge par Flash Lite que si `System.capabilities.hasMouse` est défini sur `true` ou si `System.capabilities.hasStylus` est défini sur `true`.

Disponibilité

Flash Lite 2.0

Exemple

L'exemple suivant crée une occurrence de clip déplaçable 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 aussi

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

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 par 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 un 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

Flash Lite 2.0

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 puis ajoutez le script suivant au scénario parent :

```
myMC1_mc.onRelease = function() {
    this.swapDepths(myMC2_mc);
};
myMC2_mc.onRelease = function() {
    this.swapDepths(myMC1_mc);
};
```

Voir aussi

_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 `tabChildren` est définie sur `false`, les enfants d'un clip ne sont pas inclus dans l'ordre de tabulation automatique. La valeur par défaut est `undefined`.

Disponibilité

Flash Lite 2.0

Exemple

Un widget de zone de liste sous forme de clip contient plusieurs éléments. 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, définissez la propriété `tabChildren` de la zone de liste sur `false`.

La propriété `tabChildren` n'a aucun effet si la propriété `tabIndex` est utilisée. La propriété `tabChildren` n'affecte que 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 aussi

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 elle définit au moins un gestionnaire de clip, comme `MovieClip.onRelease`. 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. Toutefois si `MovieClip.tabChildren` est définie sur `true`, les enfants du clip peuvent toujours être inclus dans l'ordre de tabulation automatique, même si `tabEnabled` est définie sur `false`.

Disponibilité

Flash Lite 2.0

Exemple

L'exemple suivant exclut `myMC2_mc` dans l'ordre de tabulation automatique :

```
myMC1_mc.onRelease = function() {};
myMC2_mc.onRelease = function() {};
myMC3_mc.onRelease = function() {};
myMC2_mc.tabEnabled = false;
```

Voir aussi

`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. Par défaut, la propriété `tabIndex` est définie sur `undefined`. Vous pouvez définir la propriété `tabIndex` sur un bouton, un clip ou une occurrence de champ texte.

Si un objet situé dans un fichier SWF contient une propriété `tabIndex`, l'ordre de tabulation automatique est désactivé et l'ordre de tabulation est calculé à partir des propriétés `tabIndex` des objets 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 dont la propriété `tabIndex` est définie sur `1` précède un objet dont la propriété `tabIndex` est définie sur `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'utilisez pas la même valeur de `tabIndex` pour plusieurs objets.

Disponibilité

Flash Lite 2.0

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 aussi

tabIndex (propriété Button.tabIndex), tabIndex (propriété TextField.tabIndex)

_target (propriété MovieClip._target)

```
public _target : String [read-only]
```

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

Flash Lite 2.0

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 : Number [read-only]
```

Renvoie le nombre total d'images dans l'occurrence de clip spécifiée par le paramètre `MovieClip`.

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 un événement de relâchement de la souris ou du stylet. Si vous faites glisser un stylet ou une souris sur un clip, puis que vous le ou la relâchez au-dessus d'un autre clip, l'événement `onRelease` est enregistré pour le deuxième clip. Ceci permet de créer des menus pour le deuxième clip. Vous pouvez définir la propriété `trackAsMenu` sur n'importe quel bouton ou objet clip. Si vous n'avez pas défini la propriété `trackAsMenu`, le comportement par défaut est `false`.

Vous pouvez modifier la propriété `trackAsMenu` à tout moment ; le clip modifié accepte immédiatement le nouveau comportement.

Remarque : Cette propriété n'est prise en charge par Flash Lite que si `System.capabilities.hasMouse` est définie sur `true` ou si `System.capabilities.hasStylus` est définie sur `true`.

Disponibilité

Flash Lite 2.0

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 aussi

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

Flash Lite 3.1

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;
}
```

Voir aussi

« Transform (flash.geom.Transform) » à la page 666

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

Flash Lite 2.0

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 aussi

removeMovieClip (méthode MovieClip.removeMovieClip), attachMovie (méthode MovieClip.attachMovie) loadMovie (méthode MovieClip.loadMovie), unloadMovie, fonction, unloadMovieNum, fonction

_url (propriété MovieClip._url)

```
public _url : String [read-only]
```

Récupère l'URL du fichier SWF, JPEG, GIF ou PNG ayant servi à télécharger le clip.

Disponibilité

Flash Lite 2.0

Exemple

L'exemple suivant affiche l'URL de l'image chargée dans l'occurrence `image_mc` du panneau 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.helpexamples.com/flash/images/image1.jpg", image_mc);
```

_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, l'utilisateur ne peut pas cliquer sur le bouton d'un clip dont la propriété `_visible` est définie sur `false`.

Disponibilité

Flash Lite 2.0

Exemple

L'exemple suivant définit la propriété `_visible` de deux clips appelés `myMC1_mc` et `myMC2_mc`. Cette propriété est définie sur `true` pour une occurrence et sur `false` pour l'autre. Notez que l'utilisateur ne peut plus cliquer sur l'occurrence `myMC1_mc` dès que 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 aussi

_visible (propriété Button._visible), _visible (propriété TextField._visible)

_width (propriété MovieClip._width)

```
public _width : Number
```

Largeur du clip, en pixels.

Disponibilité

Flash Lite 2.0

Exemple

L'exemple de code suivant affiche la hauteur et la largeur d'un clip dans le panneau 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");
```

Voir aussi

_height (propriété MovieClip._height)

_x (MovieClip._x, propriété)

```
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, il se trouve dans le système de coordonnées local de celui 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
Flash Lite 2.0

Voir aussi
_xscale (propriété MovieClip._xscale), _y (MovieClip._y, propriété), _yscale (propriété MovieClip._yscale)

_xmouse (propriété MovieClip._xmouse)

```
public _xmouse : Number [read-only]
```

Renvoie la coordonnée *x* de la position de la souris.

Remarque : Cette propriété n'est prise en charge par Flash Lite que si `System.capabilities.hasMouse` est définie sur `true` ou si `System.capabilities.hasStylus` est définie sur `true`.

Disponibilité
Flash Lite 2.0

Exemple
L'exemple suivant renvoie les coordonnées *x* et *y* actuelles de la souris sur la scène (`_level0`) et par rapport à un clip appelé `my_mc` 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>";
};
```

Voir aussi
hasMouse (propriété capabilities.hasMouse), _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 `_y` 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
Flash Lite 2.0

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 aussi

_x (MovieClip._x, propriété), _y (MovieClip._y, propriété), _yscale (propriété MovieClip._yscale), _width (propriété MovieClip._width)

_y (MovieClip._y, propriété)

```
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, il se trouve dans le système de coordonnées local de celui 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

Flash Lite 2.0

Voir aussi

_x (MovieClip._x, propriété), _xscale (propriété MovieClip._xscale), _yscale (propriété MovieClip._yscale)

_ymouse (propriété MovieClip._ymouse)

```
public _ymouse : Number [read-only]
```

Renvoie la coordonnée *y* de la position de la souris.

Remarque : Cette propriété n'est prise en charge par Flash Lite que si `System.capabilities.hasMouse` est définie sur `true` ou si `System.capabilities.hasStylus` est définie sur `true`.

Disponibilité

Flash Lite 2.0

Exemple

L'exemple suivant renvoie les coordonnées *x* et *y* actuelles de la souris sur la scène (`_level0`) et par rapport à un clip appelé `my_mc` 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>";
};
```

Voir aussi

hasMouse (propriété capabilities.hasMouse), _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

Flash Lite 2.0

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 aussi

_x (MovieClip._x, propriété), _xscale (propriété MovieClip._xscale), _y (MovieClip._y, propriété), _height (propriété MovieClip._height)

MovieClipLoader

```
Object
    |
    +-MovieClipLoader
```

```
public class MovieClipLoader
extends Object
```

La classe MovieClipLoader permet d'implémenter des rappels d'écouteur qui fournissent des informations d'état lors du chargement (téléchargement) de fichiers SWF, JPEG, GIF et PNG dans les clips. Pour exploiter les fonctionnalités de MovieClipLoader, téléchargez des ficheirs SWF en utilisant `MovieClipLoader.loadClip()` plutôt que `loadMovie()` ou `MovieClip.loadMovie()`.

Après avoir appelé la méthode `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 lors du processus de chargement.

- **Remarque :** Vous pouvez appeler `MovieClipLoader.getProgress()` à tout moment au cours du processus de chargement.
- Lorsque le fichier téléchargé est 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.

Après l'appel de `MovieClipLoader.onLoadInit`, vous pouvez définir des propriétés, utiliser des méthodes et interagir avec le clip chargé.

Si le chargement du fichier ne s'effectue pas entièrement, l'écouteur `MovieClipLoader.onLoadError` est appelé.

Disponibilité

Flash Lite 2.0

Résumé des propriétés

Propriétés héritées de la classe Object

constructor (propriété Object.constructor), __proto__ (Object.__proto__, propriété), prototype (propriété Object.prototype), __resolve (Object.__resolve, propriété)

Résumé des événements

Evénement	Description
onLoadComplete = function(listenerObject, [target_mc]) {}	Appelé lorsque le fichier chargé avec `MovieClipLoader.loadClip()` a fini son téléchargement.
onLoadError = function(target_mc, errorCode) {}	Appelé lorsque le chargement d'un fichier par `MovieClipLoader.loadClip()` a échoué.
onLoadInit = function([target_mc]) {}	Appelé une fois les actions de la première image du clip chargé exécutées.
onLoadProgress = function([target_mc], loadedBytes, totalBytes) {}	Appelé à chaque fois que du 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]) {}	Appelé lorsqu'un appel à `MovieClipLoader.loadClip()` a commencé avec succès à charger un fichier.

Récapitulatif des constructeurs

Signature	Description
MovieClipLoader()	Crée un objet MovieClipLoader que vous pouvez utiliser pour implémenter un certain nombre d'écouteurs 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 d'un fichier chargé à l'aide de MovieClipLoader.loadClip() ; pour les clips compressés, la méthode getProgress renvoie le nombre d'octets compressés.
	loadClip(url:String, target:Object) : Boolean	Charge un fichier SWF ou JPEG dans un clip Flash Lite 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

Flash Lite 2.0

Paramètres

listener : Object - Objet chargé de détecter la notification de rappel provenant des gestionnaires d'événements MovieClipLoader.

Valeur renvoyée

Boolean - Valeur booléenne. La valeur renvoyée est `true` lorsque l'écouteur est établi avec succès ; `false` dans tous les autres cas.

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);
```

Voir aussi

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 MoveClipLoader.getProgress)

```
public getProgress(target:Object) : Object
```

Renvoie le nombre d'octets chargés et le nombre total d'octets d'un fichier chargé à l'aide de `MovieClipLoader.loadClip()` ; pour les clips compressés, la méthode `getProgress` 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

Flash Lite 2.0

Paramètres

target : Object - Fichier SWF, JPEG, GIF ou PNG chargé par `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, la méthode la plus courante consiste à créer un objet d'écoute pour l'événement `onLoadProgress`. Une autre note importante sur cette méthode est que le premier appel synchrone à getProgress permet de renvoyer les paramètres bytesLoaded et bytesTotal du *conteneur*, et non pas les valeurs de l'objet demandé en externe.

```
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);
    }
}
```

Voir aussi

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 ou JPEG dans un clip Flash Lite Player lors de la lecture de l'animation d'origine. Cette méthode 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()` en lieu et place de `loadMovie()` ou de `MovieClip.loadMovie()` offre plusieurs avantages. Les gestionnaires suivants sont implémentés par l'intermédiaire d'un objet écouteur. Vous activez l'écouteur en utilisant `MovieClipLoader.addListener(listenerObject)` pour l'enregistrer dans la classe MovieClipLoader.

- Le gestionnaire `MovieClipLoader.onLoadStart` est appelé au début du chargement.
- Le gestionnaire `MovieClipLoader.onLoadError` est appelé si le clip ne peut pas être chargé.
- Le gestionnaire `MovieClipLoader.onLoadProgress` est appelé pendant le 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 un niveau unique ; les objets écouteurs MovieClipLoader sont transmis à l'occurrence de clip en cours de chargement sous forme de paramètre. 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` n'indiquent pas les valeurs réelles de `bytesLoaded` et `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.

Disponibilité

Flash Lite 2.0

Paramètres

url : String - 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:///. 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 Lite 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. La valeur renvoyée est `true` lorsque la demande d'URL est envoyée avec succès ; `false` dans tous les autres cas.

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.

Créez une méthode de gestionnaire pour l'événement `onLoadInit`.

```
public function onLoadInit(mc:MovieClip):Void {
    trace("onLoadInit: " + mc);
}
```

Créez un clip vide et chargez une image dedans à l'aide de MovieClipLoader.

```
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);
}
```

Voir aussi

onLoadInit (écouteur d'événement MovieClipLoader.onLoadInit)

Constructeur MovieClipLoader

```
public MovieClipLoader()
```

Crée un objet MovieClipLoader que vous pouvez utiliser pour implémenter un certain nombre d'écouteurs chargés de répondre aux événements pendant le téléchargement d'un fichier SWF, JPEG, GIF ou PNG.

Disponibilité

Flash Lite 2.0

Exemple

Consultez la section `MovieClipLoader.loadClip()`.

Voir aussi

addListener (méthode MovieClipLoader.addListener), loadClip (méthode MovieClipLoader.loadClip)

onLoadComplete (écouteur d'événement MovieClipLoader.onLoadComplete)

```
onLoadComplete = function(listenerObject, [target_mc]) {}
```

Appelé lorsque le fichier chargé avec `MovieClipLoader.loadClip()` a fini son téléchargement. La valeur de `target_mc` identifie le clip pour lequel cet appel est effectué. Ceci est particulièrement utile si plusieurs fichiers sont chargés avec le même jeu d'écouteurs.

Ce paramètre est transmis par Flash à votre code, mais il n'est pas nécessaire d'implémenter tous les paramètres dans la fonction d'écouteur.

Lorsque vous utilisez les événements `onLoadComplete` et `onLoadInit` avec la classe MovieClipLoader, il est important de comprendre les différences par rapport à l'utilisation d'un fichier SWF. L'événement `onLoadComplete` est appelé lorsque le chargement du fichier SWF ou JPEG est terminé, mais avant l'initialisation de l'application. A ce stade, vous ne pouvez pas accéder aux méthodes et aux propriétés du clip qui a été chargé, ce qui implique que vous ne pouvez pas appeler de fonctions, passer à 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

Flash Lite 2.0

Paramètres

listenerObject : `- Objet écouteur ajouté à l'aide de MovieClipLoader.addListener()`.

target_mc : [facultatif] - Clip chargé par une méthode `MovieClipLoader.loadClip()`. Ce paramètre est facultatif.

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 un champ texte créé de façon dynamique et 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.macromedia.com/images/shared/product_boxes/112x112/box_studio
_112x112.jpg", image_mc);
```

Voir aussi

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, errorCode) {}
```

Appelé lorsque le chargement d'un fichier par `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.

Appelez 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 pas l'écouteur `MovieClipLoader.onLoadComplete` ; par exemple si le téléchargement a été interrompu en raison d'un encombrement du serveur, d'une panne de serveur, etc.

Disponibilité

Flash Lite 2.0

Paramètres

target_mc : - Clip chargé par une méthode `MovieClipLoader.loadClip()`.

errorCode : - Chaîne qui précise les raisons de l'échec.

Exemple

L'exemple suivant affiche des informations dans le panneau Sortie lorsqu'une image ne se charge pas.

```
this.createEmptyMovieClip("image_mc", this.getNextHighestDepth());
var mclListener:Object = new Object();
mclListener.onLoadError = function(target_mc:MovieClip, errorCode:String) {
    trace("ERROR!");
    switch (errorCode) {
    case 'URLNotFound' :
        trace("\t Unable to connect to URL: "+target_mc._url);
        break;
    case 'LoadNeverCompleted' :
        trace("\t Unable to complete download: "+target_mc);
        break;
    }
};
mclListener.onLoadInit = function(target_mc:MovieClip) {
    trace("success");
    trace(image_mcl.getProgress(target_mc).bytesTotal+" bytes loaded");
};
var image_mcl:MovieClipLoader = new MovieClipLoader();
image_mcl.addListener(mclListener);
image_mcl.loadClip("http://www.fakedomain.com/images/bad_hair_day.jpg", image_mc);
```

Voir aussi

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]) {}
```

Appelé une fois les actions de la première image du clip chargé exécutées. Lorsque cet écouteur est appelé, vous pouvez définir les propriétés, utiliser les méthodes ou encore interagir avec l'animation chargée. Appelez 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.

Disponibilité

Flash Lite 2.0

Paramètres

target_mc : [facultatif] - Clip chargé par une 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. Ces 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 :

```
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);
```

Voir aussi

addListener (méthode MovieClipLoader.addListener), loadClip (méthode MovieClipLoader.loadClip), onLoadStart (écouteur d'événement MovieClipLoader.onLoadStart)

onLoadProgress (écouteur d'événement MovieClipLoader.onLoadProgress)

```
onLoadProgress = function([target_mc], loadedBytes, totalBytes) {}
```

Appelé à chaque fois que du contenu est écrit sur le disque dur au cours du processus de chargement (c'est-à-dire, entre `MovieClipLoader.onLoadStart` et `MovieClipLoader.onLoadComplete`). Appelez 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 de `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 d'animation sur un fichier local résidant sur votre disque dur, il ne fonctionne pas correctement car, en mode test d'animation, Flash Lite Player charge intégralement les fichiers locaux.

Paramètres

`target_mc: MovieClip [facultatif]` - Clip chargé par une 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.

Disponibilité
Flash Lite 2.0

Paramètres
target_mc : [facultatif] - Clip chargé par une méthode `MovieClipLoader.loadClip()`.

loadedBytes : - Nombre d'octets chargés lorsque l'écouteur a été appelé.

totalBytes : - Nombre total d'octets dans le fichier chargé.

Exemple
L'exemple suivant crée un clip, une nouvelle classe `MovieClipLoader` et un écouteur d'événement anonyme. Il doit afficher 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);
```

Voir aussi
`addListener (méthode MovieClipLoader.addListener)`, `loadClip (méthode MovieClipLoader.loadClip)`, `getProgress (méthode MoveClipLoader.getProgress)`

onLoadStart (écouteur d'événement MovieClipLoader.onLoadStart)

```
onLoadStart = function([target_mc]) {}
```

Appelé lorsqu'un appel à `MovieClipLoader.loadClip()` a commencé avec succès à charger un fichier. Appelez 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.

Disponibilité
Flash Lite 2.0

Paramètres
target_mc : [facultatif] - Clip chargé par une 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. Ces 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);
```

Voir aussi

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

Flash Lite 2.0

Paramètres

listener : Object - Objet écouteur ajouté à l'aide de `MovieClipLoader.addListener()`.

Valeur renvoyée

Boolean -

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 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;
```

Voir aussi

addListener (méthode MovieClipLoader.addListener)

unloadClip (méthode MovieClipLoader.unloadClip)

```
public unloadClip(target:Object) : Boolean
```

Supprime un clip chargé via `MovieClipLoader.loadClip()`. Si vous appelez cette méthode lors du chargement d'une animation, `MovieClipLoader.onLoadError` est appelé.

Disponibilité

Flash Lite 2.0

Paramètres

target : Object - Chaîne ou entier transmis à l'appel correspondant à `my_mcl.loadClip()`.

Valeur renvoyée

Boolean -

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 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);
```

Voir aussi

loadClip (méthode MovieClipLoader.loadClip), onLoadError (écouteur d'événement MovieClipLoader.onLoadError)

NetConnection

Crée un objet NetConnection que vous pouvez utiliser avec un objet NetStream pour appeler des commandes sur un serveur d'applications distant ou pour lire des fichiers vidéo en flux continu (FLV) soit localement, soit à partir d'un serveur.

Méthode	Description
`connect();`	`connect(command:String, ... arguments):void` Ouvre une connexion à un serveur.
`close();`	`close():void` Ferme la connexion ouverte localement ou avec le serveur et distribue l'événement netStatus avec une propriété de code de `NetConnection.Connect.Closed`.

Disponibilité

Flash Lite 3.0

connect (méthode NetConnection.connect)

Ouvre une connexion à un serveur. Grâce à cette connexion, vous pouvez lire des fichiers audio ou vidéo (FLV) à partir du système de fichiers local ou appeler des commandes sur un serveur distant.

Lorsque vous utilisez cette méthode, tenez compte du modèle de sécurité de Flash Lite Player et des considérations de sécurité suivantes :

- Par défaut, le site Web refuse l'accès entre des Sandbox. Le site Web peut autoriser l'accès à une ressource en utilisant un fichier de régulation inter-domaines.

- Un site peut refuser l'accès à une ressource en ajoutant une logique d'application ActionScript côté serveur dans Flash Media Server.
- Vous ne pouvez pas utiliser la méthode `NetConnection.connect()` si le fichier SWF appelant est dans le sandbox local avec système de fichiers.
- Pour empêcher un fichier SWF d'utiliser cette méthode, définissez le paramètre `allowNetworking` des balises object et embed dans la page HTML hébergeant le contenu SWF.

Disponibilité
Flash Lite 3.0

close (méthode NetConnection.close)

```
public function close():void
```

Ferme la connexion ouverte localement ou avec le serveur et distribue l'événement `netStatus` avec une propriété de code de `NetConnection.Connect.Closed`.

Cette méthode déconnecte tous les objets NetStream exécutés sur cette connexion. Toutes les données mises en attente et qui n'ont pas été envoyées sont supprimées. (Pour arrêter les flux sans fermer la connexion, utilisez la méthode `NetStream.close()`.) Si vous appelez cette méthode et souhaitez vous reconnecter, vous devez recréer l'objet NetStream.

Disponibilité
Flash Lite 3.0

Voir aussi
NetStream

NetConnection, constructeur

```
public « NetConnection » à la page 482()
```

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 la méthode `Connect (NetConnection.connect)` pour établir la véritable connexion.

La lecture de fichiers FLV externes offre plusieurs avantages par rapport à l'intégration de vidéo dans un document Flash : amélioration des performances et de la gestion de la mémoire, indépendance par rapport aux 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

Flash Lite 3.0

Exemple
Consultez l'exemple de méthode `connect (NetConnection.connect)`.

Voir aussi
Méthode « connect (méthode NetConnection.connect) » à la page 482, « attachVideo (méthode Video.attachVideo) » à la page 675,« NetStream » à la page 484

NetStream

Crée un flux de diffusion en continu qui permet de lire des fichiers FLV à l'aide de l'objet NetConnection spécifié.

Les URL de fichiers locaux sont également prises en charge par un simple remplacement de « http: » par « file: ». Voici un exemple d'utilisation :

```
NetStream.play("http://somefile.flv");
NetStream.play("file://somefile.flv");
```

Remarque : *Des restrictions de sécurité standard s'appliquent. Par exemple, un fichier SWF distant ne peut pas accéder aux URL file:// absolues sous la forme « file://C:/unfichier.flv ».*

Disponibilité

Flash Lite 3.0

Méthodes de la classe NetStream

Méthode	Description
`close()`	`close():void` Arrête la lecture des données du flux de diffusion en continu, définit la propriété time sur 0 et met le flux de diffusion en continu à disposition.
`pause()`	`pause():void` Suspend la lecture d'un flux vidéo.
`play()`	`play(... arguments):void` Lance la lecture d'un fichier audio ou vidéo externe (FLV).
`seek()`	Recherche l'image-clé la plus proche du nombre de secondes spécifié à partir du début du flux.
`setBufferTime()`	Spécifie la durée de la mise en mémoire tampon des messages avant le début de l'affichage.

Propriétés de la classe NetStream

Propriétés	Description
`bufferLength`	Le nombre de secondes de données enregistrées dans la mémoire tampon.
`bufferTime`	Le nombre de secondes affectées à la mémoire tampon par `setBufferTime()`.
`bytesLoaded`	Le nombre d'octets de données ayant été chargés dans le lecteur.
`bytesTotal`	La taille totale, en octets, du fichier chargé dans le lecteur.
`currentFPS`	Le nombre d'images affichées par seconde.
`time`	La position de la tête de lecture, en secondes.

Evénements de la classe NetStream

Evénement	Description
`onStatus`	Appelé à chaque changement d'état ou à chaque fois qu'une erreur est publiée pour l'objet NetStream.
`onCuePoint`	Appelé lorsqu'un point de repère intégré est atteint lors de la lecture d'un fichier FLV.
`OnMetaData`	Appelé lorsque Flash Lite Player reçoit des informations descriptives intégrées dans le fichier FLV.

bufferLength (propriété NetStream.bufferLength)

```
public bufferLength : Number [read-only]
```

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

Flash Lite 3.0

Remarque : *Cette propriété est également prise en charge dans Flash Player 6 lorsqu'elle est utilisée avec Flash Media Server. Pour plus d'informations, consultez la documentation de Flash Media Server.*

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;
}
```

bufferTime (propriété NetStream.bufferTime)

```
public bufferTime : Number [read-only]
```

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
Flash Lite 3.0

Remarque : *Cette propriété est également prise en charge dans Flash Player 6 lorsqu'elle est utilisée avec Flash Media Server. Pour plus d'informations, consultez la documentation de Flash Media Server.*

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;
}
```

bytesLoaded (propriété NetStream.bytesLoaded)

```
public bytesLoaded : Number [read-only]
```

Le nombre d'octets de données ayant été chargés dans le lecteur. Vous pouvez utiliser cette méthode 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
Flash Lite 3.0

Exemple
L'exemple suivant crée une barre de progression avec l'API de dessin, ainsi que les propriétés `bytesLoaded` et `bytesTotal`. La barre affiche la progression du chargement de video1.flv dans l'occurrence d'objet vidéo appelée 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);
       }
}
```

bytesTotal (propriété NetStream.bytesTotal)

```
public bytesTotal : Number [read-only]
```

La taille totale, en octets, du fichier chargé dans le lecteur.

Disponibilité

Flash Lite 3.0

Exemple

L'exemple suivant crée une barre de progression avec l'API de dessin, ainsi que les propriétés `bytesLoaded` et `bytesTotal`. La barre affiche la progression du chargement de video1.flv dans l'occurrence d'objet vidéo appelée 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);
       }
}
```

currentFps (propriété NetStream.currentFps)

```
public currentFps : Number [read-only]
```

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 déterminer la compression à appliquer lors de l'exportation du fichier.

Disponibilité

Flash Lite 3.0

Remarque : *Cette propriété est également prise en charge dans Flash Player 6 lorsqu'elle est utilisée avec Flash Media Server. Pour plus d'informations, consultez la documentation de Flash Media Server.*

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);
}
time (NetStream.time property)
```

NetStream, constructeur

```
public NetStream(connection:« NetConnection » à la page 482)
```

Crée un flux de diffusion en continu qui permet de lire des fichiers FLV à l'aide de l'objet NetConnection spécifié.

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 de Flash Communication Server.*

Paramètres

connection : `NetConnection` - Un 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 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 aussi

« NetConnection » à la page 482, « attachVideo (méthode Video.attachVideo) » à la page 675

time (propriété NetStream.time)

```
public time : Number [read-only]
```

La position de la tête de lecture, en secondes.

Disponibilité

Flash Lite 3.0

Remarque : *Cette propriété est également prise en charge dans Flash Player 6 lorsqu'elle est utilisée avec Flash Media Server. Pour plus d'informations, consultez la documentation de Flash Media Server.*

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 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;
}
```

onStatus (gestionnaire NetStream.onStatus)

```
onStatus = function(infoObject:Object) {}
```

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'information a une propriété de code contenant une chaîne qui décrit le résultat du gestionnaire `onStatus` et une propriété `level` contenant la chaîne `Status` ou `Error`.

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 continu 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.

Propriété du code	Propriété de niveau	Signification
`NetStream.Play.Start`	état	La lecture a repris.
`NetStream.Play.Stop`	état	La lecture s'est arrêtée.
`NetStream.Play.StreamNotFound`	état	Le fichier FLV transmis à la méthode `play()` est introuvable.
`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 `Error.message.details` contient un code de temps qui indique la dernière position valide à laquelle l'utilisateur peut effectuer une recherche. Reportez-vous à la propriété `Error.message`.
`NetStream.Seek.Notify`	état	L'opération de recherche est terminée.

Si vous recevez systématiquement des erreurs concernant la mémoire tampon, essayez de changer la mémoire tampon via la méthode `NetStream.setBufferTime()`.

Disponibilité

Flash Lite 3.0

Paramètres

infoObject:Object - Paramètre défini selon le message de statut ou 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("");
};
```

onCuePoint (gestionnaire NetStream.onCuePoint)

```
onCuePoint = function(infoObject:Object) {}
```

Appelé lorsqu'un point de repère intégré est atteint lors de la lecture d'un fichier FLV. Vous pouvez utiliser ce gestionnaire pour déclencher des actions dans votre code, lorsque la vidéo atteint un point de repère donné. Ceci vous permet de synchroniser d'autres actions dans votre application avec des événements de lecture vidéo.

Les points de repère que vous pouvez intégrer dans un fichier FLV sont de deux types.

Un point de repère de « navigation » spécifie une image-clé intégrée au fichier FLV, à laquelle correspond la propriété `time` de ce point. Ces points de repère de navigation sont souvent utilisés comme des signets ou des points d'entrée pour permettre aux utilisateurs de naviguer dans le fichier vidéo.

Un point de repère d'« événement » est spécifié par un minutage, que celui-ci corresponde ou non à une image-clé spécifique. Ces points de repère représentent généralement une heure dans la vidéo lorsqu'un événement qui survient est susceptible de déclencher d'autres événements d'application.

Le gestionnaire d'événements `onCuePoint` reçoit un objet doté des propriétés suivantes :

Propriété	Description
name	Nom donné au point de repère lors de son intégration dans le fichier FLV.
time	Moment (en secondes) auquel le point de repère se trouve dans le fichier vidéo lors de sa lecture.
type	Type de point de repère atteint (navigation ou événement).
parameters	Tableau associatif de paires nom/valeur spécifiées pour ce point de repère. Toute chaîne autorisée peut être utilisée pour le nom ou la valeur du paramètre.

Vous pouvez définir des points de repère dans un fichier FLV lorsque vous encodez le fichier pour la première fois ou lorsque vous importez un clip vidéo dans l'outil Flash Authoring à l'aide de l'assistant d'importation vidéo.

Le gestionnaire d'événements `onMetaData` récupère également des informations sur les points de repère dans un fichier vidéo. Toutefois, il collecte des informations sur tous les points de repère avant la lecture de la vidéo. Le gestionnaire d'événements `onCuePoint` reçoit des informations sur un point de repère au moment spécifié pour celui-ci lors de la lecture.

En général, si vous souhaitez que votre code corresponde à un point de repère particulier au moment où il survient, vous devez utiliser le gestionnaire d'événements `onCuePoint` pour déclencher une action dans votre code.

Vous pouvez utiliser la liste des points de repère transmise au gestionnaire d'événements `onMetaData()` pour permettre à l'utilisateur de lire la vidéo à des points prédéfinis du flux. Envoyez les valeurs de la propriété time du point de repère à la méthode `NetStream.seek()` pour lire la vidéo à partir de ce point de repère.

Disponibilité

Flash Lite 3.0

Paramètres

infoObject: Object - Objet contenant le nom, l'heure, le type et les paramètres du point de repère.

Exemple

Le code donné dans cet exemple commence par la création de nouveaux objets NetConnection et NetStream. Il définit ensuite le gestionnaire `onCuePoint()` de l'objet NetStream. Le gestionnaire passe en revue toutes les propriétés nommées dans l'objet infoObject reçu et imprime le nom et la valeur de la propriété. Lorsqu'il trouve les paramètres nommés de la propriété, il passe en revue chaque nom dans la liste et imprime leur nom et leur valeur.

```
var nc:NetConnection = new NetConnection();
nc.connect(null);
var ns:NetStream = new NetStream(nc);

ns.onCuePoint = function(infoObject:Object)
{
       trace("onCuePoint:");
       for (var propName:String in infoObject) {
           if (propName != "parameters")
           {
               trace(propName + " = " + infoObject[propName]);
           }
           else
           {
               trace("parameters =");
               if (infoObject.parameters != undefined) {
                   for (var paramName:String in infoObject.parameters)
                   {
                       trace(" " + paramName + ": " + infoObject.parameters[paramName]);
                   }
               }
               else
               {
                   trace("undefined");
               }
           }
       }
       trace("---------");
}

ns.play("http://www.helpexamples.com/flash/video/cuepoints.flv");
```

Les informations suivantes s'affichent alors :

```
onCuePoint:
parameters =
lights: beginning
type = navigation
time = 0.418
name = point1
---------
onCuePoint:
parameters =
lights: middle
type = navigation
time = 7.748
name = point2
---------
onCuePoint:
parameters =
lights: end
type = navigation
time = 16.02
name = point3
---------
```

Le nom de paramètre « lights » est un nom arbitraire utilisé par l'auteur de l'exemple de vidéo. Vous pouvez attribuer aux paramètres du point de repère le nom de votre choix.

onMetaData (gestionnaire NetStream.onMetaData)

```
onMetaData = function(infoObject:Object) {}
```

Invoqué lorsque Flash Lite 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

Flash Lite 3.0

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()` de l'objet NetStream. Le gestionnaire passe en revue 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.

Méthode close (NetStream.close)

```
public close() : Void
```

Arrête la lecture de toutes les 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 Lite 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 Media Server.

Disponibilité

Flash Lite 3.0

Remarque : *Cette méthode est également prise en charge dans Flash Player 6 lorsqu'elle est utilisée avec Flash Media Server. Pour plus d'informations, consultez la documentation de Flash Media Server.*

Exemple

La fonction `close()` 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();
};
```

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

Flash Lite 3.0

Remarque : *Cette méthode est également prise en charge dans Flash Player 6 lorsqu'elle est utilisée avec Flash Media Server. Pour plus d'informations, consultez la documentation de Flash Media Server.*

Paramètres

`flag : Boolean` [facultatif] - Valeur booléenne spécifiant la suspension `(true)` ou la reprise `(false)` de la lecture. 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
```

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 la section `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 employez cette méthode, considérez le 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 fichiers 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, consultez les sections suivantes :

Le livre blanc Sécurité de Flash Player 9 à l'adresse http://www.adobe.com/go/fp9_0_security_fr

Le livre blanc API relative à la sécurité de Flash Player 8 à l'adresse http://www.adobe.com/go/fp8_security_apis_fr

Disponibilité
Flash Lite 3.0

Remarque : *Cette méthode est également prise en charge dans Flash Player 6 lorsqu'elle est utilisée avec Flash Media Server. Pour plus d'informations, consultez la documentation de Flash Media Server.*

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 - Utilisez avec·Flash Media Server ; voir : la documentation de Flash Media Server.

`len` : Number - Utilisez avec·Flash Media Server ; voir : la documentation de Flash Media Server.

`reset` : Object - Utilisez avec·Flash Media Server ; voir : la documentation de Flash Media Server.

Exemple

L'exemple ci-dessous illustre diverses façons d'utiliser la méthode `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");
```

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 reprend lorsque cet emplacement est atteint.

Disponibilité

Flash Lite 3.0

Remarque : *Cette méthode est également prise en charge dans Flash Player 6 lorsqu'elle est utilisée avec Flash Media Server. Pour plus d'informations, consultez la documentation de Flash Media Server.*

Paramètres

offset:Number - Valeur de temps approximative, en secondes, à laquelle déplacer la tête de lecture dans un fichier FLV. La tête de lecture se place sur l'image clé de la vidéo qui est la plus proche de la valeur spécifiée par l'`offset`.

- Pour revenir au début du flux de diffusion, transmettez 0 à l'`offset`.
- 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 effectuer une recherche par rapport à la position courante, transmettez *my_ns*`.time + n` ou *my_ns*`.time - n` afin de vous positionner respectivement *n* secondes avant ou après la position courante. Par exemple, pour reculer de 20 secondes de la position courante, utilisez *my_ns*`.seek(`*my_ns*`.time - 20)`.

L'emplacement précis de la tête de lecture varie en fonction du paramètre fps (images par seconde) auquel la vidéo a été exportée. Par exemple, vous possédez deux objets vidéo représentant la même vidéo, l'un exporté à 6 fps et l'autre à 30. Si vous utilisez `my_ns.seek(15)` pour les deux objets, la tête de lecture se place à deux endroits différents.

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);
```

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 `bufferTime` sur 15 ; Flash commence la lecture du flux uniquement 15 secondes après la mise en mémoire tampon des données.

Disponibilité

Flash Lite 3.0

Remarque : *Cette méthode est également prise en charge dans Flash Player 6 lorsqu'elle est utilisée avec Flash Media Server. Pour plus d'informations, consultez la documentation de Flash Media Server.*

Paramètres

`bufferTime`:Number - Nombre de secondes pendant lesquelles les données sont placées en mémoire tampon avant que Flash ne les affiche. La valeur par défaut est 0,1 (un dixième de seconde).

Exemple

Reportez-vous à l'exemple de `NetStream.bufferLength`.

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

Flash Lite 2.0

Résumé des propriétés

Modificateurs	Propriété	Description
statique	MAX_VALUE : Number	Nombre représentable le plus élevé (double précision conformément à IEEE-754).
statique	MIN_VALUE : Number	Nombre représentable le plus faible (comportant deux décimales conformément à IEEE-754).
statique	NaN : Number	Valeur IEEE-754 ne représentant pas une valeur numérique (NaN).
statique	NEGATIVE_INFINITY : Number	Spécifie la valeur IEEE-754 représentant l'infini négatif.
statique	POSITIVE_INFINITY : Number	Spécifie la valeur IEEE-754 représentant l'infini positif.

Propriétés héritées de la classe Object

constructor (propriété Object.constructor), __proto__ (Object.__proto__, propriété), prototype (propriété Object.prototype), __resolve (Object.__resolve, propriété)

Récapitulatif des constructeurs

Signature	Description
Number (num : Object)	Crée un nouvel objet Number.

Résumé de la méthode

Modificateurs	Signature	Description
	toString (radix : Number) : String	Renvoie la représentation sous la forme d'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
Flash Lite 2.0

Exemple
Le code ActionScript suivant affiche les nombres représentables le plus élevé et le plus faible dans le panneau 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
Flash Lite 2.0

Exemple
Le code ActionScript suivant affiche les nombres représentables le plus élevé et le plus faible dans le panneau Sortie/fichier journal.

```
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
Flash Lite 2.0

Voir aussi
`isNaN, fonction`

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
Flash Lite 2.0

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
```

Constructeur Number

```
public Number(num:Object)
```

Crée un nouvel objet Number. Le nouveau constructeur `Number` est surtout utilisé en tant qu'espace réservé. Ne confondez pas l'objet Number avec la fonction `Number()`, qui convertit un paramètre en valeur primitive.

Disponibilité
Flash Lite 2.0

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 aussi
`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
Flash Lite 2.0

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

Flash Lite 2.0

Paramètres

`radix` : Number - Spécifie la base numérique (de 2 à 36) à appliquer pour la conversion nombre en 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

Flash Lite 2.0

Valeur renvoyée

Number - Chaîne.

Exemple

L'exemple suivant renvoie 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

Flash Lite 2.0

Résumé des propriétés

Modificateurs	Propriété	Description
	constructor : Object	Référence à la fonction constructor 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.
`statique`	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.

Récapitulatif 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 le chaînage de prototype de l'objet spécifié en tant qu'argument.
statique	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 `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`, la fonction de définition transmise est le paramètre `1` du numéro de type. 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

Flash Lite 2.0

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 est créée ; `false` sinon.

Exemple

Dans l'exemple suivant, un objet comporte deux méthodes internes, `setQuantity()` et `getQuantity()`. La propriété `bookcount` peut servir à appeler ces méthodes lorsqu'elle est définie ou récupérée. Une troisième méthode interne, `getTitle()`, renvoie une valeur en lecture seule associée à la propriété `bookname`. Lorsqu'un script récupère la valeur de `myBook.bookcount`, l'interprète d'ActionScript appelle automatiquement `myBook.getQuantity()`. Lorsqu'un script modifie la valeur de `myBook.bookcount`, l'interprète appelle `myObject.setQuantity()`. La propriété `bookname` ne spécifiant aucune fonction `set`, les tentatives de modification de `bookname` sont 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 fonctionne, 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. S'il y a de nombreuses propriétés, comme `bookcount` et `bookname` dans une classe, elles consomment énormément de mémoire. En revanche, vous pouvez ajouter ces propriétés à `Book.prototype` afin que les propriétés `bookcount` et `bookname` soient centralisées. Cependant, l'effet est le même que celui du code de l'exemple qui a ajouté `bookcount` et `bookname` directement dans 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 de 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 aussi

`getProperty, fonction`, `setInterval, fonction`

constructor (propriété Object.constructor)

```
public constructor : Object
```

Référence à la fonction constructor 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

Flash Lite 2.0

Exemple

L'exemple suivant constitue une référence à la fonction constructor de 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 aussi

`instanceof, opérateur`

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

Flash Lite 2.0

Paramètres

name : String - Chaîne indiquant le nom de la propriété.

Valeur renvoyée

Boolean - Valeur booléenne : `true` si la propriété est spécifiée par le paramètre `name` dans l'objet cible ; sinon, `false`.

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
Flash Lite 2.0

Paramètres
name : String - Nom de la propriété à vérifier.

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 aussi
`Instruction for..in`

isPrototypeOf (méthode Object.isPrototypeOf)

```
public isPrototypeOf(theClass:Object) : Boolean
```

Indique si une occurrence de la classe Object figure dans le chaînage de prototype 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
Flash Lite 2.0

Paramètres
theClass : Object - Classe de la chaîne de prototype dans laquelle vérifier l'objet.

Valeur renvoyée
Boolean - Valeur booléenne : `true` si l'objet figure dans le chaînage de prototype de l'objet spécifié par le paramètre `theClass` : sinon, `false`.

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
Flash Lite 2.0

Exemple
L'exemple suivant crée un objet générique appelé `myObject` :

```
var myObject:Object = new Object();
```

__proto__ (Object.__proto__, propriété)

```
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'interprète d'ActionScript utilise la propriété `__proto__` pour accéder à la propriété `prototype` de la classe ou de la fonction constructeur de l'objet afin de rechercher les propriétés et les méthodes héritées par l'objet de sa superclasse.

Disponibilité
Flash Lite 2.0

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 aussi
`prototype (propriété Object.prototype)`

prototype (propriété Object.prototype)

```
public static prototype : Object
```

Une référence à la superclasse d'un objet classe ou fonction. La propriété `prototype` est créée automatiquement et 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

Flash Lite 2.0

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 remonter 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 aussi

`__proto__` (Object.__proto__, propriété)

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 créée à l'aide de `MovieClip.attachMovie()` ou de `MovieClip.duplicateMovieClip()`, elle est enregistrée dans la classe spécifiée par `theClass` et non 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 de la classe appropriée en utilisant le mot-clé `this` pointant vers l'objet. La fonction constructor 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

Flash Lite 2.0

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 constructor de la classe ActionScript ou `null` pour annuler l'enregistrement du symbole.

Valeur renvoyée

Boolean - Valeur booléenne : si l'enregistrement dans la classe réussit, la valeur `true` est renvoyée ; la valeur `false` est renvoyée dans tous les autres cas.

Voir aussi

`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 une méthode non définie d'un objet, Flash Lite 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 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

Flash Lite 2.0

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.

Utilisation 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!
```

Utilisation 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
```

Utilisation 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
```

Utilisation 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
```

Utilisation 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 aussi

`arguments`, `Array`

toString (méthode Object.toString)

```
public toString() : String
```

Convertit l'objet spécifié en chaîne et renvoie cette dernière.

Disponibilité

Flash Lite 2.0

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
Flash Lite 2.0

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 est supprimé ; `false` sinon.

Exemple
Consultez l'exemple de `Object.watch()`.

Voir aussi
`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
Flash Lite 2.0

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
```

Reportez-vous à l'exemple de `Object.toString()` pour voir un exemple de la valeur renvoyée par `Object.valueOf()` pour une classe personnalisée qui remplace `toString()`.

Voir aussi

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 `callback` pour affecter la valeur de la propriété que vous observez. La valeur renvoyée par votre méthode `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 de la propriété, renvoyez le paramètre `oldVal`.

Si la méthode `callback` que vous définissez n'a pas d'instruction `return`, la propriété de l'objet observé prend la 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 appels suivants 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 `callback`. Pour pouvoir travailler avec une propriété de lecture/définition, `Object.watch()` doit évaluer la propriété en permanence, ce qui n'est pas efficace.

En général, les propriétés ActionScript prédéfinies, telles que `_x`, `_y`, `_width` et `_height`, sont des propriétés de lecture/définition que `Object.watch()` ne peut pas observer.

Disponibilité
Flash Lite 2.0

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 `callback` est `callback(prop, oldVal, newVal, userData)`.

userData : Object [facultatif] - Ensemble arbitraire de données ActionScript 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 est créé ; `false` sinon.

Exemple
L'exemple suivant utilise `watch()` pour vérifier si la propriété `speed` excède la limite 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 aussi

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

Flash Lite 3.1

Résumé des propriétés

Modificateurs	Propriété	Description
	length : Number	La longueur du segment de ligne de (0,0) à ce point.
	x : Number	Les coordonnées horizontales du point.
	y : Number	Les coordonnées verticales du point.

constructor (propriété Object.constructor), __proto__ (Object.__proto__, propriété), prototype (propriété Object.prototype), __resolve (Object.__resolve, propriété)

Récapitulatif 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.
statique	distance(pt1:Point, pt2:Point) : Number	Renvoie la distance entre pt1 et pt2.
	equals(toCompare:Object) : Boolean	Détermine si deux points sont égaux.
statique	interpolate(pt1:Point, pt2:Point , f:Number) : Point	Détermine un point entre deux points spécifiés.
	Normaliser(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 de la quantité spécifiée.
statique	polar(len:Number, angle:Number) : Point	Convertit une paire de coordonnées polaires en coordonnées cartésiennes.
	soustraction(v:Point : Point	Soustrait les coordonnées d'un autre point à celles de ce point pour créer un nouveau point.
	toString (méthode Point.toString)() : String	Renvoie une chaîne qui contient les valeurs des coordonnées *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)

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

Flash Lite 3.1

Paramètres

v : Point - Le point à ajouter.

Valeur renvoyée

Point - Le 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

Flash Lite 3.1

Valeur renvoyée

Point - Nouvel objet Point.

Exemple

L'exemple suivant crée une copie de l'objet Point appelée `clonedPoint` à partir des valeurs trouvées dans l'objet myPoint. L'objet 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
Flash Lite 3.1

Paramètres
pt1 : Point - Le premier point.

pt2 : Point - Le second point.

Valeur renvoyée
Number - Distance entre le premier et le second 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
Flash Lite 3.1

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
Flash Lite 3.1

Paramètres
pt1 : Point - Le premier point.

pt2 : Point - Le second 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
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)

```
public length : Number
```

La longueur du segment de ligne de (0,0) à ce point.

Disponibilité
Flash Lite 3.1

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 aussi
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
Flash Lite 3.1

Paramètres

length : Number - Valeur de redimensionnement. Si, par exemple, le point actuel se trouve à (0,5) et que vous le normalisez à 1, les coordonnées du point renvoyé sont (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 aussi

length (propriété Point.length)

offset (méthode Point.offset)

```
public offset(dx:Number, dy:Number) : Void
```

Décale l'objet Point de la quantité spécifiée. 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

Flash Lite 3.1

Paramètres

dx : Number - Valeur de décalage pour la coordonnée horizontale, *x*.

dy : Number - Valeur de décalage pour 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 aussi

add (méthode Point.add)

Constructeur Point

```
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
Flash Lite 3.1

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 `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
Flash Lite 3.1

Paramètres
len : Number - Coordonnée de longueur de la paire polaire.

angle : Number - Angle, en radians, de la paire polaire.

Valeur renvoyée
Point - Point cartésien.

Exemple
L'exemple suivant crée un objet Point `cartesianPoint` à partir de la valeur 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 droits avec des côtés ayant des rapports 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 aussi

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

Flash Lite 3.1

Paramètres

v : Point - Point à soustraire.

Valeur renvoyée

Point - Le 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

Flash Lite 3.1

Valeur renvoyée

String - 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 (propriété Point.x.)

```
public x : Number
```

Les coordonnées horizontales du point. La valeur par défaut est 0.

Disponibilité

Flash Lite 3.1

Exemple

L'exemple suivant crée un objet Point`myPoint` et définit la valeur de la 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
```

Les coordonnées verticales du point. La valeur par défaut est 0.

Disponibilité

Flash Lite 3.1

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
```

Rectangle (flash.geom.Rectangle)

```
Object
    |
    +-flash.geom.Rectangle

public class Rectangle
extends Object
```

La classe Rectangle permet de créer et de modifier les objets Rectangle. Un objet Rectangle est une zone définie par son emplacement, indiqué 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 propriétés n'a aucun effet sur les autres. Cependant, les propriétés `right` et `bottom` sont liées de façon intégrale à ces quatre propriétés. Ainsi, si vous changez la valeur `right`, vous modifiez également la valeur `width`, et si vous modifiez la valeur `bottom`, vous modifiez la valeur `height`, etc. De plus, la propriété `right` ou *x* doit être établie avant de définir la propriété `width` ou `right`.

Disponibilité

Flash Lite 3.1

Résumé des propriétés

Modificateurs	Propriété	Description
	bas : Number	Somme des propriétés `y` et `height`.
	bottomright : Point	L'emplacement du coin inférieur droit de l'objet Rectangle déterminé par la valeur des propriétés `x` et `y`.
	height : Number	La hauteur du rectangle en pixels.
	gauche : Number	Coordonnée *x* du coin supérieur gauche du rectangle.
	droite : Number	Somme des propriétés `x` et `width`.
	size : Point	Taille de l'objet Rectangle, exprimée en tant qu'objet Point avec les valeurs des propriétés `width` et `height`.
	haut : Number	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	Largeur de l'objet Rectangle en pixels.
	x : Number	Coordonnée *x* du coin supérieur gauche du rectangle.
	y : Number	Coordonnée *y* du coin supérieur gauche du rectangle.

`Constructeur Object`, `__proto__ (Object.__proto__, propriété)`, `prototype (propriété Object.prototype)`, `__resolve (Object.__resolve, propriété)`

Récapitulatif 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 et height.
	contains(x:Number, y:Number) : Boolean	Détermine si le point spécifié figure dans la zone rectangulaire définie par cet objet Rectangle.
	containsPoint(pt:Point (flash.geom.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 quantités spécifiées.
	offsetPoint(pt:Point) : Void	Règle l'emplacement de l'objet Rectangle en utilisant un objet Point comme 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 les sépare.

« addProperty (méthode Object.addProperty) » à la page 504, « hasOwnProperty (méthode Object.hasOwnProperty) » à la page 507isPropertyEnumerable (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

Somme des propriétés `y` et `height`.

Disponibilité

Flash Lite 3.1

Exemple

L'exemple suivant crée un objet Rectangle et modifie la valeur de sa propriété `bottom` de 15 à 30. Notez que la valeur de `rect.height` est également modifiée 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
```

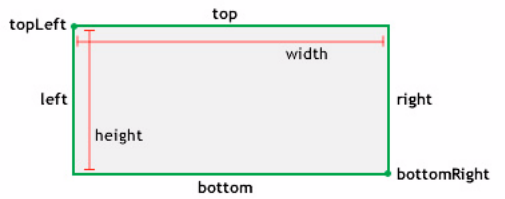
Voir aussi

`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 la valeur des propriétés `x` et `y`.



Disponibilité

Flash Lite 3.1

Exemple

L'exemple suivant définit la propriété `bottomRight` 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 aussi

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` et `height`.

Disponibilité

Flash Lite 3.1

Valeur renvoyée

Rectangle - Nouvel objet Rectangle avec les mêmes valeurs que l'objet Rectangle d'origine pour les propriétés `x`, `y`, `width` et `height`.

Exemple

L'exemple suivant crée trois objets Rectangle et les compare. `rect_1` est créé à l'aide du constructeur Rectangle. `rect_2` est créé en le définissant comme égal à `rect_1`. En outre, `clonedRect` est créé en clonant `rect_1`. Vous remarquerez que si `rect_2` est évalué comme étant égal à `rect_1`, ce n'est pas le cas de `clonedRect`, qui contient pourtant les mêmes valeurs que `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 démontrer les relations entre `rect_1`, `rect_2` et `clonedRect`, l'exemple ci-dessous modifie la propriété `x` de `rect_1`. La modification de `x` démontre que la méthode `clone()` crée une occurrence reposant sur les valeurs de `rect_1` au lieu de pointer vers elles par 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 aussi

`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

Flash Lite 3.1

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 aussi

Point (flash.geom.Point)

containsPoint (méthode Rectangle.containsPoint)

```
public containsPoint(pt:Point (flash.geom.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

Flash Lite 3.1

Paramètres

pt : Point - Point, représenté par ses valeurs *x*,*y*.

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 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 aussi

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
Flash Lite 3.1

Paramètres
rect : Rectangle - Objet Rectangle en cours de vérification.

Valeur renvoyée
Boolean - Si l'objet Rectangle que vous spécifiez est compris dans cet objet Rectangle, renvoie `true` ; renvoie `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
Flash Lite 3.1

Paramètres
toCompare : Object - Rectangle que vous voulez 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`, renvoie `true` ; renvoie `false` 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 `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));
```

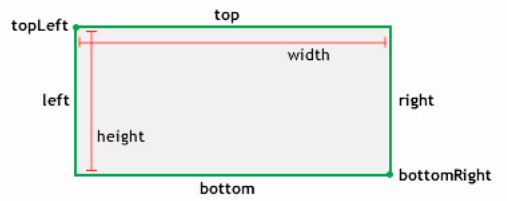
Voir aussi

`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`.



Disponibilité

Flash Lite 3.1

Exemple

L'exemple suivant crée un objet Rectangle et modifie sa propriété `height` de 10 à 20. Remarquez que `rect.bottom` est également modifié.

```
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 aussi

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 la valeur de `dx` sur la gauche et la droite, et de la valeur de `dy` vers le haut et bas.

Disponibilité

Flash Lite 3.1

Paramètres

dx : Number - Valeur à ajouter sur la gauche et la droite de l'objet Rectangle. L'équation suivante permet de 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 aussi

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.

Les deux exemples de code suivants donnent le même résultat :

```
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

Flash Lite 3.1

Paramètres

pt : 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)
```

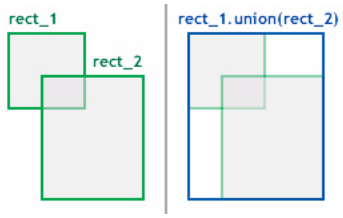
Voir aussi

`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

Flash Lite 3.1

Paramètres

toIntersect : Rectangle - Objet Rectangle à prendre comme comparaison pour voir s'il recoupe cet objet Rectangle.

Valeur renvoyée

Rectangle - Objet Rectangle qui correspond à la zone d'intersection. Si les rectangles ne se recoupent pas, cette méthode renvoie un objet Rectangle vide, c'est-à-dire un rectangle 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

Flash Lite 3.1

Paramètres

toIntersect : Rectangle - Objet Rectangle à comparer à cet objet Rectangle.

Valeur renvoyée

Boolean - Si l'objet spécifié recoupe cet objet Rectangle, renvoie `true` ; renvoie `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 aussi

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

Flash Lite 3.1

Valeur renvoyée

« Boolean » à la page 222 - Si la largeur de l'objet Rectangle ou sa hauteur est inférieure ou égale à 0, renvoie `true` ; renvoie `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.

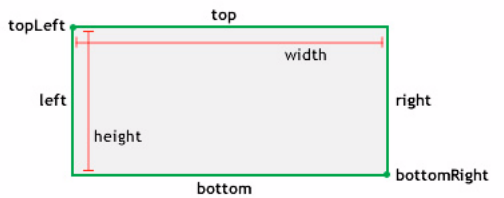
```
import flash.geom.Rectangle;

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
```

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
Flash Lite 3.1

Exemple
L'exemple suivant fait passer la propriété `left` de 0 à 10. Remarquez que `rect.x` est également modifié.

```
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 aussi
x (propriété Rectangle.x),y (propriété Rectangle.y),« width (propriété Rectangle.width) » à la page 547,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 quantités spécifiées.

Disponibilité
Flash Lite 3.1

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 comme paramètre. Cette méthode est similaire à la méthode `Rectangle.offset()`, à ceci près qu'elle prend un objet Point comme paramètre.

Disponibilité
Flash Lite 3.1

Paramètres
pt : 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 aussi
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
Flash Lite 3.1

Paramètres
x : Number - Coordonnée *x* du coin supérieur gauche du rectangle.

y : Number - Coordonnée *y* du coin supérieur gauche du rectangle.

width : Number - Largeur de l'objet Rectangle en pixels.

height : Number - La 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 aussi
x (propriété Rectangle.x), y (propriété Rectangle.y), « width (propriété Rectangle.width) » à la page 547, height (propriété Rectangle.height)

right (propriété Rectangle.right)

```
public right : Number
```

Somme des propriétés `x` et `width`.

Disponibilité

Flash Lite 3.1

Exemple

L'exemple suivant crée un objet Rectangle et modifie sa propriété `right` de 15 à 30. Remarquez que `rect.width` est également modifié.

```
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 aussi

`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

Flash Lite 3.1

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 aussi

x (propriété Rectangle.x), y (propriété Rectangle.y), width (propriété Rectangle.width), height (propriété Rectangle.height)

size (propriété Rectangle.size)

```
public size : Point
```

Taille de l'objet Rectangle, exprimée en tant qu'objet Point avec les valeurs des propriétés `width` et `height`.

Disponibilité

Flash Lite 3.1

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 aussi

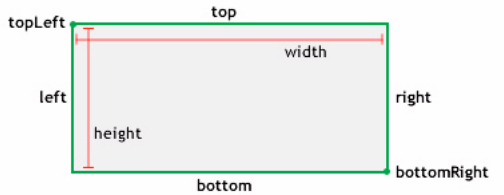
Point (flash.geom.Point)

top (propriété Rectangle.top)

```
public top : Number
```

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 `y`.



Disponibilité
Flash Lite 3.1

Exemple
Cet exemple modifie la valeur de la propriété `top` de 0 à 10. Remarquez que `rect.y` est également modifié.

```
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 aussi
`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
Flash Lite 3.1

Exemple
L'exemple suivant définit la propriété `topLeft` 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 aussi

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

Flash Lite 3.1

Valeur renvoyée

String - Chaîne répertoriant la valeur de chacune des propriétés suivantes de l'objet Rectangle : `x`, `y`, `width` et `height`.

Exemple

L'exemple suivant concatène une représentation sous forme de chaîne 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)
```

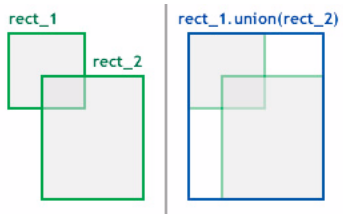
Voir aussi

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 les sépare.



Disponibilité
Flash Lite 3.1

Paramètres
toUnion : Rectangle - Objet Rectangle à ajouter à cet objet Rectangle.

Valeur renvoyée
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 `x=20`, `y=50`, `width=60` et `height=30` (20, 50, 60, 30), et un second 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
```

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
Flash Lite 3.1

Exemple

L'exemple suivant crée un objet Rectangle et modifie sa propriété `height` de 10 à 20. Remarquez que `rect.right` est également modifié.

```
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 aussi

`x (propriété Rectangle.x)`, `y (propriété Rectangle.y)`, `height (propriété Rectangle.height)`

x (propriété Rectangle.x)

```
public x : Number
```

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

Flash Lite 3.1

Exemple

L'exemple suivant crée un objet Rectangle vide et définit sa propriété `x` sur 10. Remarquez que `rect.left` est également modifié.

```
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 aussi

`left (propriété Rectangle.left)`

y (propriété Rectangle.y)

```
public y : Number
```

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
Flash Lite 3.1

Exemple
L'exemple suivant crée un objet Rectangle vide et définit sa propriété `y` sur 10. Remarquez que `rect.top` est également modifié.

```
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 aussi
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
```

La classe System.security contient des méthodes spécifiant la façon dont les fichiers SWF peuvent communiquer entre eux dans différents domaines.

Disponibilité
Flash Lite 2.0

Résumé des propriétés
Propriétés héritées de la classe Object

constructor (propriété Object.constructor), __proto__ (Object.__proto__, propriété) prototype (propriété Object.prototype), __resolve (Object.__resolve, propriété)

Résumé de la méthode

Modificateurs	Signature	Description
statique	allowDomain(domain1:String) : Void	Permet aux fichiers SWF et HTML des domaines identifiés d'accéder aux objets et variables du fichier SWF appelant ou de tout autre fichier SWF provenant du même domaine que le fichier SWF appelant.
statique	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.
statique	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 des domaines identifiés d'accéder aux objets et variables du fichier SWF appelant ou de tout autre fichier SWF provenant du même domaine que le fichier SWF appelant.

Pour les fichiers nécessitant une version inférieure ou égale à Flash Player 7, les paramètres transmis doivent appliquer les règles d'appellation de domaine exact. Par exemple, pour autoriser l'accès à des fichiers SWF hébergés sur www.domain.com ou store.domain.com, ces deux noms de domaine doivent être transmis :

```
// For Flash Player 6
System.security.allowDomain("domain.com");
// Corresponding commands to allow access by SWF files
// that are running in Flash Player 7 or later
System.security.allowDomain("www.domain.com", "store.domain.com");
```

De plus, pour les fichiers exécutés avec 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 `System.security.allowInsecureDomain()` à la place.

La situation suivante peut parfois se produire : 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 argument de cette méthode. Par exemple, si vous chargez un fichier SWF dans `my_mc`, vous pouvez appeler `System.security.allowDomain(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 la fin du chargement d'un fichier SWF enfant consiste à utiliser `MovieClipLoader.onLoadComplete`.

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. 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
Flash Lite 2.0

Paramètres
domain1 : String - Une ou plusieurs chaînes qui identifient les domaines autorisés à accéder aux données et variables du fichier SWF contenant l'appel `System.Security.allowDomain()`. Les domaines peuvent être formatés de différentes façons :

- `"domain.com"`
- `"http://domain.com"`
- `"http://IPaddress"`

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 empêcherait MovieB d'accéder aux objets et aux variables de MovieA.

Voir aussi
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. Il permet également aux fichiers SWF des domaines identifiés d'accéder aux fichiers SWF situés dans le même domaine que le fichier SWF ayant effectué l'appel.

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.

Adobe déconseille l'emploi de cette méthode pour remplacer le comportement par défaut, dans la mesure où cette opération entraîne des failles de sécurité HTTPS. Cependant, elle reste utile dans certains cas, 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 `System.security.allowDomain()` pour autoriser un accès HTTP vers HTTPS. Cependant, dans la mesure où la sécurité est implémentée différemment dans Flash Player 7, vous devez utiliser `System.Security.allowInsecureDomain()` pour autoriser ce type d'accès dans les fichiers SWF publiés pour Flash Player 7 ou une version plus récente.

Remarque : Il est parfois nécessaire d'appeler `System.security.allowInsecureDomain()` avec un argument qui correspond exactement au domaine du fichier SWF dans lequel cet appel apparaît. Ceci diffère de `System.security.allowDomain()`, qui ne doit pas nécessairement être appelé en utilisant le domaine du fichier SWF en tant qu'argument. Ceci est parfois requis avec `System.security.allowInsecureDomain()`, car, par défaut, un fichier SWF figurant dans http://foo.com ne peut pas exécuter de script sur un fichier SWF enregistré dans https://foo.com, bien que ces deux domaines soient identiques.

Disponibilité
Flash Lite 2.0

Paramètres
domain : String - Nom de domaine exact, tel que www.myDomainName.com ou store.myDomainName.com.

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 aussi
allowDomain (méthode security.allowDomain)

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 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 `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 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 1 024 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.

Disponibilité
Flash Lite 2.0

Paramètres
url : String - Chaîne. URL où réside le fichier de régulation interdomaine à charger.

Selection

```
Object
    |
    +-Selection
```

```
public class Selection
extends Object
```

La classe Selection vous permet de définir et de contrôler le champ 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.

L'objet Selection n'est valide que si le périphérique prend en charge la saisie de texte en ligne. Dans le cas contraire, et s'il repose sur un processeur frontal (FEP - front-end processor) pour saisir du texte, tous les appels à l'objet Selection sont ignorés.

Disponibilité
Flash Lite 2.0

Résumé des propriétés
Propriétés héritées de la classe Object

constructor (propriété Object.constructor), __proto__ (Object.__proto__, propriété), prototype (propriété Object.prototype), __resolve (Object.__resolve, propriété)

Résumé des événements

Evénement	Description
onSetFocus = function([oldfocus], [newfocus]) {}	Notifié lorsque le focus d'entrée change.

Résumé de la méthode

Modificateurs	Signature	Description
statique	addListener(listener: Object) : Void	Enregistre un objet pour recevoir les notifications de changement du focus clavier.
statique	getFocus() : String	Renvoie une chaîne spécifiant le chemin cible de l'objet ayant le focus.

Modificateurs	Signature	Description
statique	removeListener(listener:Object) : Boolean	Supprime un objet précédemment enregistré avec la méthode Selection.addListener().
statique	setFocus(newFocus:Object) : Boolean	Donne le focus au champ texte, bouton ou clip sélectionnable (modifiable) spécifié par le paramètre newFocus.
statique	setSelection(beginIndex:Number, endIndex:Number) : Void	Définit la plage de sélection du champ 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 la méthode `Selection.setFocus()` est appelée), la méthode `onSetFocus()` de tous les objets d'écoute enregistrés avec `addListener()` est appelée. Plusieurs objets peuvent écouter les notifications de changement de focus. Si l'écouteur spécifié est déjà enregistré, aucun changement ne se produit.

Disponibilité

Flash Lite 2.0

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 texte qui a perdu le focus et une référence au champ texte qui a reçu le focus. La fonction définit la propriété `border` du champ texte qui a perdu le focus sur `false` et définit la propriété border du champ 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);
```

Voir aussi

setFocus (méthode Selection.setFocus)

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, la méthode `getFocus()` 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, la méthode `getFocus()` renvoie le chemin cible de l'objet bouton ou du clip du bouton.
- Si ni un objet TextField, ni un bouton, un une occurrence de composant, ni un clip de bouton n'a le focus, la méthode `getFocus()` renvoie la valeur `null`.

Disponibilité

Flash Lite 2.0

Valeur renvoyée

String - Chaîne ou `null`.

Exemple

L'exemple suivant crée un champ texte permettant de renvoyer le chemin de l'objet qui a le focus. Il applique ensuite une fonction interval pour mettre à jour le champ de façon régulière. Pour effectuer un test, ajoutez plusieurs instances de bouton à la scène avec différentes instance de nom, puis ajoutez le code ActionScript suivant à votre fichier AS ou FLA.

```
this.createTextField("status_txt", this.getNextHighestDepth(), 0, 0, 150, 25);

function FocusUpdate()
{
    s = Selection.getFocus();
    if ( s )
    {
        status_txt.text = s;
    }
}

setInterval( FocusUpdate, 100 );
```

Voir aussi

onSetFocus (écouteur d'événement Selection.onSetFocus), setFocus (méthode Selection.setFocus)

onSetFocus (écouteur d'événement Selection.onSetFocus)

```
onSetFocus = function([oldfocus], [newfocus]) {}
```

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 la méthode `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

Flash Lite 2.0

Paramètres

oldfocus : [facultatif] - Objet qui perd le focus.

newfocus : [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);
```

Voir aussi

addListener (méthode Selection.addListener), setFocus (méthode Selection.setFocus)

removeListener (méthode Selection.removeListener)

```
public static removeListener(listener:Object) : Boolean
```

Supprime un objet précédemment enregistré avec la méthode `Selection.addListener()`.

Disponibilité

Flash Lite 2.0

Paramètres

listener : Object - Objet qui ne reçoit plus aucune notification de focus.

Valeur renvoyée

Boolean - Si l'objet listener a été supprimé, la méthode renvoie une valeur `true`. Si l'objet listener n'a pas été correctement supprimé, (par exemple, si `listener` ne figurait pas dans la liste des écouteurs de l'objet Selection), la méthode renvoie la valeur `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 aussi

addListener (méthode Selection.addListener)

setFocus (méthode Selection.setFocus)

```
public static setFocus(newFocus:Object) : Boolean
```

Donne le focus au champ texte, bouton ou clip sélectionnable (modifiable) spécifié par le paramètre `newFocus`. Vous pouvez utiliser la notation avec point ou avec barre oblique pour spécifier le chemin. Vous pouvez également utiliser un chemin relatif ou absolu. Si vous utilisez ActionScript 2.0, vous devez utiliser la notation avec point.

Si la valeur `null` est transmise, le focus actuel est supprimé.

Disponibilité

Flash Lite 2.0

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.

Valeur renvoyée

Boolean - Valeur booléenne : `true` si la tentative de focus aboutit, `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 d'envoi, 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");
fscommand("activateTextField");
//
function checkForm():Boolean {
    if (username_txt.text.length == 0) {
    status_txt.text = "fill in username";
    Selection.setFocus("username_txt");
    fscommand("activateTextField");
    return false;
    }
    if (password_txt.text.length == 0) {
    status_txt.text = "fill in password";
    Selection.setFocus("password_txt");
    fscommand("activateTextField");
    return false;
    }
    status_txt.text = "success!";
    Selection.setFocus(null);
    return true;
}
```

Voir aussi

getFocus (méthode Selection.getFocus)

setSelection (méthode Selection.setSelection)

```
public static setSelection(beginIndex:Number, endIndex:Number) : Void
```

Définit la plage de sélection du champ 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 texte n'a le focus. Lorsque vous appelez la méthode `setSelection()`, et si un contrôle de texte a le focus, la sélection n'est mise en valeur que si le champ texte est modifié de façon active. La méthode `setSelection()` peut être appelée après `Selection.setFocus()` ou à partir d'un gestionnaire d'événements `onSetFocus()`, les sélections ne sont visibles qu'après l'appel de la commande `fscommand activateTextField`.

Disponibilité

Flash Lite 2.0

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 associe ensuite un gestionnaire d'événements à l'événement `onSetFocus` qui sélectionne tout le champ texte et active la session de modification.

Remarque : Si la méthode `Selection.setSelection()` est appelée, le texte n'est pas affiché à l'écran tant que le champ texte n'est pas activé (suite à un appel à la commande `fscommand activateTextField`).

```
this.createTextField("myText_txt", 99, 10, 10, 200, 30);
myText_txt.type = "input";
myText_txt.text = "this is my text";
myText_txt.onSetFocus = function(){
    Selection.setSelection(0,myText_txt.text.length);
    fscommand("activateTextField");
}
```

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 version Flash Lite de la classe SharedObject permet aux fichiers Flash SWF d'enregistrer des données sur le périphérique lorsqu'il est fermé et de charger directement ces données lorsque le fichier est lu de nouveau. Les objets partagés Flash Lite stockent un ensemble de paires nom-valeur sur le périphérique.

Remarque : Le nom « SharedObject » est dérivé de la classe SharedObject de Flash. La version Flash de cette classe permet de partager les données enregistrées de plusieurs fichiers SWF Flash. Cependant, la version Flash Lite de la classe SharedObject ne prend pas en charge le partage des données entre différents fichiers SWF Flash.

Dans Flash Lite, un fichier SWF est considéré comme étant de version différente s'il a été modifié par rapport à la version d'origine, même s'il porte le même nom. Ceci diffère de Flash Player, où un fichier SWF est considéré comme identique si son URL et son nom sont identiques, même si le fichier SWF a été modifié. Dans Flash Lite, deux versions différentes d'un fichier SWF peuvent accéder mutuellement à leurs objets partagés.

Pour assurer la cohérence de la plate-forme Flash, les éléments et conventions d'appellation ActionScript sont repris par le lecteur Flash Lite.

Les exemples suivants portent sur le potentiel d'utilisation des objets partagés :

- Vous pouvez utiliser une application Flash en tant qu'interface utilisateur pour un service permettant à l'utilisateur de consulter les listes de voitures d'occasion. L'application se connecte à un serveur qui donne la liste des voitures en fonctions des critères de recherche et des préférences de l'utilisateur. L'application Flash peut enregistrer la dernière recherche de l'utilisateur et commencer à renseigner les formulaires lors des lectures suivantes du fichier SWF. Pour ce faire, créez une occurrence de SharedObject pour stocker les paramètres de recherche à chaque fois que l'utilisateur effectue une nouvelle recherche. Lorsque le fichier SWF se ferme, le lecteur enregistre les données dans l'objet partagé, sur le périphérique. Lorsque le fichier SWF est lu de nouveau, le lecteur Flash Lite charge l'objet partagé et commence de renseigner le formulaire de recherche en fonction des derniers critères saisis par l'utilisateur.
- Vous pouvez utiliser une application Flash en tant qu'interface utilisateur pour un service permettant aux utilisateurs de consulter des critiques musicales. L'application permet aux utilisateurs de stocker des informations sur leurs albums préférés. Ces informations peuvent être stockées sur le serveur distant, mais ceci risque de poser des problèmes si l'application est incapable de se connecter au service. En outre, l'extraction des données à partir d'un service distant peut être lente et risque de détériorer les conditions d'utilisation. Les objets partagés permettent à l'application de stocker les informations relatives aux albums sur le périphérique et de les charger rapidement en cas de besoin.

Remarque : Dans la mesure où l'espace est limité sur les périphériques mobiles, les données ne sont pas totalement rémanentes. Dans certains cas, la plate-forme peut supprimer les données les plus anciennes.

Pour créer un objet partagé localement, utilisez la syntaxe suivante :

```
var so:shared object = shared object.getLocal("mySharedObject");
```

La lecture et l'écriture de données sur un combiné peuvent être longues. Pour vous assurer que les données sont disponibles immédiatement lorsque l'application les demande au périphérique, Flash Lite 2.0 nécessite la mise en place d'un objet écouteur. Le lecteur appelle cet écouteur lorsque le périphérique a chargé les données de l'objet partagé. Les méthodes qui accèdent à l'occurrence SharedObject renvoyée par l'appel à `getLocal()` doivent attendre l'appel de l'écouteur avant de tenter une autre opération.

Disponibilité
Flash Lite 2.0

Exemple
Dans l'exemple suivant, un fichier SWF crée une fonction d'écoute appelée `Prefs`, puis crée un objet partagé. Le lecteur appelle la fonction `loadCompletePrefs` lorsque les données sont disponibles.

```
function loadCompletePrefs (mySO:SharedObject) {
    if (0 == mySO.getSize() )
    {
    // If the size is 0, we need to initialize the data:
        mySO.data.name = "Sigismund";
        mySO.data.email = "siggy@macromedia.com";
    }
    else
    {
        // Trace all the data in mySO:
        trace( "Prefs:" );
        for (var idx in mySO.data) {
        trace( " " + idx +": " + mySO.data[idx] );
        }
    }
}

SharedObject.addListener( "Prefs", loadCompletePrefs );

// We can now create the shared object:
var Prefs:SharedObject = SharedObject.getLocal("Prefs");
```

Lorsque le lecteur a indiqué à l'objet écouteur que les données sont disponibles, l'application peut utiliser l'objet partagé renvoyé par l'appel à la méthode `getLocal()` de la même façon qu'un objet partagé dans Flash. L'application peut ajouter, modifier ou supprimer des propriétés pendant la lecture du contenu. Lorsque le contenu est déchargé, l'objet partagé peut être écrit sur le périphérique. Cependant, pour garantir que l'objet partagé va être écrit sur le périphérique, l'application doit forcer une opération d'écriture en appelant la méthode `flush()`.

Les objets partagés de Flash Lite ne sont disponibles que pour les fichiers SWF stockés. Les fichiers SWF lus dans un navigateur compatible réseau ne peuvent pas utiliser les objets partagés de Flash Lite.

Le montant total de stockage des objets partagés de Flash Lite par fichier SWF est conditionné par le périphérique. Vous pouvez déterminer cette taille avec la méthode `SharedObject.getMaxSize()`.

Remarque : Les objets partagés distants ne sont pas pris en charge par Flash Lite 2.0.

Voir aussi

flush (méthode SharedObject.flush), onStatus (gestionnaire SharedObject.onStatus)

Résumé des propriétés

Modificateurs	Propriété	Description
	data : Object	L'ensemble des attributs associés à la propriété `data` de l'objet.

Propriétés héritées de la classe Object

constructor (propriété Object.constructor), __proto__ (Object.__proto__, propriété), prototype (propriété Object.prototype), __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
statique	addListener(objectName:String, notifyFunction:Function) : Void	Crée un écouteur d'événements appelé par le lecteur Flash Lite lorsque le lecteur a chargé les données de l'objet partagé à partir du périphérique.
	clear() : Void	Purge toutes les données de l'objet partagé, puis supprime cet objet du disque.
	flush(minDiskSpace:Number) : Object	Ecrit l'objet partagé dans un fichier local et permanent.
statique	getLocal(name:String) : SharedObject	Renvoie une référence à un objet partagé rémanent du périphérique et disponible uniquement pour le client actuel.
statique	getMaxSize() : Number	Renvoie le nombre total d'octets que le fichier SWF peut utiliser pour stocker des objets mobiles partagés sur le périphérique.
	getSize() : Number	Obtient la taille actuelle de l'objet partagé, en octets.
statique	removeListener(objectName:String)	Supprime tous les écouteurs qui ont été ajoutés à l'aide de la méthode 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 SharedObject.addListener)

```
public static addListener(objectName:String, notifyFunction:Function) : Void
```

Crée un écouteur d'événements appelé par le lecteur Flash Lite lorsque le lecteur a chargé les données de l'objet partagé à partir du périphérique. Les méthodes qui accèdent à l'occurrence SharedObject renvoyée par l'appel à la méthode `getLocal()` doivent attendre l'appel de cette fonction avant de tenter une autre opération.

Disponibilité

Flash Lite 2.0

Paramètres

objectName : String - Chaîne indiquant le nom de l'objet partagé.

notifyFunction : Function - Nom de la fonction que le lecteur appelle pour notifier l'application que la méthode `getLocal()` a été exécutée et que les données ont terminé leur chargement.

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

Flash Lite 2.0

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
```

L'ensemble des attributs associés à la propriété `data` de l'objet. Chaque attribut peut être un objet de type ActionScript ou JavaScript de base : Array, Number, Boolean, etc. Par exemple, les lignes suivantes affectent des valeurs à différents aspects d'un objet partagé.

Remarque : Pour Flash Lite, si l'objet écouteur n'a pas été appelé, la propriété `data` pourrait contenir des valeurs non définies. Consultez la description de la méthode `addListener()` pour plus de détails.

```
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]);
}

soResult = "";
for (var prop in my_so.data) {
    soResult += prop+": "+my_so.data[prop] +"\n";
}
result.text = soResult;
```

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 de valeurs directement à la propriété `data` d'un objet partagé (par exemple, `so.data = someValue`). Flash ignore ces affectations.

Pour supprimer les attributs 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 *private* pour un objet partagé (des valeurs qui sont disponibles uniquement pour l'instance cliente tandis que l'objet est en cours d'utilisation et qui ne sont pas stockées avec l'objet lorsqu'il est fermé), créez des propriétés qui ne portent pas le nom `data` pour les stocker, comme indiqué dans l'exemple ci-après :

```
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

Flash Lite 2.0

Exemple

L'exemple suivant enregistre du texte dans un objet partagé appelé `my_so` (pour consulter l'intégralité de l'exemple, consultez la section `SharedObject.getLocal()`) :

```
var my_so:SharedObject = SharedObject.getLocal("savedText");

// myText is an input text field and inputText is a dynamic text field.
myText.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.text == "undefined") {
    myText.text = "";
}

changedListener = new Object();
changedListener.onChanged = function (changedField) {
    my_so.data.myTextSaved = changedField.text;
    inputText.text = "";
    inputText.text = my_so.data.myTextSaved;
}
myText.addListener(changedListener);
```

flush (méthode SharedObject.flush)

```
public flush(minDiskSpace:Number) : Object
```

Ecrit l'objet partagé dans un fichier local et permanent. Pour garantir que l'objet partagé soit écrit sur le périphérique, l'application doit forcer une opération d'écriture en appelant la méthode `flush()`.

Contrairement à Flash Player, l'opération d'écriture est asynchrone et les résultats ne sont pas disponibles immédiatement.

Disponibilité

Flash Lite 2.0

Paramètres

minDiskSpace : Number - Entier spécifiant le nombre d'octets à affecter à cet objet. La valeur par défaut est 0.

Valeur renvoyée

Object - Valeur booléenne (`true` ou `false`) ou valeur de chaîne (`pending`). La méthode `flush()` renvoie `pending` pour la plupart des requêtes, avec les exceptions suivantes :

- S'il n'est pas nécessaire d'écrire des données (si elles ont déjà été écrites), `flush()` renvoie `true`.
- Si le paramètre `minimumDiskSpace` dépasse l'espace maximum disponible pour un fichier SWF ou l'espace restant sur un fichier SWF, ou si une erreur s'est produite pendant le traitement de la requête, `flush()` renvoie `false`.

Si la méthode `flush()` renvoie `pending`, le lecteur Flash Lite affiche une boîte de dialogue demandant à l'utilisateur de libérer de l'espace afin d'accroître l'espace disque disponible pour les objets partagés. Pour disposer d'un espace permettant à l'objet partagé de se développer à l'avenir, ce qui évite le renvoi de valeurs `pending` , transmettez une valeur pour `minimumDiskSpace`. Lorsque le lecteur Flash Lite 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.

Exemple

L'exemple suivant traite les valeurs pouvant être renvoyées par la méthode `flush()` :

```
so_big = SharedObject.getLocal("large");

so_big.data.name = "This is a long string of text.";
so_big.flush();
var flushResult = so_big.flush();

switch (flushResult) {
case 'pending' :
    result.text += "pending";
    break;
case true :
    result.text += "Data was flushed.";
    break;
case false :
    result.text += "Test failed. Data was not flushed.";
    break;
}
```

Voir aussi

`clear (méthode SharedObject.clear)`, `onStatus (gestionnaire SharedObject.onStatus)`

getLocal (méthode SharedObject.getLocal)

```
public static getLocal(name:String) : SharedObject
```

Renvoie une référence à un objet partagé rémanent du périphérique et disponible uniquement pour le client actuel. Si l'objet partagé n'existe pas encore, `getLocal()` en crée un. Il s'agit d'une méthode statique de la classe SharedObject.

Remarque : Dans Flash Lite, un objet partagé ne peut pas être partagé entre deux fichiers SWF.

Pour affecter l'objet à une variable, utilisez une syntaxe du type suivant :

```
var so:SharedObject = SharedObject.getLocal("savedData")
```

Dans la mesure où les données peuvent ne pas être immédiatement disponibles en lecture sur le périphérique, l'application doit créer et enregistrer un écouteur pour l'objet partagé identifié par `name`. Consultez la description de la méthode `addListener()` pour plus de détails.

Disponibilité

Flash Lite 2.0

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 :

```
~ % & \ ; : " ' , < > ? #
```

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 ou rechercher l'objet partagé, `getLocal()` 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 par le périphérique.

Exemple

L'exemple suivant enregistre la dernière image entrée par un utilisateur dans l'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";
}
```

getMaxSize (méthode SharedObject.getMaxSize)

```
public static getMaxSize() : Number
```

Renvoie le nombre total d'octets que le fichier SWF peut utiliser pour stocker des objets mobiles partagés sur le périphérique.

Par exemple, si cette méthode renvoie 1 K, l'animation peut enregistrer un objet partagé de 1 K, ou plusieurs objets partagés de plus petite taille, à condition que la taille totale ne dépasse pas 1 K. Il s'agit d'une méthode statique de la classe SharedObject.

Disponibilité

Flash Lite 2.0

Valeur renvoyée

Number - Valeur numérique spécifiant le nombre total d'octets pouvant être enregistrés sur le périphérique. Il s'agit également de la taille disponible pour tout le contenu chargé dynamiquement à l'aide de `loadMovie()`.

Exemple

L'exemple suivant permet de vérifier si vous disposez de plus de 1 Ko d'espace de stockage réservé avant de créer un objet mobile Flash Lite.

```
if (SharedObject.getMaxSize() > 1024) {
    var my_so:SharedObject = SharedObject.getLocal("sharedObject1");
} else {
    trace("SharedObject's maximum size is less than 1 KB.");
}
```

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.

Si l'objet écouteur partagé n'a pas encore été appelé, `getSize()` renvoie 0. Consultez la méthode `addListener()` pour plus de détails sur l'utilisation de l'écouteur.

Disponibilité

Flash Lite 2.0

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. Pour 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'information a une propriété de code contenant une chaîne qui décrit le résultat du gestionnaire `onStatus` et une propriété `level` contenant la chaîne `"Status"` ou `"Error"`.

En plus de ce gestionnaire, Flash Lite fournit également 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 informent lorsque certaines activités SharedObject ont lieu :

Propriété du code	Propriété de niveau	Signification
`SharedObject.Flush.Failed`	Error	La méthode `SharedObject.flush()` qui a renvoyé `"pending"` (en attente) a échoué (l'utilisateur n'avait pas alloué d'espace disque supplémentaire à l'objet partagé lorsque Flash Lite Player a affiché la boîte de dialogue Paramètres d'enregistrement local).
`SharedObject.Flush.Success`	Status	La méthode `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

Flash Lite 2.0

Paramètres

infoObject : Object - Paramètre défini selon le message de statut.

Exemple

L'exemple suivant affiche différents messages selon que l'utilisateur accepte ou refuse l'écriture de l'occurrence SharedObject sur le disque.

```
this.createTextField("message_txt", this.getNextHighestDepth(), 0, 30, 120, 50);
this.message_txt.wordWrap = true;

this.createTextField("status_txt", this.getNextHighestDepth(), 0, 90, 120, 100);
this.status_txt.wordWrap = 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) {
    for (var i in infoObject) {
    status_txt.text += i+"-"+infoObject[i] +"\n";
    }
};

var flushResult = my_so.flush(1000001);
switch (flushResult) {
    case 'pending' :
    message_txt.text = "flush is pending, waiting on user interaction.";
    break;
    case true :
    message_txt.text = "flush was successful. Requested storage space approved.";
    break;
    case false :
    message_txt.text = "flush failed. User denied request for additional storage.";
    break;
}
```

Voir aussi

onStatus (gestionnaire System.onStatus)

removeListener (méthode SharedObject.removeListener)

```
public static removeListener(objectName:String)
```

Supprime tous les écouteurs qui ont été ajoutés à l'aide de la méthode `addListener()`.

Disponibilité

Flash Lite 2.0

Paramètres

objectName : String - Chaîne indiquant le nom de l'objet partagé.

Sound

```
Object
    |
    +-Sound
```

```
public class Sound
extends Object
```

La classe Sound vous permet de 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 nouvel objet Sound, vous pouvez utiliser les méthodes pour contrôler le son de 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

Flash Lite 2.0

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

constructor (propriété Object.constructor), __proto__ (Object.__proto__, propriété), prototype (propriété Object.prototype), __resolve (Object.__resolve, propriété)

Résumé des événements

Evénement	Description
onID3 = fonction() {}	Appelé dès que de nouvelles données ID3 sont disponibles pour un fichier MP3 que vous chargez à l'aide de Sound.attachSound() ou de Sound.loadSound().
onLoad = function(success:Boolean) {}	Appelé automatiquement lorsqu'un son est en cours de chargement.
onSoundComplete = fonction() {}	Appelé automatiquement lorsque la lecture d'un son est terminée.

Récapitulatif 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:String) : 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() sous la forme d'un 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 au volume maximum.
	loadSound(url:String, isStreaming:Boolean) : Void	Charge un fichier MP3 dans un objet Sound.
	setPan(valeur: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.

Modificateurs	Signature	Description
	setVolume(valeur:Number) : Void	Définit le volume pour l'objet Sound.
	start([secondOffset:Number], [loops:Number]) : Void	Commence à lire le dernier son associé au début si aucun paramètre n'est spécifié ou à partir de 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

Flash Lite 2.0

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 a 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 [read-only]
```

Durée d'un son, en millisecondes.

Remarque : Flash Lite 2.0 prend uniquement en charge cette propriété pour le son Flash natif. Les formats audio propres à un périphérique hôte ne sont pas pris en charge.

Disponibilité
Flash Lite 2.0

Exemple
L'exemple suivant charge un son et affiche la durée du fichier audio dans le panneau Sortie. Ajoutez le code ActionScript suivant à votre fichier FLA ou AS :

```
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 Sortie. Lorsque la musique commence et termine son chargement, des informations sont écrites dans le fichier journal. Ajoutez le code ActionScript suivant à votre fichier FLA ou AS :

```
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+"%";
}
```

Voir aussi

position (propriété Sound.position)

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 chargé d'un son.

Disponibilité

Flash Lite 2.0

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;
}
```

Voir aussi

getBytesTotal (méthode Sound.getBytesTotal)

getBytesTotal (méthode Sound.getBytesTotal)

```
public getBytesTotal() : Number
```

Renvoie la taille, en octets, de l'objet Sound spécifié.

Disponibilité

Flash Lite 2.0

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 aussi

getBytesLoaded (méthode Sound.getBytesLoaded)

getPan (méthode Sound.getPan)

```
public getPan() : Number
```

Renvoie le niveau panoramique défini dans le dernier appel `setPan()` sous la forme d'un 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 à `setTransform()`.

Remarque : Flash Lite 2.0 prend uniquement en charge cette méthode pour le son Flash natif. Les formats audio propres à un périphérique hôte ne sont pas pris en charge.

Disponibilité
Flash Lite 2.0

Valeur renvoyée
Number - Entier.

Exemple
L'exemple suivant crée un champ texte pour afficher la valeur du niveau panoramique pour les sons Flash natifs. L'identifiant de liaison du son est « combo ». Ajoutez le code ActionScript suivant à votre fichier FLA ou AS :

```
this.createTextField("pan_txt", 1, 0, 100, 100, 100);
mix=new Sound();
mix.attachSound("combo");
mix.start();
mix.setPan(-100);
pan_txt.text = mix.getPan(this);
```

Vous pouvez utiliser l'exemple suivant pour diffuser le son de périphérique. Dans la mesure où Flash Lite ne prend pas en charge la diffusion en flux continu, il est judicieux de charger le son avant de le lire.

```
var my_sound:Sound = new Sound();
    my_sound.onLoad = function(success) {
        if (success) {
            my_sound.start();
        } else {
            output.text = "loading failure";
        }
    };
    my_sound.loadSound("song1.mp3",false);
```

Voir aussi
`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()`.

Remarque : Flash Lite 2.0 prend uniquement en charge cette méthode pour le son Flash natif. Les formats audio propres à un périphérique hôte ne sont pas pris en charge.

Disponibilité
Flash Lite 2.0

Valeur renvoyée

Object - Objet doté de propriétés qui contiennent les valeurs de pourcentage des 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>";
}
```

Voir aussi

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 au volume maximum. Le paramètre par défaut est 100.

Disponibilité

Flash Lite 2.0

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 ActionScript 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);

};
```

Voir aussi

setVolume (méthode Sound.setVolume)

id3 (propriété Sound.id3)

```
public id3 : Object [read-only]
```

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
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 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

Flash Lite 2.0

Exemple

L'exemple suivant permet de suivre les propriétés ID3 du fichier 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 aussi

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 fichiers MP3 (événement ou flux continu) chargés avec cette méthode sont enregistrés dans la mémoire cache du navigateur du système de l'utilisateur.

Remarque : Pour Flash Lite 2.0, vous pouvez ignorer le paramètre `isStreaming` dans la mesure où Flash Lite 2.0 traite chaque son comme un événement son.

Disponibilité
Flash Lite 2.0

Paramètres
url : String - Emplacement d'un fichier son MP3 sur un serveur.

isStreaming : Boolean - Valeur booléenne indiquant 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 aussi
`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 de `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
Flash Lite 2.0

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 aussi

attachSound (méthode Sound.attachSound), id3 (propriété Sound.id3), 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 (pour un exemple de chacun de ces fonctions, consultez la section `Sound.onSoundComplete`). Il est recommandé de définir ce gestionnaire avant d'appeler la méthode `mySound.loadSound()`.

Disponibilité

Flash Lite 2.0

Paramètres

success : Boolean - Valeur booléenne `true` si `my_sound` est chargé ou `false` dans le cas contraire.

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);
```

Voir aussi

loadSound (méthode Sound.loadSound)

onSoundComplete (gestionnaire Sound.onSoundComplete)

```
onSoundComplete = function() {}
```

Appelé automatiquement lorsque la lecture d'un son est terminée. 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

Flash Lite 2.0

Exemple

Utilisation 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();
```

Utilisation 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 aussi

onLoad (gestionnaire Sound.onLoad)

position (propriété Sound.position)

```
public position : Number [read-only]
```

La durée de diffusion du son exprimée en millisecondes. Si le son est lu en boucle, la position est réinitialisée à 0 au début de chaque boucle.

Remarque : Flash Lite 2.0 prend uniquement en charge cette propriété pour le son Flash natif. Les formats audio propres à un périphérique hôte ne sont pas pris en charge.

Disponibilité
Flash Lite 2.0

Exemple
Consultez `Sound.duration` pour obtenir un exemple d'utilisation de cette méthode.

Voir aussi
`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) de lecture du son.

Remarque : Flash Lite 2.0 prend uniquement en charge cette méthode pour le son Flash natif. Les formats audio propres à un périphérique hôte ne sont pas pris en charge.

Disponibilité
Flash Lite 2.0

Paramètres
valeur : 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 la section `Sound.getBytesLoaded()` pour obtenir un exemple d'utilisation de cette méthode.

Voir aussi
`attachSound (méthode Sound.attachSound)`, `getPan (méthode Sound.getPan)`, `setTransform (méthode Sound.setTransform)` `setVolume (méthode Sound.setVolume)`, `start (méthode Sound.start)`

setTransform (méthode Sound.setTransform)

```
public setTransform(transformObject:Object) : Void
```

Définit les informations de transformation du son (ou balance), pour un objet Sound.

Le paramètre `soundTransformObject` est un objet que vous créez à l'aide de la méthode de constructeur de la classe Object générique avec des paramètres spécifiant la répartition du son entre 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 stéréo comme des sons mono et ajouter des effets intéressants aux sons.

Remarque : Flash Lite 2.0 prend uniquement en charge cette méthode pour le son Flash natif. Les formats audio propres à un périphérique hôte ne sont pas pris en charge.

Les propriétés du paramètre `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 de `left_input` ou `right_input` sont déterminées par le type (stéréo ou mono) du son de votre fichier SWF.

Les sons stéréo 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 mono 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
Flash Lite 2.0

Paramètres
transformObject : Object - Objet créé avec le constructeur de 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 dernières sont combinées.

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 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};
```

Voir aussi

Object, getTransform (méthode Sound.getTransform)

setVolume (méthode Sound.setVolume)

```
public setVolume(value:Number) : Void
```

Définit le volume pour l'objet Sound.

Disponibilité

Flash Lite 2.0

Paramètres

valeur : 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 la section `Sound.getVolume()` pour obtenir un exemple d'utilisation de cette méthode.

Voir aussi

setPan (méthode Sound.setPan), setTransform (méthode Sound.setTransform)

Constructeur Sound

```
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
Flash Lite 2.0

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 les sons présents dans *my_mc*.

```
var movie_sound:Sound = new Sound(my_mc);
movie_sound.start();
```

start (méthode Sound.start)

```
public start([secondOffset:Number], [loops:Number]) : Void
```

Commence à lire le dernier son associé au début si aucun paramètre n'est spécifié ou à partir de l'endroit spécifié par le paramètre `secondOffset`.

Disponibilité
Flash Lite 2.0

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 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 a recours à la méthode `start()` pour commencer sa diffusion. 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);
```

Voir aussi

stop (méthode Sound.stop)

stop (méthode Sound.stop)

```
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

Flash Lite 2.0

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 aussi

start (méthode Sound.start)

Stage

```
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. Les méthodes et propriétés de cette classe permettent d'accéder aux informations et de les modifier dans les limites d'un fichier SWF.

Disponibilité

Flash Lite 2.0

Résumé des propriétés

Modificateurs	Propriété	Description
statique	align : String	Indique l'alignement actuel du fichier SWF dans le lecteur ou le navigateur.
statique	height : Number	Propriété (lecture seule) : indique la hauteur actuelle, en pixels, de la scène.
statique	scaleMode : String	Indique le redimensionnement actuel du fichier SWF dans Flash Lite Player.
statique	width : Number	Propriété (lecture seule) : indique la largeur actuelle, en pixels, de la scène.

Propriétés héritées de la classe Object

constructor (propriété Object.constructor), __proto__ (Object.__proto__, propriété), prototype (propriété Object.prototype), __resolve (Object.__resolve, propriété)

Résumé des événements

Evénement	Description
onResize = fonction() {}	Appelé lorsque Stage.scaleMode est défini sur noScale et que le fichier SWF est redimensionné.

Résumé de la méthode

Modificateurs	Signature	Description
statique	addListener(listener: Object) : Void	Détecte le moment où un fichier SWF est redimensionné (mais uniquement si Stage.scaleMode = "noScale").
statique	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 par défaut des clips (`showAll`) ou d'autres réglages de redimensionnement (`exactFit` et `noBorder`).

Pour utiliser `addListener()`, vous devez d'abord créer un *objet écouteur*. Les objets écouteurs de Stage reçoivent une notification de `Stage.onResize`.

Disponibilité

Flash Lite 2.0

Paramètres

listener : Object - Objet chargé de détecter la notification de rappel provenant du gestionnaire d'événements `Stage.onResize`.

Exemple

Cet exemple crée un nouvel objet écouteur appelé `stageListener`. Il utilise ensuite `stageListener` pour appeler `onResize` et définir une fonction qui sera appelée au déclenchement de `onResize`. 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 aussi

onResize (écouteur d'événements Stage.onResize),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 de 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
Flash Lite 2.0

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 Lite Player. Lorsque la valeur de `Stage.scaleMode` n'est pas `noScale`, height représente la hauteur du fichier SWF.

Disponibilité
Flash Lite 2.0

Exemple
Cet exemple crée un nouvel objet écouteur appelé `stageListener`. Il utilise ensuite `myListener` pour appeler `onResize` et définir une fonction qui sera appelée au déclenchement de `onResize`. 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 aussi

align (propriété Stage.align), scaleMode (propriété Stage.scaleMode), width (propriété Stage.width)

onResize (écouteur d'événements Stage.onResize)

```
onResize = function() {}
```

Appelé lorsque `Stage.scaleMode` est défini sur noScale et que 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.

Disponibilité

Flash Lite 2.0

Exemple

L'exemple suivant affiche un message dans le panneau 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 aussi

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

Flash Lite 2.0

Paramètres

listener : Object - Objet ajouté à la liste de rappel d'un 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 > Tester 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 aussi

`addListener (méthode Stage.addListener)`

scaleMode (propriété Stage.scaleMode)

```
public static scaleMode : String
```

Indique le redimensionnement actuel du fichier SWF dans Flash Lite Player. La propriété `scaleMode` oblige le fichier SWF à passer dans 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

Flash Lite 2.0

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);
```

Vous trouverez un autre exemple dans le fichier stagesize.fla du dossier d'exemples ActionScript, disponible à l'adresse www.adobe.com/go/learn_flt_samples_and_tutorials_fr. Téléchargez le fichier .zip de Samples_and_Tutorials pour votre version de Flash Lite, puis décompressez-le pour afficher le dossier ActionScript et consulter l'exemple.

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 Lite 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 à la 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

Flash Lite 2.0

Exemple

Cet exemple crée un nouvel objet écouteur appelé `stageListener`. Il utilise ensuite `stageListener` pour appeler `onResize` et définir une fonction qui sera appelée au déclenchement de `onResize`. 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 aussi

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'interprète d'ActionScript le convertit automatiquement en un objet String temporaire, appelle la méthode, puis supprime l'objet String temporaire. Vous pouvez aussi utiliser la propriété `String.length` avec une chaîne littérale.

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

Flash Lite 2.0

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

constructor (propriété Object.constructor), __proto__ (Object.__proto__, propriété), prototype (propriété Object.prototype), __resolve (Object.__resolve, propriété)

Récapitulatif des constructeurs

Signature	Description
String(valeur: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, compris entre 0 et 65535, qui représente le caractère spécifié par index.
	concat(valeur: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.
statique	fromCharCode() : String	Renvoie une chaîne comprenant les caractères représentés par les valeurs Unicode dans les paramètres.
	indexOf(valeur: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(valeur: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.
	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
Flash Lite 2.0

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 aussi
`charCodeAt (méthode String.charCodeAt)`

charCodeAt (méthode String.charCodeAt)

```
public charCodeAt(index:Number) : Number
```

Renvoie un entier 16 bits, compris entre 0 et 65535, qui représente le caractère spécifié par `index`. Si `index` n'est pas un nombre compris entre 0 et `string.length` - 1, `NaN` est renvoyée.

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
Flash Lite 2.0

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 aussi

charAt (méthode String.charAt)

concat (méthode String.concat)

```
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

Flash Lite 2.0

Paramètres

valeur : 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

Flash Lite 2.0

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 en base 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
Flash Lite 2.0

Paramètres
valeur : 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 aussi
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 en base 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
Flash Lite 2.0

Paramètres

valeur : 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 aussi

indexOf (méthode String.indexOf)

length (propriété String.length)

```
public length : 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

Flash Lite 2.0

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` s'affiche dans le panneau Sortie. En l'absence du caractère `@`, le programme affiche `false` dans le panneau 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 d'exemples ActionScript disponible à l'adresse www.adobe.com/go/learn_fl_samples_fr. Téléchargez le fichier .zip, puis décompressez-le pour afficher le dossier correspondant à votre version d'ActionScript et consulter l'exemple.

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` ; s'il trouve à droite de ce caractère, la méthode renvoie une chaîne vide.

Disponibilité

Flash Lite 2.0

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 la variable `my_str`, lui affecte une valeur String, puis appelle la méthode `slice()` à l'aide de différentes valeurs pour les paramètres `start` et `end`. Tout appel à `slice()` est recensé dans une instruction `trace()` qui affiche son résultat dans le panneau 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 d'exemples ActionScript disponible à l'adresse www.adobe.com/go/learn_fl_samples_fr. Téléchargez le fichier .zip, puis décompressez-le pour afficher le dossier correspondant à votre version d'ActionScript et consulter l'exemple.

Voir aussi

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

Flash Lite 2.0

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 d'exemples ActionScript disponible à l'adresse www.adobe.com/go/learn_fl_samples_fr. Téléchargez le fichier .zip, puis décompressez-le pour afficher le dossier correspondant à votre version d'ActionScript et consulter l'exemple.

Voir aussi

join (méthode Array.join)

Constructeur String

```
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

Flash Lite 2.0

Paramètres
valeur : 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
Flash Lite 2.0

Paramètres
start : Number - Entier qui indique la position du premier caractère de `my_str` à utiliser pour créer la sous-chaîne. 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 la chaîne `my_str` et utilise `substr()` pour renvoyer le second mot de la chaîne, tout d'abord en utilisant 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 d'exemples ActionScript disponible à l'adresse www.adobe.com/go/learn_fl_samples_fr. Téléchargez le fichier .zip, puis décompressez-le pour afficher le dossier correspondant à votre version d'ActionScript et consulter l'exemple.

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`, les paramètres sont automatiquement permutés avant que la fonction s'exécute et la valeur d'origine n'est pas modifiée.

Disponibilité
Flash Lite 2.0

Paramètres

start : Number - Entier qui indique la position du premier caractère de `my_str` utilisé pour créer la sous-chaîne. Les valeurs admissibles pour `start` vont de 0 à `String.length` - 1. Si `start` a une valeur négatif, 0 est utilisé.

end : Number - Entier qui correspond à 1+ l'index du dernier caractère de `my_str` à extraire. Les valeurs admissibles 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 d'exemples ActionScript disponible à l'adresse www.adobe.com/go/learn_fl_samples_fr. Téléchargez le fichier .zip, puis décompressez-le pour afficher le dossier correspondant à votre version d'ActionScript et consulter l'exemple.

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

Flash Lite 2.0

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 d'exemples ActionScript disponible à l'adresse Page d'exemples Adobe Flash. Téléchargez le fichier .zip, puis décompressez-le pour afficher le dossier correspondant à votre version d'ActionScript et consulter l'exemple.

Voir aussi

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

Flash Lite 2.0

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` ». En incluant la méthode `toString()`, la sortie voulue est obtenue : "`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

Flash Lite 2.0

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 disponible à l'adresse Page d'exemples Adobe Flash. Téléchargez le fichier .zip, puis décompressez-le pour afficher le dossier correspondant à votre version d'ActionScript et consulter l'exemple.

Voir aussi

toLowerCase (méthode String.toLowerCase)

valueOf (méthode String.valueOf)

```
public valueOf() : String
```

Renvoie une chaîne.

Disponibilité

Flash Lite 2.0

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
```

System

```
Object
    |
    +-System
```

```
public class System
extends Object
```

La classe System contient des propriétés liés à des opérations qui s'effectue sur l'ordinateur de l'utilisateur, telles que les opérations portant sur les objets partagés et le presse-papiers. Des propriétés et des méthodes supplémentaires figurent dans des classes spécifiques dans le package System : les classes capabilities (voir System.capabilities) et security (voir System.security).

Disponibilité

Flash Lite 2.0

Voir aussi

capabilities (System.capabilities), Security (System.security)

Résumé des propriétés

Modificateurs	Propriété	Description
statique	useCodepage : Boolean	Valeur booléenne qui indique à Flash Lite 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

constructor (propriété Object.constructor), __proto__ (Object.__proto__, propriété), prototype (propriété Object.prototype), __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

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)

onStatus (gestionnaire System.onStatus)

```
onStatus = function(infoObject:Object) {}
```

Gestionnaire d'événements : fournit un gestionnaire de super événements pour certains objets.

La classe SharedObject comprend un gestionnaire d'événements `onStatus` qui utilise un objet d'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.

Outre la méthode `SharedObject.onStatus()`, Flash fournit également une super fonction appelée `System.onStatus()`, qui sert de gestionnaire de messages d'erreur secondaire. Si une occurrence de la classe SharedObject transmet un objet d'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()`.

Disponibilité

Flash Lite 2.0

Paramètres

infoObject : Object - Paramètre défini selon le message de statut.

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.");
}
```

Voir aussi

`onStatus (gestionnaire SharedObject.onStatus)`

useCodepage (propriété System.useCodepage)

```
public static useCodepage : Boolean
```

Valeur booléenne qui indique à Flash Lite 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 Lite 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 Lite 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()`, ou de la classe LoadVars ou XML) doit être codé en Unicode lorsque vous enregistrez le fichier texte afin que Flash Lite 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 Lite Player interprète du texte externe à l'aide de la page de code classique du système d'exploitation exécutant Flash Lite 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

Flash Lite 2.0

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 dynamiques sont des occurrences de la classe TextField. Vous pouvez donner un nom d'occurrence à un champ texte dans l'inspecteur 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 un champ texte de façon dynamique, appelez `MovieClip.createTextField()`.

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.

Disponibilité

Flash Lite 2.0

Voir aussi

`Object`, `createTextField (méthode MovieClip.createTextField)`

Résumé des propriétés

Modificateurs	Propriété	Description
	_alpha : Number	Définit ou extrait la valeur de transparence alpha du champ texte.
	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	Valeur booléenne qui indique si le rendu du texte doit se faire à l'aide des polices vectorielles intégrées.
	_height : Number	Hauteur du champ texte, en pixels.

Modificateurs	Propriété	Description
	_highquality : Number	Déconseillé depuis Flash Player 7. Cette propriété est déconseillée au profit de `TextField._quality`. Spécifie le niveau d'anti-aliasing appliqué au fichier SWF actuel.
	hscroll : Number	Indique la position de défilement horizontal actuelle.
	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`.
	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	Propriété (globale) ; définit ou récupère la qualité de rendu utilisée pour un fichier SWF.
	_rotation : Number	Rotation du champ texte, en degrés, à partir de son orientation d'origine.
	scroll : Number	Définit la position verticale du texte dans un champ texte.
	selectable : Boolean	Une valeur booléenne qui indique si le champ texte peut être sélectionné.
	_soundbuftime : Number	Spécifie le nombre de secondes pendant lequel les sons sont chargés en mémoire tampon avant d'être diffusés en continu.
	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.
	type : String	Spécifie le type du champ texte.

Modificateurs	Propriété	Description
	_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.
	_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

```
constructor (propriété Object.constructor), __proto__ (Object.__proto__, propriété),
prototype (propriété Object.prototype), __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.
	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.

Méthodes héritées de la classe Object

addProperty (méthode Object.addProperty), hasOwnProperty (méthode Object.hasOwnProperty), isPropertyEnumerable (méthode Object.isPropertyEnumerable) vis PrototypeOf (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`. Ce paramètre est ensuite repris par une instruction `trace` pour envoyer le nom d'occurrence du champ texte au panneau de sortie. Ce paramètre est ensuite repris par une instruction `trace()` pour envoyer le nom d'occurrence du champ texte au fichier journal.

```
my_txt.onScroller = function(textfield_txt:TextField) {
    trace(textfield_txt._name+" scrolled");
};
```

Disponibilité

Flash Lite 2.0

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 un gestionnaire `onChanged` pour cet objet. Ce gestionnaire sera appelé lorsque le champ texte `my_txt` est modifié. La dernière ligne 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_(xt: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);
```

Voir aussi

`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 possibles 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.

Remarque : Cette propriété n'est pas prise en charge pour l'arabe, l'hébreu et le thaïlandais.

Disponibilité
Flash Lite 2.0

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. Définissez ensuite la liaison de la police sur `my font`. Définissez la liaison d'un symbole de police sur `my font`. Ajoutez le code ActionScript suivant à votre fichier AS ou FLA :

```
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;
```

Voir aussi
`_alpha (Button._alpha, propriété)`, `_alpha (MovieClip._alpha, propriété)`

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` 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 de 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" ou "\r"`), le bas est alors également redimensionné en fonction de 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 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" ou "\r"`), le bas est alors également redimensionné en fonction de 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" ou "\r"`), le bas est alors également redimensionné en fonction de 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
Flash Lite 2.0

Exemple
Utilisez le code suivant et entrez différentes valeurs de `autoSize` pour 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 de `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
Flash Lite 2.0

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);
```

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
Flash Lite 2.0

Exemple
Consultez l'exemple de `TextField.background`.

Voir aussi
`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
Flash Lite 2.0

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";
```

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

Flash Lite 2.0

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, 100, 100);
my_txt.border = true;
my_txt.borderColor = 0x00FF00;
my_txt.text = "Lorum ipsum";
```

Voir aussi

border (propriété TextField.border)

bottomScroll (propriété TextField.bottomScroll)

```
public bottomScroll : Number [read-only]
```

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 en base 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

Flash Lite 2.0

Exemple

L'exemple suivant crée un champ texte et y insère du texte. Créez un bouton avec le nom d'occurrence `my_btn`. Lorsque vous cliquez dessus, les propriétés `scroll` et `bottomScroll` du champ texte s'affichent dans 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 (pposite end of the color system, is (#FFFFFF).";
my_btn.onRelease = function() {
    trace("scroll: "+comment_txt.scroll);
    trace("bottomScroll: "+comment_txt.bottomScroll);
};
```

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

Flash Lite 2.0

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 AS ou FLA :

```
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;
```

Voir aussi

html (propriété TextField.html)

embedFonts (propriété TextField.embedFonts)

```
public embedFonts : Boolean
```

Valeur booléenne qui indique si le rendu du texte doit se faire à l'aide des polices vectorielles intégrées. Si la valeur est `true`, Flash Lite procède au rendu du champ texte à l'aide des polices vectorielles intégrées. Si la valeur est `false`, Flash Lite procède au rendu du champ texte à l'aide des 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 l'identifiant de liaison correspondant), le texte ne s'affiche pas.

Remarque : Cette propriété n'est pas prise en charge pour l'arabe, l'hébreu et le thaïlandais.

Disponibilité
Flash Lite 2.0

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 chaîne `my font` renvoie à un symbole de police dans la bibliothèque, avec l'identifiant de liaison `my font`. L'exemple suivant suppose que vous disposez d'un symbole de police 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;
```

getDepth (méthode TextField.getDepth)

```
public getDepth() : Number
```

Renvoie la profondeur d'un champ texte.

Disponibilité
Flash Lite 2.0

Valeur renvoyée
Number - Entier.

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 ActionScript, qui 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());
    }
}
```

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 entré par un utilisateur. Lorsque `getNewTextFormat()` est appelé, toutes les propriétés de l'objet TextFormat renvoyé sont définies. Aucune propriété n'a une valeur `null`.

Disponibilité
Flash Lite 2.0

Valeur renvoyée
TextFormat - Objet TextFormat.

Exemple
L'exemple suivant affiche l'objet TextFormat (`my_txt`) du champ texte spécifié.

```
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]);
}
```

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.

- **Utilisation 1 :** `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 *mixte*, c'est-à-dire ayant différentes valeurs à différents endroits du texte, a la valeur `null`..

- **Utilisation 2 :** `my_textField.getTextFormat(beginIndex:Number)`

  Renvoie un objet TextFormat contenant une copie du format de texte du champ texte à la position `beginIndex`.

- **Utilisation 3 :** `my_(extField.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
Flash Lite 2.0

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 pas `endIndex`, l'objet TextFormat est pour le seul caractère spécifié par `beginIndex`.

Valeur renvoyée
TextFormat - Un objet.

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]);
}
```

Voir aussi
getNewTextFormat (méthode TextField.getNewTextFormat), setNewTextFormat (méthode TextField.setNewTextFormat) setTextFormat (méthode TextField.setTextFormat)

_height (propriété TextField._height)

```
public _height : Number
```

Hauteur du champ texte, en pixels.

Disponibilité
Flash Lite 2.0

Exemple
L'exemple de code suivant définit la hauteur et la largeur d'un champ texte :

```
my_txt._width = 200;
my_txt._height = 200;
```

_highquality (propriété TextField._highquality)

```
public _highquality : Number
```

Déconseillé depuis Flash Player 7. Cette propriété est déconseillée au profit de `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
Flash Lite 2.0

Voir aussi
_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, et donc 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 est en base zéro et non en base un, comme c'est le cas de la propriété de défilement vertical `TextField.scroll`.

Disponibilité

Flash Lite 2.0

Exemple

L'exemple suivant fait défiler le champ texte `my_txt` horizontalement à l'aide de deux boutons appelés `scrollLeft_btn` et `scrollRight_btn`. Le montant de défilement s'affiche dans un champ texte appelé `TextField.scroll`. Ajoutez le code ActionScript suivant à votre fichier AS ou FLA :

```
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;
};
```

Voir aussi

`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 a un autre format.

Disponibilité

Flash Lite 2.0

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>";
```

Voir aussi

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 `text`. Vous pouvez indiquer qu'un champ texte est un champ texte HTML dans l'inspecteur Propriétés ou en définissant la propriété `html` du champ texte sur `true`.

Disponibilité

Flash Lite 2.0

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 = "< this is bold text >";
```

Voir aussi

html (propriété TextField.html)

length (propriété TextField.length)

```
public length : Number [read-only]
```

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 qu'une tabulation (`\t`) compte comme un seul caractère.

Disponibilité

Flash Lite 2.0

Exemple

L'exemple suivant renvoie le nombre de caractères du champ texte `date_txt`, 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);
```

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 la propriété `maxChars`, qui indique seulement la quantité de texte qu'un utilisateur peut saisir. Si la valeur de cette propriété est `null`, la quantité de texte qu'un utilisateur peut entrer est illimitée.

Disponibilité
Flash Lite 2.0

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;
```

maxhscroll (propriété TextField.maxhscroll)

```
public maxhscroll : Number [read-only]
```

Indique la valeur maximale de `TextField.hscroll`.

Disponibilité
Flash Lite 2.0

Exemple
Consultez l'exemple de `TextField.hscroll`.

maxscroll (propriété TextField.maxscroll)

```
public maxscroll : Number [read-only]
```

Indique la valeur maximale de `TextField.scroll`.

Disponibilité
Flash Lite 2.0

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 ActionScript ou FLA.

```
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;
};
```

multiline (propriété TextField.multiline)

```
public multiline : Boolean
```

Indique si le champ texte est un champ texte multiligne. Si la valeur est `true`, le champ de texte est multiligne ; si la valeur est `false`, le champ de texte est un champ de texte sur une seule ligne.

Disponibilité

Flash Lite 2.0

Exemple

L'exemple suivant crée un champ texte multiligne appelé `myText`.

```
this.createTextField("myText", this.getNextHighestDepth(), 10, 30, 110, 100);
myText.text = "Flash is an authoring tool that designers and developers use to create
presentations,
applications, and other content that enables user interaction.";
myText.border = true;
myText.wordWrap = true;
myText.multiline = true;
```

_name (propriété TextField._name)

```
public _name : String
```

Le nom de l'occurrence du champ texte.

Disponibilité

Flash Lite 2.0

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 ActionScript, 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 d'occurrence et la profondeur s'affichent dans le panneau Sortie. Lorsque vous testez le document, le nom d'occurrence et la profondeur s'affichent dans le fichier journal.

onChanged (gestionnaire TextField.onChanged)

```
onChanged = function(changedField:TextField) {}
```

Appelé lorsque le contenu d'un champ texte change. Par défaut, il a la valeur undefined ; vous pouvez le définir dans un script.

Une référence à 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 au panneau 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

Flash Lite 2.0

Paramètres

changedField : TextField - Champ déclenchant l'événement.

Voir aussi

getNewTextFormat (méthode TextField.getNewTextFormat), 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` qui représente le nouvel objet recevant le focus. Si aucun objet ne reçoit le focus, `newFocus` contient la valeur `null`.

Disponibilité
Flash Lite 2.0

Paramètres
newFocus : Object - Objet recevant 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 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 aussi
onSetFocus (gestionnaire TextField.onSetFocus)

onScroller (gestionnaire TextField.onScroller)

```
onScroller = function(scrolledField:TextField) {}
```

Appelé lorsque l'une des propriétés de défilement du champ texte est modifiée.

Une référence à 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 au panneau 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 une telle interaction de utilisateur. C'est ce pourquoi est utilisé `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
Flash Lite 2.0

Paramètres
scrolledField : TextField - Référence 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 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 ActionScript :

```
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;
};
```

Voir aussi
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 de tabulation 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, le paramètre `oldFocus` contient une valeur null.

Disponibilité
Flash Lite 2.0

Paramètres
oldFocus : Object - Objet perdant le focus.

Exemple
Consultez l'exemple de `TextField.onKillFocus`.

Voir aussi
`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 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 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
Flash Lite 2.0

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 Sortie :

```
first_txt'( _parent is: _level0
second_txt's _parent is: _level0.holder_mc
```

Voir aussi
`_parent (propriété Button._parent)`, `_parent (propriété MovieClip._parent)`, `_root, propriété`

password (propriété TextField.password)

```
public password : Boolean
```

Indique si le champ texte est un champ texte de mot de passe. Si la valeur du mot de passe est `true`, le champ texte est un champ de mot de passe : dès que l'utilisateur a terminé de saisir son mot de passe et clique sur OK, le champ texte remplace les caractères saisis par des astérisques. 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

Flash Lite 2.0

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`. Dès que l'utilisateur clique sur OK pour terminer la saisie de son mot de passe, les caractères du mot de passe sont remplacés par des 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_tx(.password = true;
password_txt.text = "world";
```

_quality (propriété TextField._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`.

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 `_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. 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 lissés 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 : Cette propriété n'est pas prise en charge pour l'arabe, l'hébreu et le thaïlandais.

Disponibilité
Flash Lite 2.0

Exemple
L'exemple suivant définit la qualité de rendu sur `LOW` :

```
my_txt._quality = "LOW";
```

Voir aussi
`_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
Flash Lite 2.0

Paramètres
listener : Object - Objet qui ne reçoit plus de notifications de `TextField.onChanged` ou `TextField.onScroller`.

Valeur renvoyée
Boolean - Si l'objet `listener` a été supprimé, la méthode renvoie une valeur `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 TextField), la méthode renvoie la valeur `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 Sortie. Lorsque l'utilisateur tape du texte dans le champ, les informations sur le nombre de caractères du champ texte sont écrites dans le fichier journal. Si l'utilisateur clique sur l'occurrence `removeListener_btn`, l'écouteur est supprimé et les informations ne sont plus affichées. Si l'utilisateur clique sur l'occurrence `removeListener_btn`, l'écouteur est supprimé et les informations ne sont plus consignées dans le fichier journal.

```
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");
    }
};
```

(

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

Flash Lite 2.0

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 ActionScript.

```
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();
};
```

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 des paragraphes du reste du texte.

Remarque : Vous devez utiliser la méthode `Selection.setFocus()` pour donner le focus au champ avant d'appeler la méthode `replaceSel()`.

Disponibilité
Flash Lite 2.0

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 `Entrer la date courante` 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;
```

Voir aussi
`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
Flash Lite 2.0

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 s'affiche dans le panneau Sortie. Si le symbole n'est pas trouvé, un message d'erreur est écrit dans le fichier journal.

```
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.");
}
```

_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, l'instruction `my_txt._rotation = 450` est identique à `my_txt._rotation = 90`.

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.

Remarque : Cette propriété n'est pas prise en charge pour l'arabe, l'hébreu et le thaïlandais.

Disponibilité

Flash Lite 2.0

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 chaîne `my font` renvoie à un symbole de police dans la bibliothèque, avec l'identifiant de liaison `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.

Voir aussi

_rotation (propriété Button._rotation), _rotation (propriété MovieClip._rotation), getNewTextFormat (méthode TextField.getNewTextFormat)

scroll (propriété TextField.scroll)

```
public scroll : Number
```

Définit 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 car la plupart des polices généralement utilisées sont espacées proportionnellement. En d'autres termes, 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

Flash Lite 2.0

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 ActionScript ou FLA.

```
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;
};
```

Voir aussi

hscroll (propriété TextField.hscroll), maxscroll (propriété TextField.maxscroll)

selectable (propriété TextField.selectable)

```
public selectable : Boolean
```

Une valeur booléenne qui indique si le champ texte peut être sélectionné. Lorsque sa valeur est `true`, le texte peut être sélectionné. La propriété `selectable` détermine si un champ de texte peut être sélectionné et non pas s'il peut être modifié. Un champ texte dynamique peut être sélectionné, même s'il ne peut pas être modifié. Lorsqu'un champ texte n'est pas sélectionnable, 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 Copier. 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 Copier.

Remarque : Cette propriété n'est pas prise en charge pour l'arabe, l'hébreu et le thaïlandais.

Disponibilité
Flash Lite 2.0

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();
}
```

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 saisi par un utilisateur. Lorsque du texte est inséré, le nouveau format texte par défaut lui est attribué.

Le nouveau format texte par défaut est spécifié par `textFormat`, qui est un objet TextFormat.

Disponibilité
Flash Lite 2.0

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:TextForma( = 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 aussi

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 forme spécifiée par le paramètre `textFormat` à tout ou partie du texte du champ. `textFormat` doit être un objet 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 des paragraphes. Chaque caractère d'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 identifier 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 paragraphes.

La méthode `setTextFormat()` modifie la mise en forme de texte appliquée à chaque caractère, à une plage de caractères ou à l'ensemble du contenu d'un champ texte.

- **Utilisation 1 :** `my_textField.setTextFormat(textFormat:TextFormat)`

  Applique les propriétés de `textFormat` à l'ensemble du texte dans le champ texte.

- **Utilisation 2 :** `my_textField.setTextFormat(beginIndex:Number, textFormat:TextFormat)`

  Applique les propriétés de `textFormat` au caractère situé à la position `beginIndex`.

- **Utilisation 3 :** `my_textField.setTextFormat(beginIndex:Number, endIndex:Number, textFormat:TextFormat)`

  Applique les propriétés du paramètre `textFormat` au texte situé entre les positions `beginIndex` et `endIndex`.

Notez que tout texte inséré manuellement par l'utilisateur 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, utilisez `TextField.setNewTextFormat()`.

Disponibilité

Flash Lite 2.0

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 situé après la plage de texte souhaitée. Si vous spécifiez `beginIndex` mais pas `endIndex`, l'objet 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);
```

Voir aussi

getNewTextFormat (méthode TextField.getNewTextFormat), setNewTextFormat (méthode TextField.setNewTextFormat)

_soundbuftime (propriété TextField._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 TextField, il s'agit en fait d'une propriété globale qui s'applique à l'ensemble des sons chargés. Il vous suffit donc de définir sa valeur sur `_soundbuftime`. La définition de cette propriété pour un objet TextField définit en réalité la propriété globale. Pour plus d'informations et un exemple, consultez la section `_soundbuftime`.

Disponibilité

Flash Lite 2.0

Voir aussi

_soundbuftime, propriété

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 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

Flash Lite 2.0

Exemple

L'exemple suivant crée plusieurs champs texte appelés `one_txt`, `two_txt`, `three_txt` et `four_txt`. Le champ texte `three_txt` a sa propriété `tabEnabled` définie sur `false`, ce qui l'exclut de l'ordre de tabulation automatique.

```
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;";
```

Voir aussi

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, un clip ou une occurrence de champ texte ; sa valeur par défaut est `undefined`.

Si un objet affiché du fichier SWF contient une propriété `tabIndex`, l'ordre de tabulation automatique est désactivé et calculé à partir des propriétés `tabIndex` des objets du fichier SWF. L'ordre de tabulation personnalisé n'inclut que 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 ayant une valeur `tabIndex` de `1` précède un objet avec une valeur `tabIndex` de 2. Si deux objets ont la même valeur de `tabIndex`, celui qui précède l'autre dans l'ordre de tabulation est `undefined`.

L'ordre de tabulation personnalisé défini par la propriété `tabIndex` est *flat*. 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
Flash Lite 2.0

Exemple
L'exemple de code ActionScript suivant crée de façon dynamique quatre champs texte et leur affecte un ordre de tabulation personnalisé. Ajoutez le code ActionScript suivant à votre fichier AS ou FLA :

```
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;
```

Voir aussi
tabIndex (propriété Button.tabIndex), tabIndex (propriété MovieClip.tabIndex)

_target (propriété TextField._target)

```
public _target : String [read-only]
```

Le chemin cible de l'occurrence du champ texte. La cible `_self` spécifie l'image actuelle dans la fenêtre actuelle ; `_blank`, une nouvelle fenêtre ; `_parent`, le parent de l'image actuelle ; et `_top`, l'image de premier niveau dans la fenêtre actuelle.

Disponibilité
Flash Lite 2.0

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
```

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

Flash Lite 2.0

Exemple

L'exemple suivant crée un champ texte HTML appelé `my_txt` et lui affecte une chaîne de texte au format HTML. Lorsque vous effectuez le suivi de la propriété `htmlText`, le panneau Sortie affiche la chaîne au format HTML. Lorsque vous effectuez le suivi (trace) de la valeur de la propriété `text`, la chaîne non formatée avec balises HTML s'affiche dans le panneau de sortie. Lorsque vous effectuez le suivi (trace) de la valeur de la propriété `text`, la chaîne non formatée avec balises HTML est écrite dans le fichier journal.

```
this.createTextField("my_txt", this.getNextHighestDepth(), 10, 10, 400, 22);
my_txt.html = true;
my_txt.htmlText = "<b>Remember to always update the help panel.</b>";

trace("htmlText: "+my_txt.htmlText);
trace("text: "+my_txt.text);

// output:
htmlText: <P ALIGN="LEFT"><FONT FACE="Times New Roman" SIZE="12" COLOR="#000000">
<B>Remember to always update your help panel.</B></FONT></P>
text: Remember to always update your help panel.
```

Voir aussi

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 six chiffres pour représenter les valeurs des couleurs. Chaque chiffre comporte seize valeurs ou caractères possibles. Les caractères de 0 à 9 et de A à F sont utilisés. Le noir est représenté par (`#000000`) et le blanc, à l'opposé du spectre de couleurs, est (`#FFFFFF`).

Disponibilité

Flash Lite 2.0

Exemple

Le code ActionScript suivant crée un champ texte et lui 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;
```

textHeight (propriété TextField.textHeight)

```
public textHeight : Number
```

Indique la hauteur du texte.

Disponibilité

Flash Lite 2.0

Exemple

L'exemple suivant crée un champ texte et lui attribue une chaîne de texte. Une instruction trace permet d'afficher la hauteur et la largeur du texte dans le panneau Sortie. La méthode `trace()` permet d'écrire la hauteur et la largeur du texte dans le fichier journal. La propriété `autoSize` est ensuite utilisée pour redimensionner le champ texte et les nouvelles hauteur et largeur s'affichent également dans le panneau Sortie. La propriété `autoSize` est ensuite utilisée pour redimensionner le champ texte et les nouvelles hauteur et largeur sont également écrites dans le fichier journal.

```
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 aussi

textWidth (propriété TextField.textWidth)

textWidth (propriété TextField.textWidth)

```
public textWidth : Number
```

Indique la largeur du texte.

Disponibilité

Flash Lite 2.0

Exemple

Consultez l'exemple de `TextField.textHeight`.

Voir aussi

textHeight (propriété TextField.textHeight)

type (propriété TextField.type)

```
public type : String
```

Spécifie le type du 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
Flash Lite 2.0

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 `password` de `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";
```

_url (propriété TextField._url)

```
public _url : String [read-only]
```

Récupère l'URL du fichier SWF qui a créé le champ texte.

Disponibilité
Flash Lite 2.0

Exemple
L'exemple suivant extrait l'URL du fichier SWF qui a créé le champ texte et un fichier SWF qui s'y charge.

```
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 testé et le fichier appelé best_flash_ever.swf s'affichent dans le panneau Sortie. Lorsque vous testez cet exemple, l'URL du fichier SWF testé et le fichier appelé best_flash_ever.swf sont écrits dans le fichier journal.

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
Flash Lite 2.0

Exemple
L'exemple suivant crée un champ texte appelé `my_txt` et lui associe la variable `today_date`. Lorsque vous modifiez la variable `today_date`, le texte qui s'affiche dans `my_txt` est mis à jour.

```
this.(reateTextField("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 clips qui ne sont pas visibles (dont la propriété `_visible` est définie sur `false`) sont désactivés.

Disponibilité
Flash Lite 2.0

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 aussi
`_visible (propriété Button._visible)`, `_visible (propriété MovieClip._visible)`

_width (propriété TextField._width)

```
public _width : Number
```

Largeur du champ texte, en pixels.

Disponibilité
Flash Lite 2.0

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.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.type = "input";
height_txt.text = my_txt._height;
height_txt.onChanged = function() {
    my_txt._height = this.text;
}
```

Pour tester l'exemple, essayez de saisir de nouvelles valeurs dans `width_txt` et `height_txt` pour modifier les dimensions de `my_txt`.

Voir aussi

`_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 n'en comporte pas.

Disponibilité

Flash Lite 2.0

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 Lite 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 (TextField._x, propriété)

```
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, il se trouve alors dans le système de coordonnées locales du clip qui l'encadre. Ainsi, dans le cas d'un clip qui a subi 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 font référence à la position du point d'alignement.

Disponibilité

Flash Lite 2.0

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(mouseListe(er);
```

Voir aussi

_xscale (propriété TextField._xscale), _y (propriété TextField._y), _yscale (propriété TextField._yscale)

_xmouse (propriété TextField._xmouse)

```
public _xmouse : Number [read-only]
```

Renvoie la coordonnée *x* de la position de la souris par rapport au champ texte.

Remarque : Cette propriété n'est prise en charge par Flash Lite que si `System.capabilities.hasMouse` est définie sur `true` ou si `System.capabilities.hasStylus` est définie sur `true`.

Disponibilité

Flash Lite 2.0

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);
```

Voir aussi

_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

Flash Lite 2.0

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 aussi

_x (TextField._x, propriété), _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, il se trouve dans le système de coordonnées locales du clip qui l'encadre. Ainsi, dans le cas d'un clip qui a subi 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 font référence à la position du point d'alignement.

Disponibilité

Flash Lite 2.0

Exemple

Consultez l'exemple de `TextField._x`.

Voir aussi

_x (TextField._x, propriété), _xscale (propriété TextField._xscale), _yscale (propriété TextField._yscale)

_ymouse (propriété TextField._ymouse)

```
public _ymouse : Number [read-only]
```

Indique la coordonnée *y* de la position de la souris par rapport au champ texte.

Remarque : Cette propriété n'est prise en charge par Flash Lite que si `System.capabilities.hasMouse` est définie sur `true` ou si `System.capabilities.hasStylus` est définie sur `true`.

Disponibilité

Flash Lite 2.0

Exemple

Consultez l'exemple de `TextField._xmouse`.

Voir aussi

_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

Flash Lite 2.0

Exemple

Consultez l'exemple de `TextField._xscale`.

Voir aussi

`_x (TextField._x, propriété)`, `_xscale (propriété TextField._xscale)`, `_y (propriété TextField._y)`

TextFormat

```
Object
    |
    +-TextFormat
```

```
public class TextFormat
extends Object
```

La classe TextFormat regroupe les informations de mise en forme de caractères. La classe TextFormat permet de 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 pour les polices incorporées ni pour les polices de périphérique.

Disponibilité

Flash Lite 2.0

Voir aussi

`setTextFormat (méthode TextField.setTextFormat)`, `getTextFormat (méthode TextField.getTextFormat)`

Résumé des propriétés

Modificateurs	Propriété	Description
	align : String	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	Nombre indiquant la couleur du texte.
	font : String	Chaîne qui spécifie le nom de la police du texte.
	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 est en italique.
	leading : Number	Entier représentant l'espace vertical en pixels laissé entre les lignes (appelé *interlignage*).
	leftMargin : Number	Marge gauche du paragraphe, en points.
	rightMargin : Number	Marge droite du paragraphe, en points.
	size : Number	Taille en points du texte dans ce format de texte.

Modificateurs	Propriété	Description
	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 indiquant si le texte qui utilise ce format texte est souligné (`true`) ou non (`false`).
	url : String	Indique l'URL correspondant à l'hyperlien du texte de ce format de texte.

Propriétés héritées de la classe Object

`constructor (propriété Object.constructor), __proto__ (Object.__proto__, propriété), prototype (propriété Object.prototype), __resolve (Object.__resolve, propriété)`

Récapitulatif 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	Renvoie les informations de mesure de 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
```

Chaîne indiquant l'alignement du paragraphe. Vous pouvez appliquer cette propriété tant aux champs texte sta tique qu'aux champs texte dynamique. Le tableau suivant 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.

La valeur par défaut est `null`, ce qui indique que la propriété n'est pas définie.

Disponibilité
Flash Lite 2.0

Exemple
L'exemple suivant crée un champ texte avec une bordure et utilise `TextFormat.align` pour centrer le texte.

```
var my_fmt:TextFormat = new TextFormat();
my_fmt.align = "center";

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 text field object text";
my_txt.setTextFormat(my_fmt);
```

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
Flash Lite 2.0

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 text 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. Lorsque sa valeur est `true`, le texte est en gras.

Remarque : Pour l'arabe, l'hébreu et le thaïlandais, cette propriété applique le formatage uniquement au niveau des paragraphes.

Disponibilité
Flash Lite 2.0

Exemple
L'exemple suivant crée un champ texte qui comprend 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 text 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`.

Remarque :pour Flash Lite, cette propriété ne s'applique qu'aux polices intégrées. Cette propriété n'est pas prise en charge pour l'arabe, l'hébreu et le thaïlandais.

Disponibilité
Flash Lite 2.0

Exemple
L'exemple suivant crée un champ texte lors de l'exécution et y place une chaîne avec un saut de ligne. La classe TextFormat permet de formater les caractères en ajoutant des puces à chaque ligne 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
```

Nombre indiquant la couleur du texte. Ce nombre contient trois composants RVB 8 bits ; par exemple, 0xFF0000 correspond au rouge et 0x00FF00 au vert.

Remarque : Pour l'arabe, l'hébreu et le thaïlandais, cette propriété applique le formatage uniquement au niveau des paragraphes.

Disponibilité
Flash Lite 2.0

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 text field object text";
my_txt.setTextFormat(my_fmt);
```

font (propriété TextFormat.font)

```
public font : String
```

Chaîne qui spécifie le nom de la police du texte. La valeur par défaut est `null`, ce qui indique que la propriété n'est pas définie.

Remarque :pour Flash Lite, cette propriété ne s'applique qu'aux polices intégrées. Cette propriété n'est pas prise en charge pour l'arabe, l'hébreu et le thaïlandais.

Disponibilité
Flash Lite 2.0

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 text field object text";
mytext.setTextFormat(myformat);
```

getTextExtent (méthode TextFormat.getTextExtent)

```
public getTextExtent(text:String, [width:Number]) : Object
```

Renvoie les informations de mesure de 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.

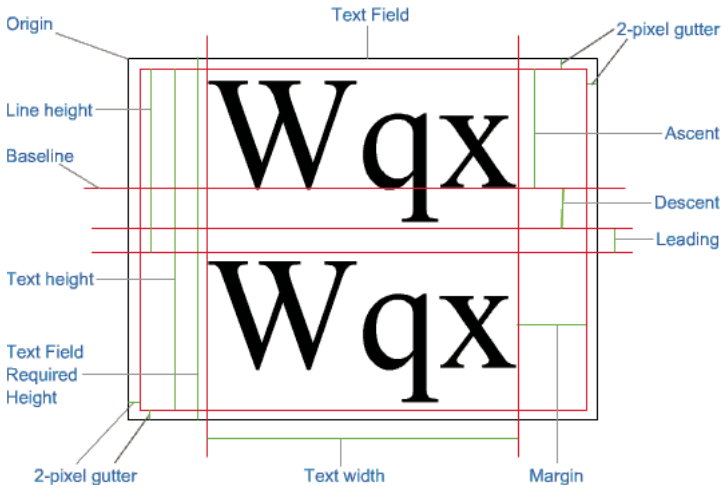
Si un paramètre `width` est spécifié, un retour à la ligne automatique est appliqué au texte spécifié. Ceci vous permet de déterminer la hauteur à laquelle un champ affiche l'ensemble du texte spécifié.

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 de la première ligne de texte est positionnée au début du champ 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 par un objet de champ texte pour afficher l'ensemble de la chaîne de texte. Les champs texte ont une marge de reliure de 2 pixels autour d'eux, si bien que la valeur de `textFieldHeight` est égale à celle de `height` + 4 ; de même, la valeur de `textFieldWidth` est toujours égale à celle de `width` + 4.

Si vous créez un champ texte basé sur les dimensions du 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 texte, y compris le nom de police, la taille de police et l'interligne. La valeur par défaut pour l'interligne est 2.

Disponibilité

Flash Lite 2.0

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 doté des 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 forme 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 forme 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 = "Adobe 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. La valeur par défaut est `null`, ce qui indique que la propriété n'est pas définie.

Disponibilité
Flash Lite 2.0

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 text field object text";
mytext.setTextFormat(myformat);
```

Voir aussi
blockIndent (propriété TextFormat.blockIndent)

italic (propriété TextFormat.italic)

```
public italic : Boolean
```

Valeur booléenne indiquant si le texte est en italique. La valeur par défaut est `null`, ce qui indique que la propriété n'est pas définie.

Remarque : Pour l'arabe, l'hébreu et le thaïlandais, cette propriété applique le formatage uniquement au niveau des paragraphes.

Disponibilité
Flash Lite 2.0

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);
```

leading (propriété TextFormat.leading)

```
public leading : Number
```

Entier représentant l'espace vertical en pixels laissé entre les lignes (appelé *interlignage*). La valeur par défaut est `null`, ce qui indique que la propriété n'est pas définie.

Disponibilité
Flash Lite 2.0

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
Flash Lite 2.0

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 text field object text";
mytext.setTextFormat(myformat);
```

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
Flash Lite 2.0

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 text 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.

Remarque : Pour l'arabe, l'hébreu et le thaïlandais, cette propriété applique le formatage uniquement au niveau des paragraphes.

Disponibilité
Flash Lite 2.0

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 indiqué 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).

Remarque :pour Flash Lite, cette propriété ne s'applique qu'aux polices intégrées. Cette propriété n'est pas prise en charge pour l'arabe, l'hébreu et le thaïlandais.

Disponibilité
Flash Lite 2.0

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 active dans la fenêtre actuelle, `_blank` spécifie une nouvelle fenêtre, `_parent` spécifie le parent de l'image active 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

Flash Lite 2.0

Exemple

L'exemple suivant crée un champ de texte incluant un hyperlien pointant vers le site Web d'Adobe. L'exemple utilise `TextFormat.target` pour afficher le site Web d'Adobe dans une nouvelle fenêtre de navigateur.

```
var myformat:TextFormat = new TextFormat();
myformat.url = "http://www.adobe.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 Adobe.com";
mytext.setTextFormat(myformat);
```

Voir aussi

url (propriété TextFormat.url)

constructeur TextFormat

```
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 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
Flash Lite 2.0

Paramètres
font : String [facultatif] - Nom d'une police pour le texte sous forme de chaîne.

size : Number [facultatif] - Entier indiquant la taille en points.

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 dans ce format de texte. Si `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. `"left"` - Le paragraphe est aligné à gauche. `"center"` - Le paragraphe est centré. `"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.

Disponibilité
Flash Lite 2.0

Exemple
L'exemple suivant crée un objet TextFormat, puis formate le champ texte `stats_txt` et crée un nouveau champ dans lequel 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 consulter un autre exemple, reportez-vous au fichier animations.fla du dossier d'exemples ActionScript disponible à l'adresse Page d'exemples Adobe Flash. Téléchargez le fichier .zip, puis décompressez-le pour afficher le dossier correspondant à votre version d'ActionScript et consulter l'exemple.

underline (propriété TextFormat.underline)

```
public underline : Boolean
```

Valeur booléenne indiquant 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 jambage. La valeur par défaut est `null`, ce qui indique que la propriété n'est pas définie.

Remarque : Pour l'arabe, l'hébreu et le thaïlandais, cette propriété applique le formatage uniquement au niveau des paragraphes.

Disponibilité

Flash Lite 2.0

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 correspondant à l'hyperlien du texte de ce format de 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

Flash Lite 2.0

Exemple

Cet exemple crée un champ de texte constituant un hyperlien qui pointe vers le site Web d'Adobe.

```
var myformat:TextFormat = new TextFormat();
myformat.url = "http://www.adobe.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 Adobe.com";
mytext.setTextFormat(myformat);
```

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

Flash Lite 3.1

Voir aussi

`transform (propriété MovieClip.transform)`, `ColorTransform (flash.geom.ColorTransform)`, `Matrix (flash.geom.Matrix)`

Résumé des propriétés

Modificateurs	Propriété	Description
	colorTransform : ColorTransform	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.

constructor (propriété Object.constructor), __proto__ (Object.__proto__, propriété), « prototype (propriété Object.prototype) » à la page 509, __resolve (Object.__resolve, propriété)

Récapitulatif des constructeurs

Signature	Description
Transform (mc : MovieClip)	Crée un nouvel objet Transform attaché à l'objet MovieClip donné.

Résumé de la méthode

« addProperty (méthode Object.addProperty) » à la page 504, « hasOwnProperty (méthode Object.hasOwnProperty) » à la page 507, « isPropertyEnumerable (méthode Object.isPropertyEnumerable) » à la page 507, « isPrototypeOf (méthode Object.isPrototypeOf) » à la page 508, « registerClass (méthode Object.registerClass) » à la page 510, « toString (méthode Object.toString) » à la page 514« unwatch (méthode Object.unwatch) » à la page 515, « valueOf (méthode Object.valueOf) » à la page 516, « watch (méthode Object.watch) » à la page 517

colorTransform (propriété Transform.colorTransform)

```
public colorTransform : ColorTransform
```

Objet ColorTransform contenant des valeurs qui règlent de façon universelle les couleurs dans le clip.

Disponibilité

Flash Lite 3.1

Exemple

L'exemple suivant applique l'objet ColorTransform `blueColorTransform` à l'objet Transform `trans`. Cet objet ColorTransform convertit la couleur de l'objet 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 aussi

« ColorTransform (flash.geom.ColorTransform) » à la page 268

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

Flash Lite 3.1

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 aussi

« ColorTransform (flash.geom.ColorTransform) » à la page 268

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

Flash Lite 3.1

Exemple

L'exemple suivant applique deux objets Transform, à la fois à un objet MovieClip parent et à un objet MovieClip enfant. Un objet Matrix `scaleMatrix` est ensuite appliqué à l'objet Transform `parentTrans`, qui met 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

Flash Lite 3.1

Exemple

L'exemple suivant applique l'objet Matrix `scaleMatrix` à l'objet Transform `trans`. Cet objet Matrix met à l'échelle l'objet 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 aussi

« Matrix (flash.geom.Matrix) » à la page 370

pixelBounds (propriété Transform.pixelBounds)

```
public pixelBounds : Rectangle
```

Objet rectangle qui définit le cadre de délimitation de l'objet MovieClip sur la Scène.

Disponibilité

Flash Lite 3.1

Exemple

L'exemple suivant crée un objet Transform `trans` et suit sa propriété `pixelBounds`. 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

Flash Lite 3.1

Paramètres

mc: MovieClip - Objet MovieClip auquel le nouvel objet Transform s'applique.

Exemple

L'exemple suivant crée l'objet 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 permet d'afficher le contenu vidéo qui est incorporé dans votre fichier SWF, stocké localement sur le périphérique hôte, ou téléchargé en flux continu à partir d'un autre emplacement.

Remarque : Le lecteur de Flash Lite 2.0 traite les vidéos de façon différente que Flash Player 7. Voici les principales différences :

- Flash Player 7 rend directement les données vidéo (incorporées ou diffusées en flux continu). Le lecteur de Flash Lite 2.0 ne rend pas les données vidéo ; il les transmet au périphérique mobile. Le lecteur de Flash Lite 3.0 prend en charge le rendu Flash Video (FLV) directement par Flash Lite.
- Flash Player 7 prend en charge de nombreux formats vidéo en supplément du format FLV. Flash Lite 2.0 prend en charge la lecture des vidéos dans les cas suivants : vidéo incorporée dans un fichier SWF ; vidéo résidant dans un fichier distinct sur le périphérique hôte et données vidéo qui sont diffusées en flux continu sur le réseau (en temps réel). Le lecteur de Flash Lite 2.0 ne prend en charge que les formats vidéo pris en charge par le périphérique portable, tandis que le lecteur pour FIash Lite 3.0 prend en charge le rendu FLV de manière native.
- Flash Player 7 permet de regrouper les données dans un fichier SWF ou de les diffuser en flux continu à l'aide de l'objet Video et en affectant soit un objet NetStream, soit un objet Camera en tant que source des informations vidéo. Cependant, le lecteur de Flash Lite 2.0 ne prend pas en charge les objets `NetStream` et Camera. Par contre, Flash Lite 2.0 utilise un nouveau symbole de bibliothèque appelé `Video` pour incorporer les données vidéo source et diffuser la vidéo en flux continu pour les périphériques mobiles. Dans la mesure où Flash Lite 2.0 ne prend pas en charge l'objet NetStream, utilisez les méthodes et les propriétés de la classe Video pour contrôler la lecture de la vidéo. Le lecteur de Flash Lite 3.0 prend en charge les objets `NetStream` et NetConnection et vous utilisez les méthodes et les propriétés de ces classes pour contrôler la lecture FLV. Flash Lite 3.0 prend également en charge une nouvelle propriété dans la classe Video, `attachVideo`, qui indique un flux vidéo à afficher dans l'objet Video sur la scène. Flash Lite 3.0 ne prend pas en charge l'objet Camera.

En raison des limitations des périphériques mobiles (processeurs plus lents, restrictions de mémoire et formats de codage propriétaires), Flash Lite 2.0 ne permet pas de rendre les informations vidéo directement. Les formats de fichier pris en charge pour la vidéo dépendent du fabricant de périphérique mobile. Pour plus d'informations sur les formats vidéo pris en charge, consultez la documentation des plates-formes matérielles sur lesquelles vous prévoyez de déployer votre application. A l'opposé, Flash Lite 3.0 peut rendre les vidéos Flash directement.

Flash Lite 2.0 ne prend pas en charge les fonctionnalités Flash Player 7 suivantes :

- Diffusion en flux continu des données vidéo à partir de Flash Media Server ;
- Enregistrement vidéo

Flash Lite 3.0 ajoute la prise en charge des fonctionnalités Flash Player 7 suivantes :

- Rendu de vidéos Flash directement par le lecteur à l'aide des versions des codecs On2 et Sorenson optimisés pour les périphériques portables
- Transmission des données vidéo sur une connexion RTMP (Real Time Messaging Protocol) vers Flash Media Server. (Les connexions RTMPT (Real Time Messaging Protocol Tunnel) et RTMPS (Real Time Messaging Protocol Secure) ne sont pas prises en charge, ni les connexions multiples.)

Flash Lite 3.0 ne prend pas en charge les fonctionnalités Flash Player 7 suivantes :

- Enregistrement vidéo
- Objet Camera

Disponibilité
Flash Lite 2.0

Résumé des propriétés
Propriétés héritées de la classe Object

constructor (propriété Object.constructor), __proto__ (Object.__proto__, propriété), prototype (propriété Object.prototype), __resolve (Object.__resolve, propriété)

Résumé des événements

Evénement	Description
onStatus = function(infoObject :Object) {}	Gestionnaire de rappel qui peut être appelé par le périphérique pour indiquer des conditions d'état ou d'erreur.

Résumé de la méthode

Modificateurs	Signature	Description
	close() : Void	Arrête la lecture de la vidéo, libère la mémoire associée à cet objet Video et efface la zone réservée à la diffusion de l'écran.
	pause() : Void	Arrête la lecture de la vidéo et poursuit le rendu de l'image affichée.
	play() : Boolean	L'appel de cette méthode ouvre une source vidéo et commence la lecture de cette dernière.
	resume() : Void	L'appel de cette méthode reprend la lecture de la vidéo.
	stop() : Void	Arrête la lecture de la vidéo et poursuit le rendu de l'image affichée.

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)

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()` 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 émise.

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 la section `MovieClip.attachAudio()`.

Disponibilité

Flash Lite 3.0

Exemple
L'exemple suivant lit un fichier enregistré précédemment et appelé video1.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 aussi
play (méthode Video.play), stop (méthode Video.stop), resume (méthode Video.resume)

close (méthode Video.close)

```
public close() : Void
```

Arrête la lecture de la vidéo, libère la mémoire associée à cet objet Video et efface la zone réservée à la diffusion de l'écran.

Disponibilité
Flash Lite 2.0

Exemple
L'exemple suivant ferme la vidéo en cours de lecture par un objet Video appelé `video1`.

```
video1.close()
```

Voir aussi
play (méthode Video.play), pause (méthode Video.pause), resume (méthode Video.resume)

onStatus (gestionnaire Video.onStatus)

```
onStatus = function(infoObject:Object) {}
```

Gestionnaire de rappel qui peut être appelé par le périphérique pour indiquer des conditions d'état ou d'erreur.

Disponibilité
Flash Lite 3.0

Paramètres
infoObject : Object - Le paramètre `infoObject` a deux propriétés :

- `code:String` - Description de l'erreur ou de la condition d'état (spécifique au périphérique).
- `level:Number` - Zéro pour une erreur et différent de zéro pour une réussite (spécifique au périphérique).

Exemple
L'exemple suivant indique comment créer une fonction `Video.onStatus()` qui affiche une condition d'état ou d'erreur.

```
var v:Video; // v is a Video object on the stage.
v.onStatus = function(o:Object)
{
    if ( o.level )
    {
        trace( "Video Status Msg (" + o.level + "): " + o.code );
    }
    else
    {
        trace( "Video Status Error: " + o.code );
    }
}
v.play("a.vid");
```

pause (méthode Video.pause)

```
public pause() : Void
```

Arrête la lecture de la vidéo et poursuit le rendu de l'image affichée. L'appel suivant à `Video.resume()` reprend la lecture à partir de la position actuelle.

Disponibilité

Flash Lite 2.0

Exemple

L'exemple suivant arrête la vidéo en cours de lecture par un objet Video (appelé `my_video`) lorsque l'utilisateur clique sur l'occurrence `close_btn`.

```
// video1 is the name of a Video object on Stage
video1.pause()
```

Voir aussi

play (méthode Video.play), stop (méthode Video.stop), resume (méthode Video.resume)

play (méthode Video.play)

```
public play() : Boolean
```

L'appel de cette méthode ouvre une source vidéo et commence la lecture de cette dernière.

Disponibilité

Flash Lite 2.0

Valeur renvoyée

Boolean - Valeur `true` si le périphérique portable peut rendre la vidéo, `false` sinon.

Exemple

L'exemple suivant interrompt et efface le fichier video1.flv, qui est lu par un objet Video (appelé `video1`).

```
video1.play( "http://www.macromedia.com/samples/videos/clock.3gp" );
```

Vous pouvez également utiliser un objet Video sur la scène pour lire des vidéos de périphérique regroupées, directement à partir de la bibliothèque. Pour ce faire, regroupez la vidéo de périphérique dans votre bibliothèque d'application. Vous pouvez également affecter un identifiant au symbole vidéo qui permet de référencer le symbole vidéo à l'aide du code ActionScript. Vous pouvez également lire des vidéos directement à partir de la bibliothèque en transmettant l'identifiant ActionScript du symbole à la méthode `Video.play()`, comme indiqué dans l'exemple suivant :

```
placeHolderVideo.play("symbol://ocean_video");
```

Pour plus d'informations sur la lecture de vidéos à partir de la bibliothèque, consultez la section « Lecture d'une vidéo regroupée directement à partir de la bibliothèque » dans le guide *Développement d'applications Flash Lite 2.x*.

Voir aussi

`stop (méthode Video.stop)`, `pause (méthode Video.pause)`, `resume (méthode Video.resume)`

resume (méthode Video.resume)

```
public resume() : Void
```

L'appel de cette méthode reprend la lecture de la vidéo.

Si `Video.pause()` a été appelée auparavant, la lecture reprend à la position actuelle. Si `Video.stop()` a été appelée auparavant, la lecture reprend à la première image.

Disponibilité

Flash Lite 2.0

Exemple

L'exemple suivant reprend la vidéo en cours de lecture par un objet Video appelé `video1`.

```
video1.resume()
```

Voir aussi

`pause (méthode Video.pause)`, `stop (méthode Video.stop)`

stop (méthode Video.stop)

```
public stop() : Void
```

Arrête la lecture de la vidéo et poursuit le rendu de l'image affichée. Tout autre appel à `Video.resume()` reprend la lecture à partir de la première image de la vidéo.

Disponibilité

Flash Lite 2.0

Exemple

L'exemple suivant arrête la vidéo en cours de lecture par un objet Video (appelé `my_video`) lorsque l'utilisateur clique sur l'occurrence `close_btn`.

```
// video1 is the name of a Video object on Stage
video1.stop();
```

Voir aussi

play (méthode Video.play), pause (méthode Video.pause), resume (méthode Video.resume)

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 les propriétés suivantes, consultez la classe XMLNode : `appendChild()`, `attributes`, `childNodes`, `cloneNode()`, `firstChild`, `hasChildNodes()`, `insertBefore()`, `lastChild`, `nextSibling`, `nodeName`, `nodeType`, `nodeValue`, `parentNode`, `previousSibling`, `removeNode()`, `toString()`

Dans les versions précédentes du Guide de référence du langage ActionScript, les méthodes et les propriétés ci-dessus 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 conformément à la recommandation DOM Level 1 de W3C: 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

Flash Lite 2.0

Voir aussi

appendChild (méthode XMLNode.appendChild), attributes (propriété XMLNode.attributes), childNodes (propriété XMLNode.childNodes), cloneNode (méthode XMLNode.cloneNode), firstChild (propriété XMLNode.firstChild) hasChildNodes (méthode XMLNode.hasChildNodes), insertBefore (méthode XMLNode.insertBefore) lastChild (propriété XMLNode.lastChild), nextSibling (propriété XMLNode.nextSibling) nodeName (propriété XMLNode.nodeName), nodeType (propriété XMLNode.nodeType) nodeValue (propriété XMLNode.nodeValue), parentNode (propriété XMLNode.parentNode) previousSibling (propriété XMLNode.previousSibling), removeNode (méthode XMLNode.removeNode) toString (méthode XMLNode.toString)

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.
	ignoreWhite : Boolean	La valeur par défaut est `false`.
	loaded : Boolean	Indique si le document XML a été chargé avec succès.
	état : 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 (propriété XMLNode.attributes),childNodes (propriété
XMLNode.childNodes)firstChild (propriété XMLNode.firstChild),lastChild (propriété
XMLNode.lastChild)nextSibling (propriété XMLNode.nextSibling),nodeName (propriété
XMLNode.nodeName),nodeType (propriété XMLNode.nodeType),nodeValue (propriété
XMLNode.nodeValue),parentNode (propriété XMLNode.parentNode)previousSibling
(propriété XMLNode.previousSibling)
```

Propriétés héritées de la classe Object

```
constructor (propriété Object.constructor),__proto__ (Object.__proto__,
propriété)prototype (propriété Object.prototype),__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.
onLoad = function(success:Boolean) {}	Appelé par Flash Lite 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 des en-têtes de requête HTTP (tels que Content-Type ou SOAPAction) envoyés avec des actions POST.
	createElement(name:String) : XMLNode	Crée un nouvel élément XML avec le nom spécifié dans le paramètre.
	createTextNode(valeur: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(valeur:String) : Void	Analyse le texte XML spécifié dans le paramètre value et renseigne l'objet XML spécifié avec l'arborescence XML obtenue.
	send(url:String, [target:String], method:String) : Boolean	Code l'objet XML spécifié dans un document XML et l'envoie à l'URL spécifiée à l'aide de la méthode POST dans un navigateur.
	sendAndLoad(url:String, resultXML:XML) : Void	Code l'objet XML spécifié dans 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 resultXMLobject spécifié dans les paramètres.

Méthodes héritées de la classe XMLNode

appendChild (méthode XMLNode.appendChild), cloneNode (méthode XMLNode.cloneNode) hasChildNodes (méthode XMLNode.hasChildNodes), insertBefore (méthode XMLNode.insertBefore) removeNode (méthode XMLNode.removeNode), toString (méthode XMLNode.toString)

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)

```
public addRequestHeader(header:Object, headerValue:String) : Void
```

Ajoute ou modifie des en-têtes de requête HTTP (tels que Content-Type ou SOAPAction) envoyés avec des actions POST. Dans la première utilisation, vous transmettez deux chaînes (header et headerValue) à la méthode. 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`, `Proxy-Authorization`, `Public`, `Range`, `Retry-After`, `Server`, `TE`, `Trailer`, `Transfer-Encoding`, `Upgrade`, `URI`, `Vary`, `Via`, `Warning` et `WWW-Authenticate`.

Disponibilité
Flash Lite 2.0

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 de requête HTTP appelé `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 aussi
addRequestHeader (méthode LoadVars.addRequestHeader)

contentType (propriété XML.contentType)

```
public contentType : 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
Flash Lite 2.0

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 aussi

send (méthode XML.send), sendAndLoad (méthode XML.sendAndLoad)

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

Flash Lite 2.0

Paramètres

name : String - Nom de balise de l'élément XML en cours de création.

Valeur renvoyée

XMLNode - Objet XMLNode ; 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 aussi

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

Flash Lite 2.0

Paramètres

valeur : String - Chaîne ; 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 aussi

createElement (méthode nodeML.createElement)

docTypeDecl (propriété XML.docTypeDecl)

```
public docTypeDecl : String
```

Spécifie des informations à propos de la déclaration `DOCTYPE` du document XML. Après l'analyse du texte XML dans un objet XML, la propriété `XML.docTypeDecl` de l'objet XML est définie sur le texte de la déclaration `DOCTYPE` du document XML (par exemple, `<!DOCTYPEgreeting 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 l'analyseur et enregistrée dans la propriété `XML.docTypeDecl`, mais la DTD n'est pas validée.

Si aucune déclaration `DOCTYPE` n'est détectée pendant 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 défini, aucune déclaration `DOCTYPE` n'est produite.

Disponibilité
Flash Lite 2.0

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 aussi
xmlDecl (propriété XML.xmlDecl)

getBytesLoaded (méthode XML.getBytesLoaded)

```
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 chargé d'un document XML.

Disponibilité
Flash Lite 2.0

Valeur renvoyée
Number - Entier indiquant le nombre d'octets chargés.

Exemple
L'exemple suivant montre comment utiliser la méthode `XML.getBytesLoaded()` avec la méthode `XML.getBytesTotal()` pour suivre l'état d'avancement 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 Lite 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 aussi

getBytesTotal (méthode XML.getBytesTotal)

getBytesTotal (méthode XML.getBytesTotal)

```
public getBytesTotal() : Number
```

Renvoie la taille, en octets, du document XML.

Disponibilité

Flash Lite 2.0

Valeur renvoyée

Number - Entier.

Exemple

Consultez l'exemple de `XML.getBytesLoaded()`.

Voir aussi

getBytesLoaded (méthode XML.getBytesLoaded)

ignoreWhite (propriété XML.ignoreWhite)

```
public ignoreWhite : Boolean
```

La valeur par défaut est `false`. Lorsque la valeur 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.

Utilisation 1 : Vous pouvez définir la propriété `ignoreWhite` pour les objets XML individuels, comme indiqué par le code suivant :

```
my_xml.ignoreWhite = true;
```

Utilisation 2 : Vous pouvez définir la propriété `ignoreWhite` par défaut pour les objets XML, comme indiqué par le code suivant :

```
XML.prototype.ignoreWhite = true;
```

Disponibilité
Flash Lite 2.0

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 `flooring.ignoreWhite` sur `false` ou si vous 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>
```

Pour un exemple, consultez les fichiers XML_blogTracker.fla et XML_languagePicker.fla dans le dossier d'exemples ActionScript à l'adresse www.adobe.com/go/learn_fl_samples. Téléchargez le fichier .zip, puis décompressez-le pour afficher le dossier correspondant à votre version d'ActionScript et consulter l'exemple.

load (méthode XML.load)

```
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()`.

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.

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 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 interdomaine* sur le serveur hébergeant le fichier SWF.

Lorsque la méthode `load()` est exécutée, la propriété `loaded` de l'objet XML est définie sur `false`. Lorsque le téléchargement des données XML est terminé, 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 d'événements `onLoad` de l'objet XML est appelé.

Disponibilité

Flash Lite 2.0

Paramètres

url : String - Chaîne qui représente l'emplacement de 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. Pour plus de détails, consultez la section Description.

Valeur renvoyée

Boolean - `false` si aucun paramètre (`null`) n'est transmis ; sinon, `true`. Utilisez le gestionnaire d'événements `onLoad()` pour vérifier le succès d'un document XML chargé.

Exemple

L'exemple simple 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 relatif à `XML.ignoreWhite`.

Voir aussi

`ignoreWhite (propriété XML.ignoreWhite)`, `loaded (propriété XML.loaded)`, `onLoad (gestionnaire XML.onLoad)`

loaded (propriété XML.loaded)

`public loaded : Boolean`

Indique si le document XML 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` 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

Flash Lite 2.0

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 `onLoad` est appelé. Si cet appel aboutit, `true` s'affiche pour le statut `loaded` dans le panneau Sortie.

```
success: true
loaded: true
status: 0
```

Voir aussi

load (méthode XML.load), onLoad (gestionnaire XML.onLoad)

onData (gestionnaire XML.onData)

```
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

Flash Lite 2.0

Paramètres

src : String - Chaîne ou `undefined` ; données brutes, généralement au format XML, envoyées 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 aussi

onLoad (gestionnaire XML.onLoad)

onLoad (gestionnaire XML.onLoad)

```
onLoad = function(success:Boolean) {}
```

Appelé par Flash Lite 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
Flash Lite 2.0

Paramètres
success : Boolean - Valeur booléenne renvoyant `true` si l'objet XML a bien été chargé à l'aide d'une opération `XML.load()` ou `XML.sendAndLoad()` ; sinon, 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 aussi
load (méthode XML.load), sendAndLoad (méthode XML.sendAndLoad)

parseXML (méthode XML.parseXML)

```
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
Flash Lite 2.0

Paramètres
valeur : String - Chaîne qui représente le texte XML à analyser et à 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 (méthode XML.send)

```
public send(url:String, [target:String], method:String) : Boolean
```

Code l'objet XML spécifié dans un document XML et l'envoie à l'URL spécifiée à l'aide de la méthode `POST` dans un navigateur. L'environnement de test Flash utilise toujours la méthode `GET`.

Disponibilité
Flash Lite 2.0

Paramètres
url : String - Chaîne, URL de destination de l'objet XML spécifié.

target : String [facultatif] - Chaîne, fenêtre de navigateur affichant 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 `window`, le système applique `_self`.

method : String [facultatif] - Méthode du protocole HTTP utilisée : `GET` ou `POST`. Dans un navigateur, la valeur par défaut est `POST`. Dans un environnement de test Flash, la valeur par défaut est `GET`.

Valeur renvoyée
Boolean - `false` si aucun paramètre n'est spécifié ; 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 aussi

sendAndLoad (méthode XML.sendAndLoad)

sendAndLoad (méthode XML.sendAndLoad)

```
public sendAndLoad(url:String, resultXML:XML) : Void
```

Code l'objet XML spécifié dans 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()`.

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.

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 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 inter-domaines* sur le serveur hébergeant le fichier SWF.

Lorsque `sendAndLoad()` est exécutée, la propriété `loaded` de l'objet XML est définie sur `false`. Lorsque le téléchargement des données XML est terminé, 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.

Disponibilité

Flash Lite 2.0

Paramètres

url : String - Chaîne, URL de destination de l'objet XML spécifié. Si le fichier SWF qui émet cet appel s'exécute sur un navigateur Web, l'`url` doit appartenir au même domaine que le fichier SWF. Pour plus de détails, consultez 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 aussi

send (méthode XML.send), load (méthode XML.load), loaded (propriété XML.loaded), onLoad (gestionnaire XML.onLoad)

status (propriété XML.status)

```
public status : Number
```

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 (données de caractères) ne s'est pas terminée correctement.
- -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

Flash Lite 2.0

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");
```

Constructeur XML

```
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 documents XML.

Disponibilité
Flash Lite 2.0

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 aussi
createElement (méthode nodeML.createElement), createTextNode (méthode XML.createTextNode)

xmlDecl (propriété XML.xmlDecl)

```
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.XML`. La méthode `XML.toString()` fournit le contenu 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
Flash Lite 2.0

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 du type de document 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();
```

Voir aussi

docTypeDecl (propriété XML.docTypeDecl)

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

Flash Lite 2.0

Voir aussi

hasXMLSocket (propriété capabilities.hasXMLSocket)

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.
	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.
	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.
	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

```
constructor (propriété Object.constructor), __proto__ (Object.__proto__,
propriété) prototype (propriété Object.prototype), __resolve (Object.__resolve,
propriété)
```

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.
	hasChildNodes() : Boolean	Indique si l'objet XML comporte ou non des nœuds enfants.
	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 (méthode XMLNode.appendChild)

```
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
Flash Lite 2.0

Paramètres
newChild : XMLNode - Nœud XMLNode qui représente le nœud à déplacer de son emplacement actuel vers la liste enfant de l'objet 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`.
- Montre comment déplacer un nœud à l'aide de 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 (propriété XMLNode.attributes)

```
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 un attribut est appelé `color`, vous récupérerez la valeur de l'attribut en spécifiant color comme nom de la propriété, tel qu'indiqué par le code suivant :

```
var myColor:String = doc.firstChild.attributes.color
```

Disponibilité

Flash Lite 2.0

Exemple

L'exemple suivant affiche les noms d'attributs XML :

```
// create a tag called 'mytag' with
// an attribute called 'name' with value 'Val'
var doc:XML = new XML("<mytag name=\"Val\"> item </mytag>");

// assign the value of the 'name' attribute to variable y
var y:String = doc.firstChild.attributes.name;
trace (y); // output: Val

// create a new attribute named 'order' with value 'first'
doc.firstChild.attributes.order = "first";

// assign the value of the 'order' attribute to variable z
var z:String = doc.firstChild.attributes.order
trace(z); // output: first
```

Les informations suivantes apparaissent dans le panneau Sortie :

```
Val
first
```

childNodes (propriété XMLNode.childNodes)

```
public childNodes : Array [read-only]
```

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

Flash Lite 2.0

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 aussi

nodeType (propriété XMLNode.nodeType), appendChild (méthode XMLNode.appendChild), insertBefore (méthode XMLNode.insertBefore) removeNode (méthode XMLNode.removeNode)

cloneNode (méthode XMLNode.cloneNode)

```
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` prend la valeur `false`, ou si le nœud `my_xml` n'a pas de nœuds enfants, `firstChild` et `lastChild` sont également null.

Disponibilité

Flash Lite 2.0

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 (propriété XMLNode.firstChild)

```
public firstChild : XMLNode [read-only]
```

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

Flash Lite 2.0

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 le trouverez 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'intégralité du script, consultez le fichier XML_languagePicker.fla dans le dossier d'exemples ActionScript à l'adresse www.adobe.com/go/learn_fl_samples. Téléchargez le fichier .zip, puis décompressez-le pour afficher le dossier correspondant à votre version d'ActionScript et consulter l'exemple.

Voir aussi

appendChild (méthode XMLNode.appendChild), insertBefore (méthode XMLNode.insertBefore), removeNode (méthode XMLNode.removeNode)

hasChildNodes (méthode XMLNode.hasChildNodes)

```
public hasChildNodes() : Boolean
```

Indique si l'objet XML comporte ou non des nœuds enfants.

Disponibilité

Flash Lite 2.0

Valeur renvoyée

Boolean - `true` si le nœud XMLNode spécifié a un ou plusieurs nœuds enfants ; sinon, `false`.

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 (méthode XMLNode.insertBefore)

```
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

Flash Lite 2.0

Paramètres

newChild : XMLNode - Objet XMLNode à insérer.

insertPoint : XMLNode - Objet XMLNode qui va suivre 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 aussi

hasXMLSocket (propriété capabilities.hasXMLSocket), cloneNode (méthode XMLNode.cloneNode)

lastChild (propriété XMLNode.lastChild)

```
public lastChild : XMLNode [read-only]
```

Valeur XMLNode qui fait référence au dernier enfant de la liste des enfants du nœud. La 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 la méthodes `appendChild()`, `insertBefore()` et `removeNode()` pour manipuler les nœuds enfants.

Disponibilité

Flash Lite 2.0

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 aussi

`appendChild (méthode XMLNode.appendChild)`, `insertBefore (méthode XMLNode.insertBefore)`, `removeNode (méthode XMLNode.removeNode)`, `hasXMLSocket (propriété capabilities.hasXMLSocket)`

nextSibling (propriété XMLNode.nextSibling)

```
public nextSibling : XMLNode [read-only]
```

Valeur XMLNode qui fait référence au frère suivant de la liste des enfants du nœud. La valeur de la propriété est `null` si le nœud n'est pas suivi par un nœud frère. Cette propriété ne peut pas être utilisée pour manipuler les nœuds enfants ; utilisez la méthodes `appendChild()`, `insertBefore()` et `removeNode()` pour manipuler les nœuds enfants.

Disponibilité

Flash Lite 2.0

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 aussi

`firstChild (propriété XMLNode.firstChild)`, `appendChild (méthode XMLNode.appendChild)`, `insertBefore (méthode XMLNode.insertBefore)`, `removeNode (méthode XMLNode.removeNode)`, `hasXMLSocket (propriété capabilities.hasXMLSocket)`

nodeName (propriété XMLNode.nodeName)

```
public nodeName : String
```

Chaîne représentant le nom du nœud de l'objet XML. Si l'objet XML est un élément XML (`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 HTML `TITLE`. Si l'objet XML est un nœud de texte (`nodeType == 3`), nodeName est `null`.

Disponibilité
Flash Lite 2.0

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 aussi
nodeType (propriété XMLNode.nodeType)

nodeType (propriété XMLNode.nodeType)

```
public nodeType : Number [read-only]
```

Valeur `nodeType`, 1 pour un élément XML ou 3 pour un nœud texte.

`nodeType` est une valeur numérique définie d'après l'énumération NodeType spécifiée dans la recommandation DOM Level 1 du W3C à l'adresse www.w3.org. Le tableau suivant répertorie les valeurs :

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 Lite Player, la classe XML intégrée prend uniquement en charge les types de nœud 1 (`ELEMENT_NODE`) et 3 (`TEXT_NODE`).

Disponibilité

Flash Lite 2.0

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 aussi

nodeValue (propriété XMLNode.nodeValue)

nodeValue (propriété XMLNode.nodeValue)

```
public nodeValue : String
```

Valeur du nœud de l'objet XML. Si l'objet XML est un nœud texte, `nodeType` est 3 et `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

Flash Lite 2.0

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 perluète (&).

```
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 aussi

nodeType (propriété XMLNode.nodeType)

parentNode (propriété XMLNode.parentNode)

```
public parentNode : XMLNode [read-only]
```

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

Flash Lite 2.0

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);

// output (line breaks added for clarity):

the parent node of 'username' is: login
contents of the parent node are:
    morton
    good&evil
```

Voir aussi

appendChild (méthode XMLNode.appendChild), insertBefore (méthode XMLNode.insertBefore), removeNode (méthode XMLNode.removeNode), hasXMLSocket (propriété capabilities.hasXMLSocket)

previousSibling (propriété XMLNode.previousSibling)

```
public previousSibling : XMLNode [read-only]
```

Valeur XMLNode qui fait référence au frère précédent de la liste des enfants du nœud. La valeur de la propriété est `null` si le nœud n'a pas de nœud frère précédent. Cette propriété ne peut pas être utilisée pour manipuler les nœuds enfants ; utilisez la méthodes `appendChild()`, `insertBefore()` et `removeNode()` pour manipuler les nœuds enfants.

Disponibilité
Flash Lite 2.0

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 aussi
lastChild (propriété XMLNode.lastChild), appendChild (méthode XMLNode.appendChild), insertBefore (méthode XMLNode.insertBefore), removeNode (méthode XMLNode.removeNode), hasXMLSocket (propriété capabilities.hasXMLSocket)

removeNode (méthode XMLNode.removeNode)

```
public removeNode() : Void
```

Supprime l'objet XML spécifié de son parent. Supprime également tous les descendants du nœud.

Disponibilité
Flash Lite 2.0

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 (méthode XMLNode.toString)

```
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 premier niveau (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 définie sur `undefined`.

Disponibilité
Flash Lite 2.0

Valeur renvoyée
String - String.

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());
        // <LAST>NG<
```

Voir aussi
docTypeDecl (propriété XML.docTypeDecl), xmlDecl (propriété XML.xmlDecl)

XMLSocket

```
Object
    |
    +-XMLSocket

public class XMLSocket
extends Object
```

La classe XMLSocket implémente les sockets client qui permettent au périphérique sur lequel le lecteur Flash Lite s'exécute de communiquer avec un 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. Un système 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 processus de type démon qui comprend le protocole utilisé par la classe XMLSocket. La liste suivante décrit le protocole :

- 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 nul (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 :

- Pour connecter un objet XMLSocket à un port dont le numéro est inférieur à 1024, vous devez tout d'abord charge un fichier de régulation avec la méthode `System.security.loadPolicyFile()`, même lorsque votre application se connecte à son propre domaine.
- 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 de la fonction `loadVariables()` et des méthodes `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 autorise l'accès 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 les méthodes de 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, `XMLSocket`, pour créer un objet XMLSocket.

Disponibilité

Flash Lite 2.1

Voir aussi

`loadPolicyFile (méthode security.loadPolicyFile)`

Résumé des propriétés

Propriétés héritées de la classe Object

`constructor (propriété Object.constructor), __proto__ (Object.__proto__, propriété) prototype (propriété Object.prototype), __resolve (Object.__resolve, propriété)`

Résumé des événements

Evénement	Description
`onClose = fonction() {}`	Appelé uniquement lorsque le serveur ferme une connexion.
`onConnect = function(success:Boolean) {}`	Appel asynchrone que le lecteur Flash Lite effectue lorsqu'une demande de connexion effectuée à l'aide de `XMLSocket.connect()` réussit ou échoue.
`onData = function(src:String) {}`	Appelé lorsqu'un message est téléchargé à partir du serveur et se termine par un octet nul (0).
`onXML = function(src:XML) {}`	Appelé par le lecteur Flash Lite 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 à l'hôte Internet spécifié à l'aide du port TCP spécifié et renvoie true ou false, selon que la connexion est 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 nul (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 (méthode XMLSocket.close)

```
public close() : Void
```

Ferme la connexion spécifiée par l'objet XMLSocket.

Disponibilité

Flash Lite 2.1

Exemple

L'exemple simple suivant crée un objet XMLSocket, tente de se connecter au serveur, puis met fin à la connexion.

```
var socket:XMLSocket = new XMLSocket();
socket.connect(null, 2000);
socket.close();
```

Voir aussi

connect (méthode XMLSocket.connect)

connect (méthode XMLSocket.connect)

```
public connect(url:String, port:Number) : Boolean
```

Etablit une connexion à l'hôte Internet spécifié à l'aide du port TCP spécifié et renvoie true ou false, selon que la connexion est établie ou non. Si la méthode 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.

Si vous ne connaissez pas le numéro de port de votre ordinateur hôte Internet, contactez votre administrateur réseau. Pour connecter un objet XMLSocket à un port dont le numéro est inférieur à 1024, vous devez tout d'abord charger un fichier de régulation à l'aide de la méthode System.security.loadPolicyFile().

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.example.com, le fait de spécifier `null` pour le paramètre hôte équivaut à entrer l'adresse IP de www.example.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 ne peut charger des variables qu'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 interdomaine* sur le serveur hébergeant le fichier SWF qui est utilisé.

Remarque : La méthode `XMLSocket.connect()` renvoie `false` si `System.capabilities.hasXMLSocket` est défini sur `false`.

Disponibilité
Flash Lite 2.1

Paramètres
url : String - Chaîne ; nom de domaine DNS entièrement qualifié 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, le paramètre `host` doit appartenir au même domaine que le fichier SWF.

port : Number - Nombre ; numéro de port TCP de l'hôte utilisé pour établir une connexion.

Valeur renvoyée
Boolean - Valeur `true` si la connexion est établie ; `false` sinon.

Exemple
L'exemple suivant utilise la méthode `XMLSocket.connect()` pour la connexion à l'hôte hébergeant le fichier SWF et utilise la fonction `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 aussi
onConnect (gestionnaire XMLSocket.onConnect), Array, fonction, loadPolicyFile (méthode security.loadPolicyFile)

onClose (gestionnaire XMLSocket.onClose)

```
onClose = function() {}
```

Appelé uniquement lorsque le serveur ferme une connexion. 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
Flash Lite 2.1

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 aussi
`onConnect (gestionnaire XMLSocket.onConnect)`, `Array`, `fonction`

onConnect (gestionnaire XMLSocket.onConnect)

```
onConnect = function(success:Boolean) {}
```

Appel asynchrone que le lecteur Flash Lite effectue lorsqu'une demande de connexion effectuée à l'aide de `XMLSocket.connect()` réussit ou échoue. Si la connexion aboutit, le paramètre `success` a la valeur `true` ; sinon, le paramètre `success` a la valeur `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
Flash Lite 2.1

Paramètres
success : Boolean - Valeur booléenne indiquant si une connexion au socket est établie. Si la connexion aboutit, le paramètre `success` a la valeur `true` ; sinon, le paramètre `success` a la valeur `false`.

Exemple
L'exemple suivant illustre le processus de spécification d'une fonction de remplacement pour le gestionnaire d'événements `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");
    }
}
```

Une fois le gestionnaire `onConnect()` défini, la méthode connect() est appelée pour tenter d'établir la connexion. Si la méthode `connect()` renvoie la valeur `false`, le fichier SWF est envoyé directement vers une image appelée connectionFailed et `onConnect()` n'est plus appelé. Si `connect()` renvoie `true`, le fichier SWF revient à une image appelée waitForConnection, qui correspond à l'écran d'attente. 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 aussi

`connect (méthode XMLSocket.connect)`, `Array`, `fonction`

onData (gestionnaire XMLSocket.onData)

```
onData = function(src:String) {}
```

Appelé lorsqu'un message est téléchargé à partir du serveur et se termine par un octet nul (0). Vous pouvez contourner le gestionnaire d'événements `XMLSocket.onData` pour intercepter les données que le serveur envoie sans les analyser en tant que code XML. Cette fonctionnalité 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 Lite 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

Flash Lite 2.1

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 à octet nul (0) n'est pas inclus dans la chaîne.

```
XMLSocket.prototype.onData = function (src) {
    this.onXML(new XML(src));
}
```

onXML (gestionnaire XMLSocket.onXML)

```
onXML = function(src:XML) {}
```

Appelé par le lecteur Flash Lite 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 nul (0). Lorsque le lecteur Flash Lite 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

Flash Lite 2.1

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 une application simple de discussion. La fonction `myOnXML()` oblige l'application de discussion à reconnaître un élément XML unique, `MESSAGE`, au format suivant :

```
<MESSAGE USER="John" TEXT="Hello, my name is John!" />
var socket:XMLSocket = new XMLSocket();
```

Dans l'exemple suivant, la fonction `displayMessage()` est considérée comme une fonction définie par l'utilisateur qui affiche le message reçu par l'utilisateur :

```
socket.onXML = function (doc) {
    var e = doc.firstChild;
    if (e != null && e.nodeName == "MESSAGE") {
        displayMessage(e.attributes.user, e.attributes.text);
    }
}
```

Voir aussi

Array, fonction

send (méthode XMLSocket.send)

```
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 nul (0). Si `object` est un objet XML, la chaîne est la représentation textuelle 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.

Lorsque l'objet XMLSocket n'est pas connecté au serveur (avec la méthode `XMLSocket.connect()`), l'opération `XMLSocket.send()` échoue.

Disponibilité
Flash Lite 2.1

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 XML `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 aussi
`connect (méthode XMLSocket.connect)`

Constructeur XMLSocket

```
public XMLSocket()
```

Crée un objet XMLSocket. L'objet XMLSocket n'est initialement pas connecté à un serveur. Vous devez appeler la méthode `XMLSocket.connect()` pour établir la connexion.

Disponibilité
Flash Lite 2.1

Exemple
L'exemple suivant crée un objet XMLSocket :

```
var socket:XMLSocket = new XMLSocket();
```

Chapitre 3 : Code ActionScript déconseillé

L'évolution de ActionScript a rendu caducs de nombreux éléments du langage. Cette section liste les articles déconseillés et suggère d'autres possibilités quand cela est possible. Bien que des éléments déconseillés fonctionnent encore dans Flash Player 2.0 et versions ultérieures, Adobe recommande de ne plus les employer dans votre code. Le support futur des éléments déconseillés n'est pas garanti.

Fonctions déconseillées

Modificateurs	Nom de la fonction	Description
	call(frame:Object)	Déconseillé depuis Flash Player 5. Cette action est déconseillée au profit de l'instruction `function`.
	chr(number:Number)String	Déconseillé depuis Flash Player 5. Cette fonction est déconseillée au profit de `String.fromCharCode()`.
	getProperty(my_mc:Object, property:Object)Object	Déconseillé depuis Flash Player 5. Cette fonction est déconseillée depuis Flash Player 5, au profit de la syntaxe à point.
	ifFrameLoaded([scene:String], frame:Object, statement(s):Object)	Déconseillé depuis Flash Player 5. Adobe recommande d'employer la propriété `MovieClip._framesloaded`.
	int(value:Number)Number	Déconseillé depuis Flash Player 5. Cette fonction est déconseillée au profit de `Math.round()`.
	length(expression:String, variable:Object)Number	Déconseillé depuis Flash Player 5. Cette fonction, de même que les fonctions de chaîne, est déconseillée. Adobe recommande d'employer les méthodes de la classe String et la propriété `String.length` pour effectuer les mêmes opérations.
	mbchr(number:Number)	Déconseillé depuis Flash Player 5. Cette fonction est déconseillée au profit de la méthode `String.fromCharCode()`.
	mblength(string:String)Number	Déconseillé depuis Flash Player 5. Cette fonction est déconseillée au profit de la propriété `String.length`.
	mbord(character:String)Number	Déconseillé depuis Flash Player 5. Cette fonction est déconseillée au profit de la méthode `String.charCodeAt()`.
	mbsubstring(value:String, index:Number, count:Number)String	Déconseillé depuis Flash Player 5. Cette fonction est déconseillée au profit de la méthode `String.substr()`.
	ord(character:String)Number	Déconseillé depuis Flash Player 5. Cette fonction est déconseillée au profit des méthodes et des propriétés de la classe String.
	random(value:Number)Number	Déconseillé depuis Flash Player 5. Cette fonction est déconseillée au profit de `Math.random()`.

Modificateurs	Nom de la fonction	Description
	substring(string:String, index:Number, count:Number)String	Déconseillé depuis Flash Player 5. Cette fonction est déconseillée au profit de `String.substr()`.
	tellTarget(target:String, statement(s):Object)	Déconseillé depuis Flash Player 5. Adobe recommande d'utiliser une notation de type point (.) et l'instruction `with`.
	toggleHighQuality()	Déconseillé depuis Flash Player 5. Cette fonction est déconseillée au profit de `_quality`.

Propriétés déconseillées

Modificateurs	Nom de la propriété	Description
	$version	Déconseillé depuis Flash Lite Player 2.0. Cette action est déconseillée au profit de la propriété `System.capabilities.version`.
	_cap4WayKeyAS	Déconseillé depuis Flash Lite Player 2.0. Cette action est déconseillée au profit de la propriété `System.capabilities.has4WayKeyAS`.
	_capCompoundSound	Déconseillé depuis Flash Lite Player 2.0. Cette action est déconseillée au profit de la propriété `System.capabilities.hasCompoundSound`.
	_capEmail	Déconseillé depuis Flash Lite Player 2.0. Cette action est déconseillée au profit de la propriété `System.capabilities.hasEmail`.
	_capLoadData	Déconseillé depuis Flash Lite Player 2.0. Cette action est déconseillée au profit de la propriété `System.capabilities.hasDataLoading`.
	_capMFi	Déconseillé depuis Flash Lite Player 2.0. Cette action est déconseillée au profit de la propriété `System.capabilities.hasMFi`.
	_capMIDI	Déconseillé depuis Flash Lite Player 2.0. Cette action est déconseillée au profit de la propriété `System.capabilities.hasMIDI`.
	_capMMS	Déconseillé depuis Flash Lite Player 2.0. Cette action est déconseillée au profit de la propriété `System.capabilities.hasMMS`.
	_capSMAF	Déconseillé depuis Flash Lite Player 2.0. Cette action est déconseillée au profit de la propriété `System.capabilities.hasSMAF`.
	_capSMS	Déconseillé depuis Flash Lite Player 2.0. Cette action est déconseillée au profit de la propriété `System.capabilities.hasSMS`.
	_capStreamSound	Déconseillé depuis Flash Lite Player 2.0. Cette action est déconseillée au profit de la propriété `System.capabilities.hasStreamingAudio`.
	Bouton._highquality	Déconseillé depuis Flash Player 7. Cette propriété est déconseillée au profit de `Button._quality`.

Modificateurs	Nom de la propriété	Description
	MovieClip._highquality	Déconseillé depuis Flash Player 7. Cette propriété est déconseillée au profit de `MovieClip._quality`.
	TextField._highquality	Déconseillé depuis Flash Player 7. Cette propriété est déconseillée au profit de `TextField._quality`.
	_highquality	Déconseillé depuis Flash Player 5. Cette propriété est déconseillée au profit de `_quality`.
	maxscroll	Déconseillé depuis Flash Player 5. Cette propriété est déconseillée au profit de `TextField.maxscroll`.
	scroll	Déconseillé depuis Flash Player 5. Cette propriété est déconseillée au profit de `TextField.scroll`.

Opérateurs déconseillés

Opérateur	Description
<> (inégalité)	Déconseillé depuis Flash Player 5. Adobe recommande d'utiliser l'opérateur `!=` (inégalité).
add (concaténation (chaînes))	Déconseillé depuis Flash Player 5. Adobe recommande d'utiliser l'opérateur d'addition (+) lorsque vous créez du contenu pour Flash Player 5 ou version ultérieure. **Remarque** : Flash Lite 2.0 remplace également l'opérateur `add` au profit de l'opérateur d'addition (+).
and (AND logique)	Déconseillé depuis Flash Player 5. Adobe recommande d'utiliser l'opérateur logique AND (`&&`) .
eq (égalité (chaînes))	Déconseillé depuis Flash Player 5. Cet opérateur est déconseillé au profit de l'opérateur `==` (égalité).
ge (supérieur ou égal à (chaînes))	Déconseillé depuis Flash Player 5. Cet opérateur est déconseillé au profit de l'opérateur `>=` (supérieur ou égal à).
gt (supérieur à (chaînes))	Déconseillé depuis Flash Player 5. Cet opérateur est déconseillé au profit de l'opérateur `>` (supérieur à).
le (inférieur ou égal à (chaînes))	Déconseillé depuis Flash Player 5. Cet opérateur est déconseillé dans Flash 5 au profit de l'opérateur `<=` (inférieur ou égal à).
lt (inférieur à (chaînes))	Déconseillé depuis Flash Player 5. Cet opérateur est déconseillé au profit de l'opérateur `>` (supérieur à).
ne (inégalité (chaînes))	Déconseillé depuis Flash Player 5. Cet opérateur est déconseillé au profit de l'opérateur `!=` (inégalité).
not (NOT logique)	Déconseillé depuis Flash Player 5. Cet opérateur est déconseillé au profit de l'opérateur `!` (NOT logique), opérateur.
or (OR logique)	Déconseillé depuis Flash Player 5. Cet opérateur est déconseillé au profit de l'opérateur `||` (OU logique).

Chapitre 4 : Eléments de code ActionScript non pris en charge

Les listes ci-dessous présentent les classes, méthodes, propriétés, fonctions globales, gestionnaires d'événements et fscommands pris en charge par ActionScript 2.0, mais non pris en charge par une version de Flash Lite. Pour de plus amples informations, reportez-vous aux éléments de langage et classes ActionScript pris en charge intégralement ou partiellement, et non pris en charge par les applications de développement Adobe Flash Lite 2.x et 3.x.

Classes non prises en charge

Accessibility, BevelFilter, BitmapFilter, BlurFilter, Camera, ColorMatrixFilter, ContextMenu, ContextMenuItem, ConvolutionFilter, CustomActions, DisplacementMapFilter, DropShadowFilter, FileReference, FileReferenceList, GlowFilter, GradientBevelFilter, GradientGlowFilter, IME, Locale (mx.lang.Locale), Microphone, PrintJob, TextField.StyleSheet, TextRenderer, TextSnapshot, XMLUI

Méthodes non prises en charge

BitmapData.applyFilter, BitmapData.generateFilterRect, BitmapData.noise, BitmapData.paletteMap, BitmapData.perlinNoise, BitmapData.pixelDissolve, BitmapData.scroll, BitmapData.threshold, Matrix.createGradientBox, Mouse.hide, Mouse.show, MovieClip.attachAudio, MovieClip.getTextSnapshot, Selection.getBeginIndex, Selection.getCaretIndex, Selection.getEndIndex, System.setClipboard, System.showSettings, TextField.getFontList, Video.clear

Propriétés non prises en charge

Button.blendMode, Button.cacheAsBitmap, Button.filters, Button.menu, Button.useHandCursor, System.capabilities.language, System.capabilities.manufacturer, System.capabilities.pixelAspectRatio, System.capabilities.playerType, System.capabilities.screenColor, System.capabilities.screenDPI, System.capabilities.serverString, Key.isToggled, MovieClip.menu, MovieClip.useHandCursor, Stage.showMenu, System.exactSettings, TextField.menu, TextField.mouseWheelEnabled, TextField.restrict, Video._alpha, Video.deblocking, Video._height, Video.height, Video._name, Video._parent, Video._rotation, Video.smoothing, Video._visible, Video._width, Video.width, Video._x, Video._xmouse, Video._xscale, Video._y, Video._ymouse, Video._yscale

Fonctions globales non prises en charge

asfunction, MMExecute, print, printAsBitmap, printAsBitmapNum, printNum, updateAfterEvent

Gestionnaires d'événements non pris en charge

onUpdate, Mouse.onMouseWheel

Commandes fs non prises en charge

allowscale, exec, fullscreen, quit, showmenu, trapallkeys

Index