MX

macromedia®
FLASH™MX
2004

Guide de référence ActionScript

Remerciements

Directeur : Erick Vera

Gestion du projet : Stephanie Gowin, Barbara Nelson

Rédaction : Jody Bleyle, Mary Burger, Kim Diezel, Stephanie Gowin, Dan Harris, Barbara Herbert, Barbara Nelson, Shirley Ong, Tim Statler

Rédactrice en chef : Rosana Francescato

Révision : Linda Adler, Mary Ferguson, Mary Kraemer, Noreen Maher, Antonio Padial, Lisa Stanziano, Anne Szabla

Gestion de la production : Patrice O'Neill

Conception et production des supports : Adam Barnett, Christopher Basmajian, Aaron Begley, John Francis, Jeff Harmon

Localisation : Tim Hussey, Seungmin Lee, Masayo Noda, Simone Pux, Yuko Yagi, Florian de Joannès

Première édition : Octobre 2003

Macromedia, Inc.
600 Townsend St.
San Francisco, CA 94103

TABLE DES MATIERES

PARTIE V : Référence

INTRODUCTION
Bien démarrer avec ActionScript

Macromedia Flash MX 2004 et Flash MX Professionnel 2004 sont les outils standard des professionnels pour la création de contenu web percutant. ActionScript est le langage que vous utilisez pour développer une application dans Flash. Vous pouvez très bien vous servir de Flash sans ActionScript. Toutefois, si l'interactivité utilisateur est une de vos priorités ou si vous voulez utiliser des objets autres que ceux qui sont intégrés dans Flash (tels les boutons et les clips) ou créer des applications plus complexes à partir de fichiers SWF, il est probable que vous aurez recours à ActionScript.

Public visé

Ce manuel présume que vous avez déjà installé Flash MX 2004 ou Flash MX Professionnel 2004 et que vous savez comment l'utiliser. Il considère également comme acquis que vous savez placer des objets sur la scène et les manipuler dans l'environnement auteur de Flash. Si vous avez déjà écrit des programmes, vous vous sentirez en terrain connu avec ActionScript. Et si vous êtes un novice, vous n'aurez aucun mal à faire son apprentissage. Le plus simple consiste à commencer par les commandes de base et à aborder les éléments plus complexes progressivement.

Configuration système requise

La configuration système requise par ActionScript est identique à celle de Flash MX 2004 ou Flash MX Professionnel 2004. Toutefois, la documentation suppose que vous utilisez les paramètres de publication par défaut pour vos fichiers Flash : Flash Player 7 et ActionScript 2.0. Si vous modifiez ces paramètres, les explications et les exemples de code contenus dans la documentation risquent de ne pas fonctionner correctement.

Utilisation de la documentation

Ce manuel passe en revue les principes généraux de la syntaxe ActionScript, explique comment utiliser ce langage pour intervenir sur différents types d'objet et décrit en détail la syntaxe et l'utilisation de chaque élément. Commencez par vous familiariser avec la terminologie et les concepts de base utilisés dans le reste du manuel (consultez le Chapitre 2, *Notions de base du langage ActionScript*, page 27). Vous pourrez ensuite vous concentrer sur l'écriture et le débogage de scripts Flash (consultez le Chapitre 3, *Rédaction et débogage de scripts*, page 59).

Avant de créer vos propres scripts, suivez les leçons « Rédiger des scripts avec ActionScript » et « Créer un formulaire contenant une logique conditionnelle et envoyer des données », qui constituent une introduction interactive à l'utilisation d'ActionScript. Pour localiser ces leçons, sélectionnez Aide > Comment > Manuel de prise en main rapide.

Une fois que vous maîtrisez les concepts de base, vous pouvez rechercher, dans le reste du manuel, les informations qui vous permettront d'obtenir l'effet que vous recherchez. Si, par exemple, vous voulez savoir comment écrire un script qui exécute une action spécifique lorsqu'un utilisateur clique avec la souris, consultez le Chapitre 4, *Gestion d'événements*, page 89.

Lorsque vous trouvez des informations sur une commande qui vous intéresse, vous pouvez consulter l'entrée correspondante dans le Chapitre 12, *Dictionnaire ActionScript*, page 215 ; tous les éléments du langage y sont répertoriés par ordre alphabétique.

Conventions typographiques

Ce manuel utilise les conventions typographiques suivantes :

- `La police de code` indique le code ActionScript.
- *`La police de code en italique`* identifie un élément, tel un paramètre ou un nom d'objet ActionScript, que vous remplacez par votre propre texte lorsque vous créez un script.

Terminologie utilisée dans ce manuel

Ce manuel utilise les termes suivants :

- *Vous* fait référence au programmeur qui écrit un script ou une application.
- *L'utilisateur* désigne la personne qui exécute vos scripts et applications.
- *Compilation* correspond au moment auquel vous publiez, exportez, testez ou déboguez votre document.
- *Exécution* représente le moment auquel votre script s'exécute dans Flash Player.

Les termes ActionScript tels que *méthode* et *objet* sont définis dans le Chapitre 2, *Notions de base du langage ActionScript*, page 27.

Ressources supplémentaires

La documentation spécifique à Flash et aux produits apparentés est disponible séparément.

- Pour plus d'informations sur l'utilisation de l'environnement auteur de Flash, consultez le guide Utilisation de Flash de l'aide. Pour en savoir plus sur l'utilisation des composants, consultez le guide Utilisation des composants de l'aide.
- Pour plus d'informations sur la création d'applications de communication, consultez les leçons « *Développement d'applications de communication* » et « *Gestion du serveur de communications Flash* ».
- Pour plus d'informations sur l'accès aux services web avec des applications Flash, consultez *Using Flash Remoting*.

Le site web Macromedia DevNet (www.macromedia.com/go/developer_fr) est régulièrement actualisé et propose les informations les plus récentes sur Flash, ainsi que des conseils d'utilisateurs expérimentés, des rubriques avancées, des exemples, des astuces et d'autres mises à jour. Consultez régulièrement ce site web pour vous tenir au courant des nouveautés de Flash et tirer le meilleur parti de votre programme.

Le Centre de support Macromedia Flash (www.macromedia.com/go/flash_support_fr) fournit des TechNotes, des mises à jour de la documentation et des liens vers des ressources supplémentaires dans la communauté Flash.

PARTIE I
Bienvenue dans ActionScript

Cette section inclut des informations élémentaires sur le langage ActionScript.

Le Chapitre 1 comprend des informations concernant les nouveautés ou les modifications apportées à ActionScript et Flash Player 7. Si vous avez déjà utilisé ActionScript, veillez à lire attentivement ces informations.

Si vous n'avez jamais utilisé ActionScript, les chapitres 2 et 3 contiennent les bases qui vous permettront de comprendre la terminologie et la syntaxe d'ActionScript et de savoir écrire et déboguer vos scripts.

CHAPITRE 1
Nouveautés du langage ActionScript dans Flash MX 2004

Macromedia Flash MX 2004 et Macromedia Flash MX Professionnel 2004 bénéficient de plusieurs améliorations pour vous permettre d'écrire facilement des scripts plus robustes à l'aide du langage ActionScript. Ces améliorations, décrites dans ce chapitre, portent notamment sur de nouveaux éléments de langage, les outils de débogage et d'édition (consultez *Modifications de l'éditeur ActionScript*, page 23 et *Modifications apportées dans le domaine du débogage*, page 24), ainsi que l'utilisation d'un modèle de programmation plus orienté objet (consultez *Nouveau modèle de programmation orientée objet*, page 25).

Ce chapitre contient également une section détaillée à lire attentivement si vous projetez de publier l'un de vos fichiers existants Flash MX ou d'une version antérieure dans Flash Player 7 (consultez *Portage de scripts existants sur Flash Player 7*, page 17).

Eléments de langage nouveaux et modifiés

Cette section décrit les éléments du langage ActionScript qui sont nouveaux ou ont été modifiés dans Flash MX 2004. Pour pouvoir utiliser ces éléments dans vos scripts, vous devez utiliser Flash Player 7 (lecteur par défaut) lors de la publication des documents.

- Les méthodes `Array.sort` et `Array.sortOn()` vous permettent d'ajouter des paramètres pour spécifier des options de tri supplémentaires, par exemple un tri ascendant ou descendant, le respect de la casse lors du tri et bien d'autres options encore.
- Les propriétés `Button.menu`, `MovieClip.menu` et `TextField.menu` se conjuguent aux nouvelles classes ContextMenu et ContextMenuItem pour vous permettre d'associer des éléments de menu contextuel à des objets Button, MovieClip ou TextField.
- La *Classe ContextMenu* et la *Classe ContextMenuItem* vous permettent de personnaliser le menu contextuel qui s'affiche lorsqu'un utilisateur clique avec le bouton droit de la souris (Windows) ou en appuyant sur la touche Contrôle (Macintosh) dans Flash Player.
- La *Classe Error* et les commandes `throw` et `try..catch..finally` vous permettent d'implémenter une gestion des exceptions plus robuste.
- Les méthodes `LoadVars.addRequestHeader()` et `XML.addRequestHeader()` ajoutent ou modifient les en-têtes de requête HTTP (tels que « `Content-Type` » ou « `SOAPAction` ») envoyés avec les actions `POST`.
- La fonction `MMExecute()` vous permet d'émettre des commandes de l'API Flash JavaScript à partir d'ActionScript.

- (Windows uniquement) L'écouteur d'événement `Mouse.onMouseWheel` est généré lorsque l'utilisateur effectue un défilement en utilisant la molette de la souris.
- La méthode `MovieClip.getNextHighestDepth()` vous permet de créer des occurrences de MovieClip au moment de l'exécution et de vous assurer que leurs objets seront placés devant les autres objets dans l'espace *z* d'un clip parent. La méthode `MovieClip.getInstanceAtDepth()` vous permet d'accéder à des occurrences de MovieClip créées dynamiquement en utilisant la profondeur comme index de recherche.
- La méthode `MovieClip.getSWFVersion()` vous permet de déterminer quelle version de Flash Player est supportée par un fichier SWF chargé.
- Les méthodes `MovieClip.getTextSnapshot()` et l'*Objet TextSnapshot* vous permettent d'intervenir sur du texte qui figure dans des champs de texte statiques dans un clip.
- La propriété `MovieClip._lockroot` vous permet d'indiquer qu'un clip agira en tant que `_root` pour tout clip qui y sera chargé, ou que la signification de `_root` dans un clip ne sera pas modifiée si le clip est chargé dans un autre clip.
- La *Classe MovieClipLoader* vous permet de contrôler la progression du téléchargement de fichiers dans des clips.
- La *Classe NetConnection* et la *Classe NetStream* vous permettent de lire en continu des fichiers vidéo en local (fichiers FLV).
- La *Classe PrintJob* vous donne (ainsi qu'à l'utilisateur) davantage de contrôle sur les impressions à partir de Flash Player.
- Le gestionnaire d'événement `Sound.onID3` donne accès aux données ID3 associées à un objet Sound contenant un fichier MP3.
- La propriété `Sound.ID3` donne accès aux métadonnées faisant partie d'un fichier MP3.
- La *Classe System* comporte de nouveaux objets et méthodes et l'*Objet System.capabilities* possède plusieurs propriétés nouvelles.
- La propriété `TextField.condenseWhite` vous permet de supprimer l'espace blanc supplémentaire dans les champs de texte HTML qui sont rendus dans un navigateur.
- La propriété `TextField.mouseWheelEnabled` vous permet de spécifier si le contenu d'un champ de texte doit défiler lorsque le pointeur de la souris est placé dessus et que l'utilisateur actionne la molette.
- La *Classe TextField.StyleSheet* vous permet de créer un objet feuille de style contenant des règles de formatage de texte, comme la taille et la couleur de la police, et d'autres styles de formatage.
- La propriété `TextField.styleSheet` vous permet d'associer un objet feuille de style à un champ de texte.
- La méthode `TextFormat.getTextExtent()` accepte un nouveau paramètre et l'objet qu'elle renvoie contient un nouveau membre.
- La méthode `XML.addRequestHeader()` vous permet d'ajouter ou de modifier les en-têtes de requête HTTP (tels que « `Content-Type` » ou « `SOAPAction` ») envoyés avec les actions `POST`.

Nouveau modèle de sécurité et fichiers SWF hérités

Les règles qui permettent à Flash Player de déterminer si deux domaines sont identiques ont changé dans Flash Player 7. Les règles qui déterminent si, et de quelle façon, un fichier SWF servi par un domaine HTTP peut accéder à un fichier SWF ou charger des données à partir d'un domaine HTTPS ont également changé. Dans la plupart des cas, ces modifications n'ont pas d'incidence, à moins que vous ne portiez vos fichiers SWF existants sur Flash Player 7.

Toutefois, si vous avez publié, pour Flash Player 6 ou une version antérieure, des fichiers SWF qui chargent des données issues d'un fichier résidant sur un serveur et que le fichier SWF à l'origine de l'appel s'exécute dans Flash Player 7, une nouvelle boîte de dialogue invitant l'utilisateur à autoriser l'accès est susceptible de s'afficher. Vous pouvez empêcher l'affichage de cette boîte de dialogue en mettant en œuvre un *fichier de régulation* sur le site contenant les données. Pour plus d'informations sur cette boîte de dialogue, consultez *A propos de la compatibilité avec les précédents modèles de sécurité Flash Player*, page 202.

Il peut également être nécessaire d'implémenter un fichier de régulation si vous utilisez des bibliothèques partagées à l'exécution. Si le fichier SWF chargé ou en chargement est publié pour Flash Player 7 et que les fichiers chargés ou en chargement ne sont pas servis par le même domaine, utilisez un fichier de régulation pour autoriser l'accès. Pour plus d'informations sur les fichiers de régulation, consultez *A propos de l'autorisation de chargement de données inter-domaines*, page 201.

Portage de scripts existants sur Flash Player 7

A l'instar de toute nouvelle version, Flash Player 7 prend en charge un plus grand nombre de commandes ActionScript que ses versions précédentes. Ces commandes vous permettent d'implémenter des scripts plus robustes. (Consultez *Eléments de langage nouveaux et modifiés*, page 15.) Toutefois, si vous avez utilisé l'une de ces commandes dans vos scripts existants, il est possible que le script ne fonctionne pas correctement si vous le publiez pour Flash Player 7.

Par exemple, si l'un de vos scripts contient une fonction nommée Error, le script peut sembler se compiler correctement mais ne pas être exécuté comme prévu dans Flash Player 7, car Error est désormais une classe intégrée (et donc un mot réservé) dans ActionScript. Vous pouvez corriger votre script en renommant la fonction Error, par exemple en ConditionErreur.

De plus, Flash Player 7 implémente différentes modifications qui affectent la façon dont un fichier SWF accède à un autre fichier SWF, la façon dont les données externes peuvent être chargées et le mode d'accès aux paramètres et données locaux (par exemple les paramètres de contrôle de l'accès et les objets partagés localement). Enfin, le comportement de certaines fonctions existantes a été modifié.

Si vous voulez publier, pour Flash Player 7, des scripts destinés à Flash Player 6 ou une version antérieure, vous devrez peut-être les modifier afin qu'ils se conforment à l'implémentation de Flash Player 7 et fonctionnent correctement. Les modifications requises sont décrites dans cette section.

Conformité à la norme ECMA-262 version 4

Flash Player 7 a subi plusieurs modifications pour mieux se conformer à la norme ECMA-262 version 4 (consultez la page www.mozilla.org/js/language/es4/index.html). Outre les techniques de programmation basées sur les classes disponibles dans ActionScript 2,0 (consultez *Nouveau modèle de programmation orientée objet*, page 25), d'autres fonctions ont été ajoutées et certains comportements modifiés. De plus, lorsque vous effectuez une publication pour Flash Player 7 et que vous utilisez ActionScript 2.0, vous pouvez attribuer un type d'objet à un autre. Pour plus d'informations, consultez *Attribution d'objets*, page 42. Vous n'êtes pas obligé d'actualiser vos scripts existants. Toutefois, il est souhaitable d'utiliser ces fonctions si vous publiez, pour Flash Player 7, des scripts que vous modifierez et enrichirez par la suite.

Contrairement aux modifications mentionnées plus haut, les modifications répertoriées dans le tableau suivant (dont certaines renforcent la conformité à la norme ECMA) sont susceptibles de changer le comportement des scripts existants. Si vous utilisez ces fonctions dans des scripts existants que vous avez l'intention de publier pour Flash Player 7, passez les modifications en revue pour déterminer si le code fonctionne toujours correctement ou si vous devez le récrire. Plus particulièrement, il est conseillé d'initialiser toutes les variables des scripts que vous portez sur Flash Player 7, car `undefined` est évalué de façon différente dans certains cas.

Fichier SWF publié pour Flash Player 7	Fichier SWF publié pour des versions précédentes de Flash Player
Le respect de la casse est pris en charge (les noms de variable dont la casse est différente sont interprétés comme étant différents). Cette modification a également des répercussions sur les fichiers chargés au moyen de `#include` et les variables externes chargées via `LoadVars.load()`. Pour plus d'informations, consultez *Hauteur de casse*, page 32.	Le respect de la casse n'est pas pris en charge (les noms de variable dont la casse est différente sont interprétés comme étant identiques).

L'évaluation de `undefined` dans un contexte numérique renvoie `NaN`.
`monNombre +=1;`
`trace(monNombre); // NaN`
	L'évaluation de `undefined` dans un contexte numérique renvoie 0.
	`monNombre +=1;`
	`trace(monNombre); // 1`

Lorsque `undefined` est converti en chaîne, le résultat est `undefined`.
`prénom = "Joan ";`
`nom = "Flender";`
`trace(prénom + deuxième prénom + nom);`
`// Joan undefinedFlender`
	Lorsque `undefined` est converti en chaîne, le résultat est "" (une chaîne vide).
	`prénom = "Joan ";`
	`nom = "Flender";`
	`trace(prénom + deuxième prénom + nom);`
	`// Joan Flender`

Fichier SWF publié pour Flash Player 7	Fichier SWF publié pour des versions précédentes de Flash Player
Lorsque vous convertissez une chaîne en valeur booléenne, le résultat est `true` si la longueur de la chaîne est supérieure à zéro et `false` pour une chaîne vide.	Lorsque vous convertissez une chaîne en valeur booléenne, la chaîne est tout d'abord convertie en nombre. Le résultat est `true` si le nombre n'est pas égal à zéro et `false` dans le cas contraire.

Lorsque vous définissez la longueur d'un tableau, seule une chaîne de nombre valide peut définir la longueur. Par exemple, "6" fonctionne, et non " 6" ou "6xyz".
`mon_array=new Array();`
`mon_array["6"] ="x";`
`trace(mon_array.length); // 0`
`mon_array["6xyz"] ="x";`
`trace(mon_array.length); // 0`
`mon_array["6"] ="x";`
`trace(mon_array.length); // 7`
	Lorsque vous définissez la longueur d'un tableau, même une chaîne de nombre mal formulée peut définir la longueur :
	`mon_array=new Array();`
	`mon_array["6"] ="x";`
	`trace(mon_array.length); // 7`
	`mon_array["6xyz"] ="x";`
	`trace(mon_array.length); // 7`
	`mon_array["6"] ="x";`
	`trace(mon_array.length); // 7`

Règles de nom de domaine pour les paramètres et les données locales

Dans Flash Player 6, les règles de correspondance de superdomaine sont utilisées par défaut pour accéder aux paramètres locaux (tels que les autorisations d'accès de la caméra ou du microphone) ou aux données persistantes localement (objets partagés). Cela signifie que les paramètres et données de fichiers SWF résidant sur ici.xyz.com, la.xyz.com et xyz.com sont partagés et tous enregistrés sur xyz.com.

Dans Flash Player 7, les règles de correspondance exacte de domaine sont utilisées par défaut. Cela signifie que les paramètres et données d'un fichier hébergé sur ici.xyz.com sont enregistrés sur ici.xyz.com, les paramètres et données d'un fichier hébergé sur la.xyz.com sont enregistrés sur la.xyz.com et ainsi de suite.

Une nouvelle propriété, `System.exactSettings`, vous permet de spécifier les règles à utiliser. Cette propriété est supportée pour les fichiers publiés pour Flash Player 6 ou une version ultérieure. Pour les fichiers publiés pour Flash Player 6, la valeur par défaut est `false`, ce qui signifie que les règles de correspondance de superdomaine sont utilisées. Pour les fichiers publiés pour Flash Player 7, la valeur par défaut est `true`, ce qui signifie que les règles de correspondance exacte de domaine sont utilisées.

Si vous utilisez des paramètres ou des données locales persistantes et souhaitez publier un fichier SWF Flash Player 6 pour Flash Player 7, il peut être nécessaire de définir cette valeur sur `false` dans le fichier porté.

Pour plus d'informations, consultez `System.exactSettings`, page 755.

Accès interdomaine et accès aux sous-domaines entre les fichiers SWF

Lorsque vous développez une série de fichiers SWF qui communiquent les uns avec les autres (par exemple, lorsque vous utilisez `loadMovie()`, `MovieClip.loadMovie()`, `MovieClipLoader.LoadClip()` ou des objets Local Connection), vous pouvez héberger les animations dans des domaines différents ou dans des sous-domaines différents d'un même superdomaine.

Dans les fichiers publiés pour Flash Player 5 ou une version ultérieure, l'accès aux inter-domaines et sous-domaines n'était soumis à aucune restriction.

Dans les fichiers publiés pour Flash Player 6, vous pouviez utiliser le gestionnaire `LocalConnection.allowDomain` ou la méthode `System.security.allowDomain()` pour spécifier l'accès inter-domaines autorisé (par exemple pour autoriser l'accès à un fichier situé sur unSite.com par un fichier situé sur unAutreSite.com). Aucune commande n'était nécessaire pour autoriser l'accès au superdomaine (par exemple, un fichier situé sur store.unSite.com pouvait accéder à un fichier situé sur www.unSite.com).

Les fichiers publiés pour Flash Player 7 implémentent l'accès entre les fichiers SWF différemment des versions précédentes, et ce de deux manières : tout d'abord, Flash Player 7 implémente les règles de correspondance exacte de domaine et non les règles de correspondance de superdomaine. Ainsi, le fichier auquel on accède (même s'il est publié pour une version antérieure à Flash Player 7) doit autoriser de façon explicite l'accès aux inter-domaines et sous-domaines. Ce sujet est abordé plus bas. Ensuite, un fichier hébergé sur un site qui utilise un protocole sécurisé (HTTPS) doit autoriser de façon explicite l'accès à partir d'un fichier hébergé sur un site utilisant un protocole non sécurisé (HTTP ou FTP). Ce sujet est abordé dans la section suivante (consultez *Accès du protocole HTTP vers le protocole HTTPS entre les fichiers SWF*, page 21).

Etant donné que Flash Player 7 implémente les règles de correspondance exacte de domaine et non les règles de correspondance de superdomaine, il peut être nécessaire de modifier les scripts existants si vous souhaitez y accéder à partir de fichiers publiés pour Flash Player 7. (Vous pouvez toujours publier les fichiers modifiés pour Flash Player 6.) Si vous avez utilisé une instruction `LocalConnection.allowDomain()` ou `System.security.allowDomain()` dans vos fichiers et que vous avez spécifié des sites de superdomaine à autoriser, vous devez modifier vos paramètres pour spécifier les domaines exacts à la place. Le code suivant présente un exemple des types de modifications qu'il peut être nécessaire d'apporter :

```
// commandes Flash Player 6 d'un fichier SWF situé sur www.unAncienSite.com
// pour autoriser l'accès par des fichiers SWF hébergés sur www.unSite.com
// ou sur store.unSite.com
System.security.allowDomain("unSite.com");
ma_lc.allowDomain = function(domaineDenvoi) {
  return(domaineDenvoi=="unSite.com");
}
// Commandes correspondantes pour autoriser l'accès par les fichiers SWF
// qui sont publiés pour Flash Player 7
System.security.allowDomain("www.unSite.com", "store.unSite.com");
ma_lc.allowDomain = function(domaineDenvoi) {
  return(domaineDenvoi=="www.unSite.com" ||
    domaineDenvoi=="store.unSite.com");
}
```

Il peut également être nécessaire d'ajouter des instructions similaires dans vos fichiers si vous ne les utilisez pas actuellement. Par exemple, si votre fichier SWF est hébergé sur www.unSite.com et que vous souhaitez autoriser l'accès par un fichier SWF publié pour Flash Player 7 sur store.unSite.com, vous devez ajouter des instructions semblables aux instructions suivantes au fichier situé sur www.unSite.com (vous pouvez toujours publier le fichier situé sur www.unSite.com pour Flash Player 6) :

```
System.security.allowDomain("store.unSite.com");
ma_lc.allowDomain = function(domaineDenvoi) {
  return(domaineDenvoi=="store.unSite.com");
}
```

Pour résumer, il peut être nécessaire de modifier vos fichiers pour ajouter ou modifier des instructions `allowDomain` si vous publiez des fichiers pour Flash Player 7 dans les conditions suivantes :

- Vous avez implémenté des scripts entre des fichiers SWF (à l'aide de `loadMovie()`, `MovieClip.loadMovie()`, `MovieClipLoader.LoadClip()`, ou d'objets Local Connection).
- Le fichier SWF appelé (de n'importe quelle version) n'est pas hébergé sur un site utilisant un protocole sécurisé (HTTPS), ou le fichier SWF appelé et celui effectuant l'appel sont tous deux hébergés sur des sites HTTPS (si seul le fichier SWF appelé est HTTPS, consultez *Accès du protocole HTTP vers le protocole HTTPS entre les fichiers SWF*, page 21).
- Les fichiers SWF ne se trouvent pas dans le même domaine (par exemple, l'un d'eux réside sur www.domaine.com et l'autre sur store.domaine.com).

Vous devez effectuer les modifications suivantes :

- Si le fichier SWF appelé est publié pour Flash Player 7, incluez `System.security.allowDomain` ou `LocalConnection.allowDomain` dans le fichier SWF appelé, en utilisant la correspondance exacte de domaine.
- Si le fichier SWF appelé est publié pour Flash Player 6, modifiez-le en ajoutant ou en modifiant une instruction `System.security.allowDomain` ou `LocalConnection.allowDomain`, en utilisant la correspondance exacte de domaine, comme illustré dans les exemples de code plus haut dans cette section. Vous pouvez publier le fichier modifié pour flash Player 6 ou 7.
- Si le fichier SWF appelé est publié pour Flash Player 5 ou version antérieure, portez le fichier appelé sur Flash Player 6 ou 7 et ajoutez une instruction `System.security.allowDomain` en utilisant la correspondance exacte de domaine, comme illustré dans les exemples de code plus haut dans cette section. (Les objets LocalConnection ne sont pas supportés dans Flash Player 5 ou version antérieure.)

Accès du protocole HTTP vers le protocole HTTPS entre les fichiers SWF

Comme nous l'avons vu dans la section précédente, les règles d'accès aux inter-domaines et sous-domaines ont été modifiées dans Flash Player 7. En plus des règles de correspondance exacte de domaine dorénavant implémentées, vous devez autoriser de façon explicite l'accès aux fichiers situés sur des sites utilisant un protocole sécurisé (HTTPS) par des fichiers hébergés sur des sites utilisant des protocoles non sécurisés. Selon que votre fichier est publié pour Flash Player 7 ou Flash Player 6, vous devez implémenter l'une des deux instructions `allowDomain` (consultez *Accès interdomaine et accès aux sous-domaines entre les fichiers SWF*, page 19), ou utiliser les nouvelles instructions `LocalConnection.allowInsecure Domain` ou `System.security.allowInsecureDomain()`.

Avertissement : L'implémentation d'une instruction `allowInsecureDomain()` compromet la sécurité assurée par le protocole HTTPS. Vous devez effectuer ces modifications uniquement si vous ne pouvez pas réorganiser votre site de manière à ce que tous les fichiers SWF soient servis par le protocole HTTPS.

Le code suivant présente un exemple des types de modifications qu'il peut être nécessaire d'apporter :

```
// Commandes dans un fichier SWF Flash Player 6 situé sur https://
  www.unSite.com
// pour autoriser l'accès par des fichiers SWF Flash Player 7 hébergés
// sur http://www.unSite.com ou http://www.unAutreSite.com
```

```
System.security.allowDomain("unAutreSite.com");
ma_lc.allowDomain = function(domaineDenvoi) {
  return(domaineDenvoi=="unAutreSite.com");
}
// Commandes correspondantes dans un fichier SWF Flash Player 7
// pour autoriser l'accès par des fichiers SWF Flash Player 7 hébergés
// sur http://www.unSite.com ou http://www.unAutreSite.com
System.security.allowInsecureDomain("www.unSite.com", "www.unAutreSite.com");
ma_lc.allowInsecureDomain = function(domaineDenvoi) {
  return(domaineDenvoi=="www.unSite.com" ||
    domaineDenvoi=="www.unAutreSite.com");
}
```

Il peut également être nécessaire d'ajouter des instructions similaires dans vos fichiers si vous ne les utilisez pas actuellement. Une modification peut s'avérer nécessaire même si les deux fichiers résident dans le même domaine (par exemple, un fichier situé sur http://www.domaine.com appelle un fichier situé sur https://www.domaine.com).

Pour résumer, il peut être nécessaire de modifier vos fichiers pour ajouter ou modifier des instructions si vous publiez des fichiers pour Flash Player 7 qui correspondent aux critères suivants :

- Vous avez implémenté des scripts entre des fichiers SWF (à l'aide de `loadMovie()`, `MovieClip.loadMovie()`, `MovieClipLoader.LoadClip()`, ou d'objets Local Connection).
- Le fichier effectuant l'appel n'est pas hébergé à l'aide d'un protocole HTTPS et le fichier appelé est HTTPS.

Vous devez effectuer les modifications suivantes :

- Si le fichier appelé est publié pour Flash Player 7, incluez `System.security.allowInsecureDomain` ou `LocalConnection.allowInsecureDomain` dans le fichier appelé en utilisant la correspondance exacte de domaine, comme illustré dans les exemples de code, plus haut dans cette section. Cette instruction est requise même lorsque le fichier SWF appelé et celui effectuant l'appel se trouvent sur le même domaine.
- Si le fichier appelé est publié pour Flash Player 6 ou une version antérieure, et que le fichier effectuant l'appel et le fichier appelé résident dans le même domaine (par exemple, si un fichier sur http://www.domaine.com appelle un fichier situé sur https://www.domaine.com), aucune modification n'est nécessaire.
- Si le fichier appelé est publié pour Flash Player 6, que les fichiers ne se trouvent pas sur le même domaine, et que vous ne souhaitez pas porter le fichier appelé sur Flash Player 7, modifiez le fichier appelé en ajoutant ou en modifiant une instruction `System.security.allowDomain` ou `LocalConnection.allowDomain` en utilisant la correspondance exacte de domaine, comme illustré dans les exemples de code plus haut dans cette section.
- Si le fichier appelé est publié pour Flash Player 6 et que vous souhaitez le porter sur Flash Player 7, incluez `System.security.allowInsecureDomain` ou `LocalConnection.allowInsecureDomain` dans le fichier appelé en utilisant la correspondance exacte de domaine, comme illustré dans les exemples de code, plus haut dans cette section. Cette instruction est requise même si les deux fichiers se trouvent dans le même domaine.

- Si le fichier appelé est publié pour Flash Player 5 ou une version antérieure et que les deux fichiers ne se trouvent pas sur le même domaine, vous pouvez procéder de l'une des deux manières suivantes. Vous pouvez soit porter le fichier appelé sur Flash Player 6 et ajouter ou modifier une instruction `System.security.allowDomain` en utilisant la correspondance exacte de domaine, soit porter le fichier appelé sur Flash Player 7, et inclure une instruction `System.security.allowInsecureDomain` dans le fichier appelé en utilisant le filtrage de domaine, comme illustré dans les exemples de code, plus haut dans cette section.

Fichiers de régulation côté serveur pour autoriser l'accès aux données

Un document Flash peut charger les données depuis une source externe à l'aide de l'un des appels de chargement de données suivants : `XML.load()`, `XML.sendAndLoad()`, `LoadVars.load()`, `LoadVars.sendAndLoad()`, `loadVariables()`, `loadVariablesNum()`, `MovieClip.loadVariables()`, `XMLSocket.connect()` et Macromedia Flash Remoting (`NetServices.createGatewayConnection`). De plus, un fichier SWF peut importer des bibliothèques partagées à l'exécution (RSL) ou des actifs définis dans un autre fichier SWF, au moment de l'exécution. Par défaut, les données ou le support RSL doivent se trouver sur le même domaine que le fichier SWF qui charge ces données externes ou ce support.

Pour que les fichiers SWF situés dans différents domaines puissent accéder aux données et aux actifs contenus dans des bibliothèques partagées à l'exécution, utilisez un *fichier de régulation inter-domaines*. Il s'agit d'un fichier XML qui permet au serveur d'indiquer que ses données et ses documents sont disponibles pour les fichiers SWF servis par certains domaines ou par tous les domaines. Tout fichier SWF servi par un domaine spécifié par le fichier de régulation du serveur peut accéder aux données et aux RSL de ce serveur.

Si vous chargez des données externes, il est conseillé de créer des fichiers de régulation même si vous ne projetez pas de porter vos fichiers sur Flash Player 7. Si vous utilisez des bibliothèques partagées à l'exécution (RSL), il est conseillé de créer des fichiers de régulation si le fichier appelé ou le fichier effectuant l'appel est publié pour Flash Player 7.

Pour plus d'informations, consultez *A propos de l'autorisation de chargement de données inter-domaines*, page 201.

Modifications de l'éditeur ActionScript

L'éditeur ActionScript a subi différentes mises à jour afin d'être plus robuste et plus simple à utiliser. Cette section récapitule toutes ces modifications.

Retour automatique à la ligne Pour activer ou désactiver le retour automatique à la ligne, vous pouvez désormais utiliser le menu d'options contextuel de la fenêtre de script, du panneau Débogueur et du panneau de sortie. Vous pouvez également l'activer ou le désactiver à l'aide du menu déroulant du panneau Actions. Le raccourci clavier est Ctrl+Maj+W (Windows) ou Commande+Maj+W (Macintosh).

Affichage de l'aide contextuelle Lorsque vous positionnez votre pointeur sur un élément du langage ActionScript dans la boîte à outils ou la fenêtre de script du panneau Actions, vous pouvez utiliser l'option Afficher l'aide du menu contextuel pour afficher une page d'aide concernant cet élément.

Importation de scripts Lorsque vous sélectionnez Importer le script dans le menu déroulant du panneau Actions, le script importé est copié dans le script à l'emplacement du point d'insertion dans votre fichier code. Dans les versions antérieures de Flash, l'importation d'un script écrasait le contenu du script existant.

Points d'arrêt en un clic Pour ajouter un point d'arrêt de débogage devant une ligne de code dans les panneaux Débogueur, Script ou Actions, vous pouvez cliquer dans la marge de gauche. Dans les versions antérieures de Flash, le fait de cliquer dans la marge de gauche sélectionnait une ligne de code. La nouvelle façon de sélectionner une ligne de code consiste à cliquer en appuyant sur Ctrl (Windows) ou à cliquer en appuyant sur Commande (Macintosh).

Modes normal et expert ne figurant plus dans le panneau Actions Dans les versions antérieures de Flash, vous pouviez travailler dans le panneau Actions soit en mode normal, dans lequel vous complétiez des options et des paramètres pour créer du code, soit en mode expert, dans lequel vous ajoutiez des commandes directement dans la fenêtre de script. Dans Flash MX 2004 et Flash MX Professionnel 2004, vous pouvez uniquement travailler dans le panneau Actions en ajoutant des commandes directement dans la fenêtre de script. Vous pouvez toujours faire glisser des commandes de la boîte à outils vers la fenêtre de script ou utiliser le bouton Ajouter (+) situé au-dessus de la fenêtre de script pour ajouter des commandes à un script.

Verrouillage de plusieurs scripts Vous pouvez verrouiller plusieurs scripts figurant dans un fichier FLA le long du bord inférieur de la fenêtre de script dans le panneau Actions. Dans les versions précédentes de Flash, vous ne pouviez verrouiller qu'un seul script à la fois.

Navigateur de script La partie gauche du panneau Actions contient désormais deux panneaux : la boîte à outils Actions et un nouveau navigateur de script. Le navigateur de script est une représentation visuelle de la structure de votre fichier FLA ; il vous permet de parcourir votre fichier FLA pour localiser le code ActionScript.

Fenêtre de script intégrée pour la modification de fichiers externes (Flash Professionnel uniquement) Vous pouvez utiliser l'éditeur ActionScript dans une fenêtre de script (distincte du panneau Actions) pour écrire et modifier des fichiers scripts externes. La coloration de la syntaxe, les conseils de code et d'autres préférences sont supportés dans la fenêtre de script. Une boîte à outils est également disponible. Pour afficher la fenêtre de script, cliquez sur Fichier > Nouveau, puis sélectionnez le type de fichier externe à modifier. Vous pouvez ouvrir plusieurs fichiers externes simultanément ; les noms de fichier s'affichent alors sur des onglets en haut de la fenêtre de script. Cette fonction est uniquement disponible sous Windows.

Modifications apportées dans le domaine du débogage

Cette section décrit les modifications permettant d'améliorer votre capacité à déboguer vos scripts.

Fenêtre Sortie changée en panneau de sortie Vous pouvez désormais déplacer et ancrer le panneau de sortie de la même manière que n'importe quel autre panneau dans Flash.

Amélioration de la signalisation des erreurs au moment de la compilation ActionScript 2.0 fournit non seulement une gestion des exceptions plus robuste, mais il propose également plusieurs nouvelles erreurs de compilation. Pour plus d'informations, consultez l'Annexe A, *Messages d'erreur*, page 893.

Amélioration de la gestion des exeptions La classe Error et les commandes `throw` et `try..catch..finally` vous permettent d'implémenter une gestion des exceptions plus robuste.

Nouveau modèle de programmation orientée objet

Depuis son introduction il y a quelques années de cela, le langage ActionScript n'a cessé de se développer. A chaque nouvelle version de Flash, de nouveaux mots-clés, objets, méthodes et autres éléments de langage sont ajoutés. Cependant, contrairement aux versions antérieures de Flash, Flash MX 2004 et Flash MX Professionnel 2004 présentent plusieurs nouveaux éléments de langage qui renforcent sa standardisation selon le modèle de programmation orientée objet. Ces éléments de langage constituent une amélioration significative du langage ActionScript de base et représentent donc une nouvelle version d'ActionScript lui-même : ActionScript 2.0.

ActionScript 2.0 n'est pas un nouveau langage. Il s'agit plutôt d'un ensemble d'éléments de langage de base qui simplifient le développement de programmes orientés objet. L'introduction de mots-clés tels que `class`, `interface`, `extends` et `implements` facilite l'apprentissage de la syntaxe ActionScript aux programmeurs habitués à d'autres langages. Les nouveaux programmeurs apprennent ainsi une terminologie standard applicable à d'autres langages orientés objet qu'ils étudieront sans doute ultérieurement.

ActionScript 2.0 supporte tous les éléments standard du langage ActionScript. Il vous permet simplement d'écrire des scripts plus conformes aux normes utilisées dans d'autres langages orientés objet, tels que Java. ActionScript 2.0 intéressera principalement les développeurs Flash de niveau intermédiaire ou avancé qui créent des applications nécessitant l'implémentation de classes et de sous-classes. ActionScript 2.0 vous permet également de déclarer le type d'objet d'une variable au moment de sa création (consultez *Typage strict des données*, page 40) et améliore considérablement les erreurs de compilateur (consultez l'Annexe A, *Messages d'erreur*, page 893).

Les nouveaux éléments d'ActionScript 2.0 sont répertoriés ci-dessous :

- `class`
- `extends`
- `implements`
- `interface`
- `dynamic`
- `static`
- `public`
- `private`
- `get`
- `set`
- `import`

Les principaux points à connaître sur ActionScript 2,0 sont les suivants :

- Les scripts utilisant ActionScript 2.0 pour définir des interfaces doivent être enregistrés en tant que fichiers scripts externes, avec une seule classe définie dans chaque script. Cela implique que les classes et les interfaces ne peuvent pas être définies dans le panneau Actions.
- Vous pouvez importer des fichiers de classe individuels de façon implicite (en les enregistrant dans un emplacement spécifié par des chemins de recherche généraux ou spécifiques à des documents et en les utilisant dans un script) ou de façon explicite (en utilisant la commande `import`). Vous pouvez importer des ensembles de fichiers (ensembles de fichiers de classe dans un répertoire) en utilisant des caractères génériques.

- Les applications développées à l'aide d'ActionScript 2.0 sont supportées par Flash Player 6 et les versions ultérieures.

Attention : Le paramètre de publication par défaut des nouveaux fichiers créés dans Flash MX 2004 est ActionScript 2.0 Si vous projetez de modifier un fichier FLA existant pour utiliser la syntaxe ActionScript 2.0, vérifiez que le fichier FLA spécifie bien ActionScript 2.0 dans ses paramètres de publication. Si ce n'est pas le cas, votre fichier ne sera pas compilé correctement, même si Flash ne génère pas d'erreurs de compilateur.

Pour plus d'informations sur l'utilisation d'ActionScript 2.0 pour l'écriture de programmes orientés objet en Flash, consultez le Chapitre 9, *Création de classes avec ActionScript 2.0*, page 163.

CHAPITRE 2
Notions de base du langage ActionScript

ActionScript possède des règles de grammaire et de ponctuation qui déterminent les caractères ou les mots porteurs de sens et l'ordre dans lequel ils peuvent être rédigés. Par exemple, en français, un point termine une phrase. Dans ActionScript, c'est un point-virgule qui termine une instruction.

Les règles générales suivantes s'appliquent à tous les scripts ActionScript. La plupart des termes ActionScript font également l'objet de règles individuelles ; pour en savoir plus sur les règles qui s'appliquent à un terme déterminé, consultez l'entrée correspondante dans le Chapitre 12, *Dictionnaire ActionScript*, page 215.

Différences entre ActionScript et JavaScript

ActionScript est similaire au langage de programmation JavaScript. La connaissance de JavaScript n'est pas indispensable pour apprendre à utiliser ActionScript. Toutefois, si vous connaissez JavaScript, ActionScript vous semblera familier.

Cet ouvrage n'a pas pour but d'enseigner la programmation générale. Il existe de nombreuses sources qui fournissent des informations complémentaires sur les concepts de programmation généraux et sur le langage JavaScript.

- L'ECMA (European Computers Manufacturers Association) a établi la spécification ECMA-262, dérivée du langage JavaScript, qui sert de norme internationale pour le langage JavaScript. ActionScript est basé sur la spécification ECMA-262.
- Netscape DevEdge Online offre un centre de développement JavaScript (http://developer.netscape.com/tech/javascript/index.html) qui contient de la documentation et des articles utiles à la compréhension d'ActionScript. La ressource la plus importante est le *Core JavaScript Guide.*

Les principales différences qui existent entre ActionScript et JavaScript sont les suivantes :

- ActionScript ne supporte pas les objets spécifiques aux navigateurs que sont les documents, fenêtres et ancres, par exemple.
- ActionScript ne supporte pas entièrement tous les objets JavaScript intégrés.
- ActionScript ne supporte pas certaines constructions syntaxiques JavaScript, telles que les étiquettes d'instructions.
- Dans ActionScript, l'action `eval()` ne peut effectuer que des références aux variables.

Support du format Unicode pour ActionScript

Macromedia Flash MX 2004 et Macromedia Flash MX Professionnel 2004 prennent en charge le codage du texte au format Unicode pour ActionScript. Vous pouvez donc intégrer du texte en différentes langues dans un même fichier ActionScript. Vous pouvez par exemple intégrer du texte en anglais, en japonais et en français dans un même fichier.

Vous pouvez définir les préférences d'ActionScript afin de spécifier le type de codage à utiliser lors de l'importation ou de l'exportation de fichiers ActionScript. Vous avez le choix entre le codage UTF-8 ou le codage par défaut. UTF-8 est le format Unicode 8 bits. Le codage par défaut, ou *page de code classique,* est le codage pris en charge par la langue utilisée actuellement sur votre système.

En règle générale, si vous importez ou exportez des fichiers ActionScript au format UTF-8, choisissez UTF-8. Si vous importez ou exportez des fichiers dans la page de code classique de votre système, choisissez Codage par défaut comme préférence.

Si le texte que contiennent vos scripts ne s'affiche pas correctement lorsque vous ouvrez ou importez un fichier, modifiez la préférence de codage pour l'importation. Si vous obtenez un avertissement lors de l'exportation de fichiers ActionScript, vous pouvez modifier la préférence de codage pour l'exportation ou désactiver l'affichage de cet avertissement dans les préférences ActionScript.

Pour choisir des options de codage de texte pour l'importation ou l'exportation de fichiers ActionScript :

1 Dans la boîte de dialogue Préférences (Modifier > Préférences), cliquez sur l'onglet ActionScript.

2 Dans la section Options d'édition, effectuez au moins une des actions suivantes :

- Pour l'option Ouvrir/Importer, choisissez UTF-8 pour ouvrir ou importer en utilisant le codage Unicode, ou choisissez Codage par défaut pour ouvrir ou importer en utilisant le format de codage de la langue de votre système actuel.
- Pour l'option Enregistrer/Exporter, choisissez UTF-8 pour enregistrer ou exporter en utilisant le codage Unicode, ou choisissez Codage par défaut pour enregistrer ou exporter en utilisant le format de codage de la langue de votre système actuel.

Pour activer ou désactiver le message d'avertissement de codage d'exportation :

1 Dans la boîte de dialogue Préférences (Edition > Préférences), cliquez sur l'onglet Avertissements.

2 Activez ou désactivez l'option Avertir des conflits de codage pendant l'exportation de fichiers .as.

Attention : La commande Tester l'animation (consultez *Débogage de scripts*, page 73) échoue si une partie du chemin du fichier SWF contient des caractères ne pouvant pas être représentés à l'aide du système de codage MBCS. Par exemple, les chemins japonais dans un système anglais ne fonctionnent pas. Toutes les zones de l'application utilisant le lecteur externe sont soumises à cette restriction.

Terminologie

A l'instar de tout langage de programmation, ActionScript utilise une terminologie qui lui est propre. La liste suivante présente les principaux termes ActionScript.

Les **actions** sont des instructions qui ordonnent à un fichier SWF d'exécuter une opération déterminée lors de sa lecture. Par exemple, `gotoAndStop()` envoie la tête de lecture à une image ou une étiquette spécifique. Dans cet ouvrage, les termes *action* et *instruction* sont interchangeables.

Une **valeur booléenne** est une valeur `true` (vraie) ou `false` (fausse).

Les **classes** sont des types de données que vous pouvez créer pour définir un nouveau type d'objet. Pour définir une classe, utilisez le mot-clé `class` dans un fichier de script externe (et non dans le script en cours de rédaction dans le panneau Actions).

Les **constantes** sont des éléments qui ne changent pas. Par exemple, la constante `Key.TAB` a toujours la même signification : elle indique la touche Tab du clavier. Les constantes sont utiles pour comparer des valeurs.

Les **constructeurs** sont des fonctions que vous utilisez pour définir les propriétés et les méthodes d'une classe. Par définition, les constructeurs sont des fonctions au sein d'une définition de classe qui portent le même nom que la classe. Par exemple, le code suivant définit une classe « Cercle » et implémente une fonction constructeur :

```
// fichier Cercle.as
class Cercle {
  private var radius:Number
  private var circumference:Number
// constructeur
  function Cercle(radius:Number) {
    circumference = 2 * Math.PI * radius;
  }
}
```

Le terme *constructeur* est également utilisé lorsque vous créez (instanciez) un objet en fonction d'une classe particulière. Les instructions suivantes sont des constructeurs pour la classe « Array » intégrée et pour la classe « Cercle » personnalisée :

```
mon_array:Array = new Array();
mon_cercle:Cercle = new Cercle();
```

Les **types de données** décrivent le genre d'informations qu'une variable ou qu'un élément ActionScript peut contenir. Les types de données ActionScript sont les suivants : String, Number, Boolean, Object, MovieClip, Function, null et undefined. Pour plus d'informations, consultez *A propos des types de données*, page 36.

Les **événements** sont des actions qui se produisent lors de la lecture d'un fichier SWF. Par exemple, différents événements sont générés lorsqu'un clip est chargé, que la tête de lecture entre dans une image, que l'utilisateur clique sur un bouton ou clip ou qu'il tape sur le clavier.

Les **gestionnaires d'événement** sont des actions spéciales qui gèrent les événements tels que `mouseDown` ou `load`. Il existe deux sortes de gestionnaires d'événement ActionScript : les méthodes de gestionnaire d'événement et les écouteurs d'événement. (Il existe également deux gestionnaires d'événement, `on()` et `onClipEvent()`, que vous pouvez affecter directement aux boutons et aux clips.) Dans la boîte à outils du panneau Actions, chaque objet ActionScript qui possède des méthodes de gestionnaire d'événement ou des écouteurs d'événement est associé à une sous-catégorie appelée Evénements ou Ecouteurs. Certaines commandes, qui peuvent être utilisées à la fois en tant que gestionnaires d'événement et écouteurs d'événement, sont incluses dans les deux sous-catégories.

Une **expression** est toute combinaison légale de symboles ActionScript qui représentent une valeur. Une expression se compose d'opérateurs et d'opérandes. Par exemple, dans l'expression `x + 2`, `x` et `2` sont des opérandes et `+` est un opérateur.

Les **fonctions** sont des blocs de code réutilisables qui peuvent recevoir des paramètres et renvoyer une valeur. Pour plus d'informations, consultez *Création de fonctions*, page 54.

Les **identifiants** sont des noms utilisés pour indiquer une variable, une propriété, un objet, une fonction ou une méthode. Le premier caractère doit être une lettre, un trait de soulignement (_) ou un dollar ($). Chaque caractère qui suit doit être une lettre, un chiffre, un trait de soulignement ou un dollar. Par exemple, `prénom` est le nom d'une variable.

Les **occurrences** sont des objets qui appartiennent à une certaine classe. Chaque occurrence d'une classe contient toutes les propriétés et les méthodes de cette classe. Par exemple, tous les clips sont des occurrences de la classe MovieClip, vous pouvez alors utiliser n'importe quelle méthode ou propriété de la classe MovieClip avec n'importe quelle occurrence de clip.

Les **noms d'occurrence** sont des noms uniques qui vous permettent de cibler des occurrences de clip et de bouton dans les scripts. L'inspecteur des propriétés permet d'affecter des noms aux occurrences présentes sur la scène. Par exemple, un symbole principal de la bibliothèque pourrait s'appeler `compteur` et les deux occurrences de ce symbole dans le fichier SWF pourraient avoir comme noms d'occurrence `scoreJoueur1_mc` et `scoreJoueur2_mc`. Le code suivant définit une variable appelée `score` à l'intérieur de chaque occurrence de clip en utilisant le nom des occurrences :

```
_root.scorePlayer1_mc.score += 1;
_root.scorePlayer2_mc.score -= 1;
```

Vous pouvez utiliser des suffixes spéciaux lors de l'affectation d'un nom à une occurrence, afin que des conseils de code (consultez *Utilisation des conseils de code*, page 68) s'affichent lorsque vous tapez du code. Pour plus d'informations, consultez *Utilisation de suffixes pour déclencher des conseils de code*, page 66.

Les **mots-clés** sont des mots réservés avec une signification particulière. Par exemple, `var` est un mot-clé utilisé pour déclarer des variables locales. Vous ne pouvez pas utiliser un mot-clé comme identificateur. Par exemple, `var` n'est pas un nom de variable légal. Pour obtenir une liste des mots-clés, consultez *Mots-clés*, page 36.

Les **méthodes** sont des fonctions associées à une classe. Par exemple, `getBytesLoaded()` est une méthode intégrée associée à la classe MovieClip. Vous pouvez aussi créer des fonctions qui agissent ensuite comme des méthodes pour les objets basés sur les classes intégrées ou sur les classes que vous créez. Par exemple, dans le code suivant, `clear()` devient une méthode d'un objet `contrôleur` que vous avez précédemment défini :

```
function reset(){
  this.x_pos = 0;
  this.x_pos = 0;
}
contrôleur.clear = reset;
contrôleur.clear();
```

Les **objets** sont des groupes de propriétés et méthodes, chaque objet ayant son propre nom et étant une occurrence d'une classe particulière. Les objets intégrés sont prédéfinis dans le langage ActionScript. Par exemple, l'objet intégré Date fournit des informations provenant de l'horloge système.

Les **opérateurs** sont des termes qui calculent une nouvelle valeur à partir d'une ou de plusieurs valeurs. Par exemple, l'opérateur d'addition (+) additionne deux ou plusieurs valeurs pour en obtenir une nouvelle. Les valeurs manipulées par les opérateurs sont appelées *opérandes*.

Les **paramètres**, également appelés *arguments*, sont des espaces réservés qui vous permettent de transmettre des valeurs aux fonctions. Par exemple, la fonction `bienvenue()` suivante, utilise deux valeurs qu'elle reçoit dans les paramètres `prénom` et `hobby` :

```
function bienvenue(prénom, hobby) {
  texteDeBienvenue = "Bonjour " + prénom + "Votre hobby est : " + hobby;
}
```

Les **paquets** sont des répertoires qui contiennent un ou plusieurs fichiers de classe et résident dans un répertoire de chemin de classe désigné (consultez *Compréhension du chemin de classe*, page 177).

Les **propriétés** sont des attributs qui définissent un objet. Par exemple, `_visible` est une propriété de tous les clips qui définit si ceux-ci sont visibles ou masqués.

Les **chemins cibles** sont des adresses hiérarchiques de noms d'occurrences de clips, de variables et d'objets d'un fichier SWF. Les occurrences de clips sont nommées dans l'inspecteur des propriétés des clips correspondants. Le scénario principal porte toujours le nom `_root`. Vous pouvez utiliser un chemin cible pour réaliser une action sur un clip ou pour obtenir ou définir la valeur d'une variable. Par exemple, l'instruction suivante est le chemin cible de la variable `volume` dans le clip `contrôleStéréo` :

```
_root.contrôleStéréo.volume
```

Pour plus d'informations sur les chemins cibles, consultez Chemins cibles absolus et relatifs dans le guide Utilisation de Flash de l'aide.

Les **variables** sont des identifiants qui contiennent des valeurs de n'importe quel type de données. Les variables peuvent être créées, modifiées et mise à jour. Vous pouvez récupérer les valeurs qu'elles contiennent et les utiliser dans des scripts. Dans l'exemple suivant, les identifiants situés à gauche du signe égal sont des variables :

```
var x = 5;
var name = "Lolo";
var c_color = new Color(mcinstanceName);
```

Pour plus d'informations sur les variables, consultez *A propos des variables*, page 43.

Syntaxe

Comme dans tout langage, ActionScript implique des règles syntaxiques à respecter pour créer des scripts pouvant être compilés et exécutés correctement. Cette section décrit les éléments de syntaxe ActionScript.

Hauteur de casse

Dans un langage de programmation sensible à la casse, les noms de variable qui diffèrent uniquement par leur casse (`livre` et `Livre`) ne sont pas considérés comme identiques. Par conséquent, il est toujours judicieux d'employer les majuscules et les minuscules selon des conventions fixes, comme celles utilisées dans cet ouvrage, car cela permet d'identifier plus facilement les noms des fonctions et des variables dans le code ActionScript.

Lorsque vous publiez des fichiers pour Flash Player 7 ou une version ultérieure, Flash implémente la sensibilité à la casse que vous utilisiez ActionScript 1 ou ActionScript 2.0. Cela signifie que les mots-clés, noms de classe, variables, noms de méthode etc. sont tous sensibles à la casse. Par exemple :

```
// Dans un fichier ciblant Flash Player 7
// et dans ActionScript 1 ou ActionScript 2.0
//
// Définit les propriétés de deux objets différents
chat.hilite = true;
CHAT.hilite = true;

// Crée trois variables différentes
var maVar=10;
var mavar=10;
var mAvAr=10;
// Ne génère pas d'erreur
var tableau = new Array();
var date = new Date();
```

Cette modification a également des répercussions sur les variables externes chargées via `LoadVars.load()`.

En outre, la sensibilité à la casse est implémentée pour les scripts externes, tels que les scripts ou les fichiers de classe ActionScript 2.0 que vous importez en utilisant la commande `#include`. Si vous publiez des fichiers pour Flash Player 7 et avez préalablement créé des fichiers externes que vous ajoutez à vos scripts via l'instruction `#include`, vous devez passer chacun d'eux en revue pour vous assurer que la casse est correcte. Pour ce faire, ouvrez le fichier dans la fenêtre de script (Flash Professionnel uniquement) ou, dans un nouveau fichier FLA, réglez vos paramètres de publication sur Flash Player 7 et copiez le contenu du fichier dans le panneau Actions. Utilisez ensuite le bouton Vérifier la syntaxe (consultez *Vérification de la syntaxe et de la ponctuation*, page 71) ou publiez votre fichier. Les erreurs dues à des conflits de noms sont signalées dans le panneau de sortie.

Lorsque la coloration de la syntaxe est activée, les éléments du langage dont la casse est correcte sont affichés en bleu par défaut. Pour plus d'informations, consultez *Mots-clés*, page 36, et *Mise en évidence de la syntaxe*, page 65.

Syntaxe pointée

Dans ActionScript, un point (.) est utilisé pour indiquer les propriétés ou les méthodes associées à un objet ou à un clip. Il est également utilisé pour identifier le chemin cible d'un clip, d'une variable, d'une fonction ou d'un objet. Une expression en syntaxe à point commence par le nom de l'objet ou du clip suivi d'un point et se termine par l'élément que vous souhaitez spécifier.

Par exemple, la propriété _x d'un clip indique la position sur l'axe *x* du clip sur la scène. L'expression `balleMC._x` fait référence à la propriété _x de l'occurrence de clip `balleMC`.

Dans un autre exemple, `envoyer` est une variable définie dans le clip `formulaire`, qui est imbriqué dans le clip `panier`. L'expression `panier.formulaire.envoyer = true` définit la variable `envoyer` du `formulaire` de l'occurrence sur `true`.

L'expression d'une méthode d'un objet ou clip se fait selon le même schéma. Par exemple, la méthode `play()` de l'occurrence de clip `balle_mc` déplace la tête de lecture dans le scénario de `balle_mc`, comme indiqué dans l'instruction suivante :

```
balle_mc.play();
```

La syntaxe pointée utilise également deux alias spéciaux, `_root` et `_parent`. L'alias `_root` fait référence au scénario principal. Vous pouvez utiliser l'alias `_root` pour créer un chemin cible absolu. Par exemple, l'instruction suivante appelle la fonction `constructPlateau()` dans le clip `fonctions` du scénario principal :

```
_root.fonctions.constructPlateau();
```

Vous pouvez utiliser l'alias `_parent` pour faire référence à un clip dans lequel est imbriqué l'objet courant. Vous pouvez également utiliser `_parent` pour créer un chemin cible relatif. Par exemple, si le clip `chien_mc` est imbriqué dans le clip `animal_mc`, l'instruction suivante de l'occurrence `chien_mc` indique à `animal_mc` de s'arrêter :

```
_parent.stop();
```

Syntaxe à barre oblique

La syntaxe à barre oblique était utilisée dans Flash 3 et 4 pour indiquer le chemin cible d'un clip ou d'une variable. Cette syntaxe est toujours supportée dans Flash Player 7, mais son utilisation n'est pas recommandée. La syntaxe à barre oblique n'est pas supportée dans ActionScript 2.0. Toutefois, si vous créez du contenu destiné spécialement à Flash Player 4, vous devrez utiliser la syntaxe à barre oblique. Pour plus d'informations, consultez *Utilisation de la syntaxe à barre oblique*, page 909.

Accolades

Les gestionnaires d'événement, les définitions de classe et les fonctions ActionScript sont regroupés en blocs délimités par des accolades ({}). Vous pouvez placer l'accolade ouvrante sur la même ligne que la déclaration ou sur la ligne suivante, comme illustré dans les exemples ci-dessous. Pour améliorer la lisibilité du code, il est préférable de choisir un format et de s'y tenir.

```
//Gestionnaire d'événement
on(release) {
  maDate = new Date();
  moisCourant = maDate.getMonth();
}

on(release)
```

```
{
  maDate = new Date();
  moisCourant = maDate.getMonth();
}

// Classe
class Cercle(radius) {
}

class Square(side)
{
}

//Fonction
circleArea = function(radius) {
  return radius * radius * MATH.PI;
}
squareArea = function(side)
{
  return side * side;
}
```

Vous pouvez contrôler qu'il ne manque pas d'accolades dans vos scripts ; consultez *Vérification de la syntaxe et de la ponctuation*, page 71.

Points-virgules

Une instruction ActionScript se termine par un point-virgule (;), comme dans les exemples suivants :

```
var colonne = dateTransmise.getDay();
var ligne    = 0;
```

L'omission du point-virgule final n'empêche pas Flash de compiler le script correctement. Cependant, l'emploi de points-virgules est une bonne habitude à prendre lors de la rédaction de scripts.

Parenthèses

Lorsque vous définissez une fonction, placez les paramètres entre parenthèses :

```
function maFonction (nom, âge, lecteur){
  // entrez votre code ici
}
```

Lorsque vous appelez une fonction, incluez tous les paramètres transmis à la fonction entre parenthèses, comme suit :

```
maFonction ("Steve", 10, true);
```

Vous pouvez également utiliser les parenthèses pour supplanter l'ordre de priorité d'ActionScript ou pour faciliter la lecture des instructions ActionScript. Pour plus d'informations, consultez *Priorité et associativité des opérateurs*, page 48.

Les parenthèses servent également à évaluer une expression située à gauche d'un point dans la syntaxe pointée. Par exemple, dans l'instruction suivante, les parenthèses obligent à l'évaluation de `new Color(this)` et à la création d'un objet Color :

```
onClipEvent(enterFrame){
  (new Color(this)).setRGB(0xffffff);
}
```

Si vous n'utilisez pas de parenthèses, vous devez ajouter une instruction pour évaluer l'expression :

```
onClipEvent(enterFrame){
  maCouleur = new Color(this);
  maCouleur.setRGB(0xffffff);
}
```

Vous pouvez contrôler qu'il ne manque pas de parenthèses dans vos scripts ; consultez *Vérification de la syntaxe et de la ponctuation*, page 71.

Commentaires

Il est vivement recommandé d'utiliser des commentaires pour ajouter des notes aux scripts. Les commentaires sont particulièrement utiles pour consigner vos intentions et transmettre des informations à d'autres développeurs (si vous travaillez en équipe ou si vous fournissez des échantillons). Même un script simple est plus facile à comprendre si vous l'annotez lors de sa création.

Pour indiquer qu'une ligne ou portion de ligne est un commentaire, faites-la précéder de deux barres obliques(//) :

```
on(release) {
  // créer un nouvel objet Date
  maDate = new Date();
  moisCourant = maDate.getMonth();
  // convertir le chiffre du mois en son nom
  nomDuMois = calcMois(moisCourant);
  année = maDate.getFullYear();
  dateDuJour = maDate.getDate();
}
```

Lorsque la coloration de la syntaxe est activée (consultez *Mise en évidence de la syntaxe*, page 65), les commentaires apparaissent en gris, par défaut. Les commentaires peuvent avoir n'importe quelle longueur sans que cela affecte la taille du fichier exporté. Ils ne suivent aucune règle de syntaxe ou de mots-clés relative à ActionScript.

Si vous souhaitez appliquer un commentaire à une portion complète de script, intégrez-le dans un bloc de commentaire plutôt que de devoir ajouter // au début de chaque ligne. Cette technique est plus simple et plus pratique, surtout lorsque vous souhaitez tester uniquement certaines parties d'un script en en commentant de grandes parties.

Pour créer un bloc de commentaire, entrez /* au début du commentaire, puis */ à la fin. Par exemple, lorsque le script suivant est exécuté, le code intégré dans le bloc de commentaire n'est pas exécuté :

```
// Le code ci-dessous est exécuté
var x:Number = 15;
var y:Number = 20;
// Le code ci-dessous ne s'exécute pas
/*
on(release) {
  // créer un nouvel objet Date
  maDate = new Date();
  moisCourant = maDate.getMonth();
  // convertir le chiffre du mois en son nom
  nomDuMois = calcMois(moisCourant);
  année = maDate.getFullYear();
  dateDuJour = maDate.getDate();
}
```

```
*/
// Le code ci-dessous est exécuté
var nom:String = "Je m'appelle";
var âge:Number = 20;
```

Mots-clés

ActionScript réserve certains mots à des usages spécifiques au sein du langage et vous ne pouvez pas vous en servir comme identifiants (noms de variable, de fonction, d'étiquette, etc.). Le tableau suivant répertorie tous les mots-clés ActionScript :

break	case	classe	continue
default	delete	dynamic	else
extends	for	function	get
if	implements	import	in
instanceof	interface	intrinsic	new
private	public	return	set
static	switch	this	typeof
var	void	while	with

Constantes

Une constante est une propriété dont la valeur ne varie jamais.

Par exemple, les constantes `BACKSPACE`, `ENTER`, `QUOTE`, `RETURN`, `SPACE` et `TAB` sont des propriétés de l'objet Key et font référence aux touches du clavier. Pour savoir si un utilisateur appuie sur la touche Entrée, vous pourriez utiliser l'instruction suivante :

```
if(Key.getCode() == Key.ENTER) {
  alert = "Etes-vous prêt(e) ?";
  controlMC.gotoAndStop(5);
}
```

A propos des types de données

Un type de données décrit le genre d'informations qu'une variable ou qu'un élément ActionScript peut contenir. Il existe deux sortes de types de données intégrées dans Flash : primitives et de référence. Les données primitives (String, Number et Boolean) ont une valeur constante et peuvent donc contenir la valeur réelle de l'élément qu'elles représentent. Les données de référence (MovieClip et Object) possèdent des valeurs qui peuvent changer et contiennent donc des références à la valeur réelle de l'élément. Les variables contenant des données primitives ont un comportement différent de celles contenant des références dans certaines situations. Pour plus d'informations, consultez *Utilisation des variables dans un programme*, page 46. Il existe également deux types spéciaux de données : null et undefined.

Dans Flash, tout objet intégré qui n'est pas une donnée primitive, ni une donnée de clip, telle que Array ou Math, est une donnée d'objet.

Chaque type de données possède ses propres règles et est décrit sous les rubriques suivantes :

Lorsque vous déboguez des scripts, vous pouvez avoir besoin de déterminer le type de données d'une expression ou d'une variable pour comprendre pourquoi elle se comporte de telle manière. Vous pouvez effectuer cette opération avec l'opérateur `typeof` (consultez *Définition du type de données d'un objet*, page 39).

Vous pouvez convertir un type de données en un type différent en utilisant l'une des fonctions de conversion suivantes : `Array()`, `Boolean()`, `Number`, `Objet()`, `String`.

String

Une chaîne est une séquence de caractères (lettres, chiffres et signes de ponctuation, par exemple). Vous insérez des chaînes dans une instruction ActionScript en les plaçant entre des guillemets droits simples ou doubles. Les chaînes sont traitées comme des caractères et non comme des variables. Par exemple, dans l'instruction suivante, `"L7"` est une chaîne :

```
groupePréféré = "L7";
```

Vous pouvez utiliser l'opérateur d'addition (+) pour *concaténer*, ou joindre, deux chaînes. ActionScript traite les espaces au début ou à la fin d'une chaîne comme faisant partie de la chaîne. L'expression suivante contient un espace après la virgule :

```
salutations = "Bonjour, " + prénom;
```

Pour inclure un guillemet dans une chaîne, il faut le faire précéder d'une barre oblique inverse (\). Cette opération s'appelle *échapper* un caractère. D'autres caractères ne peuvent pas être représentés dans ActionScript sans l'emploi de séquences d'échappement particulières. Le tableau suivant répertorie l'ensemble des caractères d'échappement d'ActionScript :

Séquence d'échappement	Caractère
`\b`	Caractère de retour arrière (ASCII 8)
`\f`	Caractère de changement de page (ASCII 12)
`\n`	Caractère de changement de ligne (ASCII 10)
`\r`	Caractère de retour chariot (ASCII 13)
`\t`	Caractère de tabulation (ASCII 9)
`\"`	Guillemet droit double
`\'`	Guillemet droit simple
`\\`	Barre oblique inverse

Séquence d'échappement	Caractère
\000 - \377	Un octet spécifié en octal
\x00 - \xFF	Un octet spécifié en hexadécimal
\u0000 - \uFFFF	Un caractère Unicode 16 bits spécifié en hexadécimal

Number

Le type de données Number correspond à un nombre à virgule flottante à double précision. Vous pouvez manipuler les nombres avec les opérateurs arithmétiques d'addition (+), de soustraction (-), de multiplication (*), de division (/), de modulo (%), d'incrémentation (++) et de décrémentation (--). Vous pouvez également utiliser des méthodes des classes intégrées Math et Number pour manipuler les nombres. L'exemple suivant utilise la méthode `sqrt()` (racine carrée) pour renvoyer la racine carrée de 100 :

```
Math.sqrt(100);
```

Pour plus d'informations, consultez *Opérateurs numériques*, page 48.

Boolean

Une valeur booléenne est soit `true` (vraie), soit `false` (fausse). ActionScript convertit également les valeurs `true` et `false` en 1 et 0 lorsque cela est nécessaire. Les valeurs booléennes sont le plus souvent utilisées dans les instructions ActionScript effectuant des comparaisons pour contrôler le déroulement d'un script. Par exemple, dans le script suivant, le fichier SWF est lu si la variable `motDePasse` est vraie (`true`) :

```
onClipEvent(enterFrame){
  if (nomDutilisateur == true && motDePasse == true){
    play();
  }
}
```

Consultez *Utilisation de fonctions intégrées*, page 54 et *Opérateurs logiques*, page 50.

Object

Un objet est une collection de propriétés. Chaque propriété possède un nom et une valeur. La valeur d'une propriété peut être de n'importe quel type de données Flash, même un type de données Object. Cela vous permet d'arranger les objets les uns dans les autres, ou de les *imbriquer*. Pour spécifier les objets et leurs propriétés, vous devez utiliser l'opérateur point (`.`). Par exemple, dans le code suivant, `heuresTravaillées` est une propriété de `statsHebdo`, qui est une propriété de `personnel` :

```
personnel.statsHebdo.heuresTravaillées
```

Vous pouvez utiliser les objets ActionScript intégrés pour localiser et manipuler certains types d'informations spécifiques. Par exemple, l'objet Math possède des méthodes qui effectuent des opérations mathématiques sur les nombres que vous leur transmettez. Cet exemple utilise la méthode `sqrt()` :

```
racineCarrée = Math.sqrt(100);
```

L'objet ActionScript MovieClip possède des méthodes qui vous permettent de contrôler les occurrences de symbole de clip sur la scène. Cet exemple utilise les méthodes `play()` et `nextFrame()` :

```
nomDoccurrenceMC.play();
nomDoccurrenceMC2.nextFrame();
```

Vous pouvez aussi créer des objets personnalisés pour organiser les informations dans votre application Flash. Pour ajouter de l'interactivité à votre application avec ActionScript, vous aurez besoin d'un certain nombre d'informations : un nom d'utilisateur, la vitesse d'une balle, les noms des objets contenus dans un panier, le nombre d'images chargées, le code postal de l'utilisateur et la dernière touche utilisée, par exemple. La création d'objets personnalisés vous permet d'organiser ces informations dans des groupes, de simplifier la rédaction et de réutiliser vos scripts.

MovieClip

Les clips sont des symboles qui peuvent lire des effets animés dans une application Flash. Ils sont le seul type de données faisant référence à un élément graphique. Le type de données MovieClip vous permet de contrôler les symboles de clip au moyen des méthodes de la classe MovieClip. Vous appelez les méthodes en utilisant l'opérateur point (`.`), comme ci-dessous :

```
mon_mc.startDrag(true);
parent_mc.getURL("http://www.macromedia.com/support/" + produit);
```

Null

Le type de données nul ne possède qu'une valeur, `null`. Cette valeur signifie en fait « pas de valeur », c'est-à-dire une absence de données. La valeur `null` peut être utilisée dans diverses situations. En voici quelques exemples :

- Pour indiquer qu'une variable n'a pas encore reçu de valeur.
- Pour indiquer qu'une variable ne contient plus de valeur.
- En tant que valeur de retour d'une fonction, afin d'indiquer qu'aucune valeur n'a pu être retournée par la fonction.
- En tant que paramètre d'une fonction, afin d'indiquer qu'un paramètre est omis.

Undefined

Le type de données undefined ne possède qu'une valeur, `undefined`, et est utilisé pour les variables auxquelles aucune valeur n'a été affectée.

Définition du type de données d'un objet

Lors du test et du débogage de vos programmes, des problèmes liés aux types de données peuvent se produire. Dans ce cas, il peut être souhaitable de déterminer le type de données d'un objet. Pour ce faire, utilisez l'opérateur `typeof`, comme dans l'exemple suivant :

```
trace(typeof(nomDeVariable));
```

Pour plus d'informations sur le test et le débogage, consultez le Chapitre 3, *Rédaction et débogage de scripts*, page 59.

Affectation de types de données aux éléments

Flash affecte automatiquement les types de données aux éléments de langage suivants, comme indiqué dans la section suivante, *Typage des données automatique* :

- Variables
- Paramètres transmis à une fonction, à une méthode ou à une classe
- Valeurs renvoyées d'une fonction ou d'une méthode
- Objets créés comme sous-classes de classes existantes

Vous pouvez également affecter explicitement des types de données aux éléments, ce qui peut vous aider à éviter ou diagnostiquer certaines erreurs dans vos scripts. Pour plus d'informations, consultez *Typage strict des données*, page 40.

Typage des données automatique

Dans Flash, il n'est pas nécessaire de définir explicitement un élément comme contenant un nombre, une chaîne ou un autre type de données. Flash détermine le type de données d'un élément lorsque celui est affecté :

```
var x = 3;
```

Dans l'expression `var x = 3`, Flash évalue l'élément à droite de l'opérateur et détermine qu'il s'agit du type de données nombre. Une affectation ultérieure pourra changer le type de `x`. Par exemple, l'instruction `x = "bonjour"` change le type de `x` en chaîne. Une variable à laquelle aucune valeur n'a été affectée est du type `undefined` (non défini).

ActionScript convertit automatiquement les types de données lorsqu'une expression le nécessite. Par exemple, lorsque vous transmettez une valeur à l'action `trace()`, `trace()` convertit automatiquement la valeur en chaîne et l'envoie au panneau de sortie. Dans les expressions avec opérateurs, ActionScript convertit les types de données en fonction des besoins, par exemple, lors de l'utilisation dans une chaîne, l'opérateur + s'attend à ce que l'autre opérande soit une chaîne.

```
"Suivant : le numéro " + 7
```

ActionScript convertit le chiffre 7 en chaîne `"7"` et l'ajoute à la fin de la première chaîne, ce qui aboutit à la chaîne suivante :

```
"Suivant : le numéro 7"
```

Typage strict des données

ActionScript 2.0 vous permet de déclarer explicitement le type d'objet d'une variable lorsque vous la créez, ce qui est appelé *typage strict*. Les incompatibilités de type de données déclenchent des erreurs de compilation, le typage strict vous permet donc d'éviter d'affecter un type de données incorrect à une variable existante. Pour affecter un type de données spécifique à un élément, spécifiez son type à l'aide d'une syntaxe utilisant le mot-clé `var` ainsi que deux points :

```
// typage strict de variable ou objet
var x:Number = 7;
var anniversaire:Date = new Date();

// typage strict de paramètres
function welcome(firstName:String, age:Number){
}
```

```
// typage strict de paramètre et de valeur renvoyée
function square(x:Number):Number {
  var squared = x*x;
  return squared;
}
```

Etant donné que vous devez utiliser le mot-clé `var` pour le typage strict des variables, vous ne pouvez pas définir strictement le type d'une variable globale (consultez *Domaine et déclaration de variables*, page 44).

Vous pouvez typer des objets en fonction des classes intégrées (Button, Date, MovieClip, etc.) et des classes et interfaces que vous créez. Par exemple, vous avez un fichier qui s'appelle Etudiant.as dans lequel vous avez défini une classe Etudiant, vous pouvez spécifier que les objets que vous créez sont de type Etudiant :

```
var étudiant:Student = new Student();
```

Vous pouvez également spécifier que les objets sont de type Function ou Void.

L'utilisation du typage strict vous empêche d'affecter, par inadvertance, un type de valeur incorrect à un objet. Flash recherche les incompatibilités de type au moment de la compilation. Par exemple, supposons que vous tapez ce qui suit :

```
// dans le fichier de classe Etudiant.as
class Etudiant {
  var status:Boolean; // propriété des objets Etudiant
}

// dans un script
var étudiantMarieLago:Student = new Student();
étudiantMarieLago.status = "inscrite";
```

Lorsque Flash compile ce script, une erreur « Incompatibilité de types » se produit.

Un des autres avantages du typage strict des données réside dans le fait que Flash MX 2004 affiche automatiquement des conseils de code pour les objets intégrés que vous typez strictement. Pour plus d'informations, consultez *Typage strict des objets pour déclencher des conseils de code*, page 66.

Les fichiers publiés à l'aide d'ActionScript 1 ne respectent pas les affectations de typage strict des données lors de la compilation. Ainsi, l'affectation du mauvais type de valeur à une variable que vous avez strictement typée ne génère pas d'erreur de compilation.

```
var x:String = "abc"
x = 12 ; // aucune erreur dans ActionScript 1, erreur de compatibilité de type
  dans ActionScript 2
```

Ceci est dû au fait que lorsque vous publiez un fichier pour ActionScript 1, Flash interprète une instruction telle que `var x:String = "abc"` comme une syntaxe à barre oblique et non comme un typage strict. (ActionScript 2.0 ne supporte pas la syntaxe à barre oblique.) Il peut alors en résulter un objet affecté à une variable de type erroné. Le compilateur autorise alors que des appels de méthodes illégaux et que des références de propriété non définies passent sans être signalés.

Ainsi, si vous implémentez le typage strict de données, assurez-vous de publier les fichiers pour ActionScript 2.0.

Attribution d'objets

ActionScript 2.0 vous permet d'attribuer un type de données à un autre. L'opérateur d'attribution utilisé par Flash se présente sous la forme d'un appel de fonction et concorde avec la *coercition explicite*, telle qu'elle est définie dans la norme ECMA-262 version 4. L'attribution vous permet d'affirmer qu'un objet est d'un type spécifique de telle sorte que, lors de la vérification du type, le compilateur considère que l'objet est doté de propriétés absentes du type d'origine. Ceci peut s'avérer utile, par exemple, lors de l'itération sur un tableau d'objets pouvant être de types différents.

Dans les fichiers publiés pour Flash Player 7 ou une version ultérieure, les instructions d'attribution qui échouent à l'exécution renvoient `null`. Dans les fichiers publiés pour Flash Player 6, les attributions ayant échouées ne sont pas supportées à l'exécution.

La syntaxe d'attribution est *`type(élément)`* : le compilateur doit se comporter comme si le type de données de *`élément`* était *`type`*. L'attribution est essentiellement un appel de fonction, et l'appel de fonction renvoi `null` si l'attribution échoue. Si l'attribution réussit, l'appel de fonction renvoie l'objet original. Toutefois, le compilateur ne génère pas d'erreurs d'incompatibilité de type lorsque vous attribuez des éléments à des types de données que vous avez créés dans des fichiers de classe externes, même si l'attribution échoue à l'exécution.

```
// dans Animal.as
class Animal {}

// dans Chien.as
class Chien extends Animal { function aboie (){} }

// dans Chat.as
class Chat extends Animal { function miaule (){} }

// dans le fichier FLA
var spot:Chien = new Chien();
var temp:Chat = Chat (spot);  // affirme qu'un objet Chien est de type Chat
temp.miaou(); // n'a aucun effet et n'entraîne pas d'erreur de compilation non
  plus
```

Dans cette situation, vous avez indiqué au compilateur que `temp` est un objet Chat, et le compilateur suppose donc que `temp.miaou()` est une instruction légale. Toutefois, le compilateur ne sait pas que l'attribution échouera (c'est-à-dire que vous avez essayé d'attribuer un objet Chien à un type Chat), et aucune erreur de compilation ne se produit. Si vous incorporez une vérification dans votre script de manière à vous assurer que l'attribution réussit, vous pouvez trouver des erreurs d'incompatibilité de types à l'exécution.

```
var spot:Chien = new Chien();
var temp:Chat = Chat (spot);
trace(temp);  //affiche null à l'exécution
```

Vous pouvez attribuer une expression à une interface. Si l'expression est un objet qui implémente l'interface, ou si elle possède une classe de base qui implémente l'interface, l'objet est renvoyé. Sinon, `null` est renvoyé.

L'exemple suivant montre les résultats de l'attribution de types d'objet intégrés. Comme le montre la première ligne du bloc `with(results)`, une attribution illégale (dans le cas présent, l'attribution d'une chaîne à un clip) renvoie `null`. Comme le montrent les deux dernières lignes, l'attribution vers null ou undefined renvoie `undefined`.

```
var mc:MovieClip;
var tab:Array;
var bool:Boolean;
var num3:Number;
var obj:Object;
var str:String;
_root.createTextField("results",2,100,100,300,300);
with(results){
text = "type MovieClip : "+(typeof MovieClip(str));     // renvoie null
text += "\ntype object : "+(typeof Object(str));        // renvoie object
text += "\ntype Array : "+(typeof Array(num3));         // renvoie object
text += "\ntype Boolean : "+(typeof Boolean(mc));       // renvoie boolean
text += "\ntype String : "+(typeof String(mc));         // renvoie string
text += "\ntype Number : "+(typeof Number(obj));        // renvoie number
text += "\ntype Function : "+(typeof Function(mc));     // renvoie object
text += "\ntype null : "+(typeof null(arr));            // renvoie undefined
text += "\ntype undefined : "+(typeof undefined(obj)); // renvoie undefined
}
//Résultats dans le panneau de sortie
type MovieClip : null
type object : objet
type Array : objet
type Boolean : boolean
type String : chaîne
type Number : nombre
type Function : objet
type null : undefined
type undefined : undefined
```

Vous ne pouvez pas supplanter les types de données primitifs tels que Boolean, Date et Number par un opérateur d'attribution du même nom.

A propos des variables

Une *variable* est un conteneur qui stocke des informations. 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.

Il est toujours judicieux d'affecter une valeur connue à une variable que vous définissez pour la première fois. Cette opération, appelée *initialisation de la variable*, est souvent effectuée dans la première image du fichier SWF. Elle facilite le suivi et la comparaison de la valeur de la variable pendant la lecture du fichier SWF.

Les variables peuvent contenir tous les types de données (consultez *A propos des types de données*, page 36). Le type de données contenu dans une variable affecte la façon dont est modifiée la valeur de la variable lorsqu'elle est affectée à un script.

Les types d'informations standard que vous pouvez stocker dans une variable sont les URL, les noms d'utilisateur, les résultats d'opérations mathématiques, le nombre de fois qu'un événement s'est produit ou si un bouton a été actionné. Chaque fichier SWF et chaque occurrence de clip dispose d'un jeu de variables, dont la valeur est indépendante des variables figurant dans d'autres fichiers SWF ou clips.

Pour tester la valeur d'une variable, utilisez l'action `trace()` pour envoyer la valeur au panneau de sortie. Par exemple, `trace(heuresTravaillées)` envoie la valeur de la variable `heuresTravaillées` au panneau de sortie en mode de test. Vous pouvez également vérifier et définir les valeurs des variables dans le débogueur en mode de test. Pour plus d'informations, consultez *Utilisation de l'instruction trace*, page 85 et *Affichage et modification de variables*, page 77.

Affectation d'un nom à une variable

Le nom des variables doit suivre les règles suivantes :

- Il doit s'agir d'un identifiant (consultez *Terminologie*, page 29).
- Il ne peut pas s'agir d'un mot-clé ni d'un littéral ActionScript, tel que `true`, `false`, `null` ou `undefined`.
- Il doit être unique dans son domaine (consultez *Domaine et déclaration de variables*, page 44).

Par ailleurs, n'utilisez pas les éléments du langage ActionScript comme noms de variable : cela donnerait lieu à des erreurs de syntaxe ou à des résultats inattendus. Si vous appelez une variable `String`, par exemple, puis tentez de créer un objet String au moyen de `new String()`, le nouvel objet n'est pas défini.

```
bonjour_str = new String();
trace(bonjour_str.length); // renvoie 0

String = "bonjour"; // Attribution du nom d'une classe intégrée à une variable
bonjour_str = new String();
trace(bonjour_str.length); // renvoie undefined
```

L'éditeur ActionScript prend en charge les conseils de code pour les classes intégrées et pour les variables basées sur ces classes. Si vous souhaitez obtenir des conseils de code pour un type d'objet particulier affecté à une variable, vous pouvez définir strictement le type de cette dernière ou la nommer en utilisant un suffixe spécial.

Par exemple, supposons que vous tapez ce qui suit :

```
var membres:Array = new Array();
membres.
```

Dès que vous tapez le point (.), Flash affiche la liste des méthodes et propriétés disponibles pour les objets Array. Pour plus d'informations, consultez *Rédaction de code qui déclenche des conseils de code*, page 66.

Domaine et déclaration de variables

Le *domaine* d'une variable fait référence au domaine dans lequel la variable est connue et peut être référencée. Il existe trois types de domaine de variable dans ActionScript :

- Les *Variables locales* sont disponibles dans le corps de la fonction dans lequel elles sont déclarées (délimité par des accolades).
- Les *Variables de scénario* sont disponibles pour tout script dans ce scénario.
- Les *Variables globales* et les fonctions sont visibles par tout scénario et domaine du document.

Remarque : Les classes ActionScript 2.0 que vous créez supportent des domaines de variables publics, privés et statiques. Pour plus d'informations, consultez *Contrôle de l'accès des membres*, page 172 et *Création des membres de classe*, page 174.

Variables locales

Pour déclarer des variables locales, utilisez l'instruction `var` dans le corps de la fonction. Une variable locale a un domaine limité au bloc et expire à la fin du bloc. Une variable locale qui n'est pas déclarée dans un bloc expire à la fin de son script.

Par exemple, les variables `i` et `j` sont souvent utilisées comme compteurs de boucles. Dans l'exemple suivant, `i` est utilisée comme variable locale et existe uniquement dans la fonction `constructJours()` :

```
function constructJours() {
  var i;
  for( i = 0; i < tableauDeMois[month]; i++ ) {

    _root.Jours.attachMovie( "affichageDesJours", i, i + 2000 );

    _root.Jours[i].num = i + 1;
    _root.Jours[i]._x = colonne * _root.Jours[i]._width;
    _root.Jours[i]._y = ligne * _root.Jours[i]._height;

    colonne = colonne + 1;

    if (colonne == 7 ) {

      colonne = 0;
      ligne = ligne + 1;
    }
  }
}
```

Les variables locales permettent aussi d'empêcher les conflits de noms, qui peuvent donner lieu à des erreurs dans une application. Par exemple, si vous créez la variable locale `nom`, vous pouvez l'utiliser pour stocker un nom d'utilisateur dans un contexte et une occurrence de clip dans un autre. Ces variables fonctionnant dans des domaines distincts, il n'y a pas de risques de conflits.

Il est toujours judicieux d'utiliser des variables locales dans le corps d'une fonction pour que celle-ci puisse agir en tant que partie de code indépendante. Une variable locale n'est modifiable qu'au sein de son propre bloc de code. Si une expression d'une fonction utilise une variable globale, un élément extérieur peut modifier sa valeur, ce qui modifierait la fonction.

Vous pouvez affecter un type de données à une variable locale lorsque vous la définissez, ce qui vous évite d'affecter un type de données erroné à une variable existante. Pour plus d'informations, consultez *Typage strict des données*, page 40.

Variables de scénario

Les variables de scénario sont disponibles pour tout script dans le scénario. Pour déclarer les variables de scénario, initialisez-les sur n'importe quelle image du scénario. Veillez à initialiser la variable avant de tenter d'y accéder dans un script. Si, par exemple, vous placez le code `varx = 10;` sur l'image 20, un script associé à une image précédant l'image 20 n'aura pas accès à cette variable.

Variables globales

Les variables et les fonctions globales sont visibles par tout scénario et domaine du document. Pour créer une variable globale, faites précéder son nom de l'identifiant `_global` et n'utilisez pas la syntaxe `var =`. Par exemple, le code suivant crée la variable globale `monNom` :

```
var _global.monNom = "George"; // erreur de syntaxe
_global.monNom = "George";
```

Toutefois, si vous initialisez une variable locale portant le même nom qu'une variable globale, vous n'avez pas accès à cette dernière dans le domaine de la variable locale :

```
_global.compteur = 100;
compteur++;
trace(compteur); // affiche 101
function count(){
  for( var compteur = 0; compteur <= 10 ; compteur++ ) {
  trace(compteur); // affiche 0 à 10
  }
}
count();
compteur++;
trace(compteur); // affiche 102
```

Utilisation des variables dans un programme

Vous devez déclarer une variable dans un script avant de pouvoir l'utiliser dans une expression. Si vous utilisez une variable non déclarée, comme dans l'exemple suivant, elle prend la valeur `NaN` ou `undefined`, et votre script est susceptible de générer des résultats inattendus :

```
var squared = x*x;
trace(squared); // NaN
var x = 6;
```

Dans l'exemple ci-dessous, l'instruction déclarant la variable `x` doit être placée en premier de sorte que `squared` puisse être remplacée par une valeur.

```
var x = 6;
var squared = x*x;
trace(squared); // 36
```

La même chose se produit lorsque vous transmettez une valeur non définie à une méthode ou à une fonction :

```
getURL(onSiteWeb); // aucune action
var monSiteWeb = "http://www.macromedia.com";

var monSiteWeb = "http://www.macromedia.com";
getURL(monSiteWeb); // le navigateur affiche www.macromedia.com
```

Vous pouvez changer plusieurs fois la valeur d'une variable dans un script. Le type de données contenu dans la variable affecte les conditions et le moment où la variable sera modifiée. Les types primitifs de données, comme les chaînes et les nombres, sont transmis par valeur. Cela signifie que le contenu réel de la variable est transmis à la variable.

Dans l'exemple suivant, `x` est défini sur 15 et cette valeur est copiée dans `y`. Lorsque `x` devient 30 à la ligne 3, la valeur de `y` reste 15 étant donné que `y` ne va pas chercher sa valeur dans `x` ; elle contient la valeur de `x` qu'elle a reçu à la ligne 2.

```
var x = 15;
var y = x;
var x = 30;
```

Dans un autre exemple, la variable `valeurEntrée` contient une valeur primitive, 3, la valeur réelle étant donc transmise à la fonction `sqrt()` et la valeur renvoyée étant 9 :

```
function sqrt(x){
  return x * x;
}

var valeurEntrée = 3;
var Sortie = sqrt(valeurEntrée);
```

La valeur de la variable `valeurEntrée` ne change pas.

Le type de données objet peut contenir tant d'informations complexes qu'une variable de ce type ne contient pas la valeur réelle, mais contient une référence à la valeur. Cette référence est un « alias » qui désigne le contenu de la variable. Lorsque la variable a besoin de connaître sa valeur, la référence demande le contenu et renvoie la réponse sans transférer la valeur à la variable.

Le code suivant est un exemple de transmission par référence :

```
var monTableau = ["tom", "josie"];
var nouveauTableau = monTableau;
monTableau[1] = "jack";
trace(nouveauTableau);
```

Le code ci-dessus crée un objet Array appelé `monTableau` qui contient deux éléments. La variable `nouveauTableau` est créée et reçoit une référence à `monTableau`. La modification du deuxième élément de `monTableau` affecte toutes les variables qui y font référence. L'action `trace()` envoie `tom, jack` au panneau de sortie.

Dans l'exemple suivant, `monTableau` contient un objet Array qui est transmis à la fonction `tableauNul()` par référence. La fonction `tableauNu()l` change le contenu du tableau en `monTableau`.

```
function tableauNul(leTableau){
  var i;
  for (i=0; i < leTableau.length; i++) {
    leTableau[i] = 0;
  }
}

var monTableau = new Array();
monTableau[0] = 1;
monTableau[1] = 2;
monTableau[2] = 3;
tableauNul(monTableau);
```

La fonction `tableauNul()` accepte un objet Array comme paramètre et définit tous les éléments de ce tableau sur 0. Elle peut modifier ce tableau car il est transmis par référence.

Utilisation d'opérateurs pour manipuler les valeurs des expressions

Une expression est une instruction que Flash pourra évaluer et qui renvoie une valeur. Pour créer une expression, vous pouvez associer des opérateurs et des valeurs ou appeler une fonction.

Les opérateurs sont des caractères qui spécifient comment combiner, comparer ou modifier les valeurs d'une expression. Les éléments sur lesquels les opérateurs agissent sont appelés *opérandes*. Par exemple, dans l'instruction suivante, l'opérateur + additionne la valeur d'un littéral numérique à la valeur de la variable `truc`; `truc` et `3` sont les opérandes :

```
truc + 3
```

Cette section décrit des règles générales au sujet de types courants d'opérateurs, de la priorité des opérateurs et de leur associativité. Pour plus d'informations sur chaque opérateur, de même que sur les opérateurs spéciaux n'appartenant pas à ces catégories, consultez le Chapitre 12, *Dictionnaire ActionScript*, page 215.

Priorité et associativité des opérateurs

Lorsque deux opérateurs ou plus sont utilisés dans la même instruction, certains opérateurs sont prioritaires par rapport à d'autres. ActionScript suit une hiérarchie précise pour déterminer les opérateurs à exécuter en premier. Par exemple, une multiplication est toujours effectuée avant une addition, les éléments entre parenthèses restant cependant prioritaires sur la multiplication. Donc, sans parenthèses, ActionScript effectue la multiplication en premier, comme dans l'exemple suivant :

```
total = 2 + 4 * 3;
```

Le résultat est 14.

Mais si l'addition est mise entre parenthèses, ActionScript effectue l'addition en premier :

```
total = (2 + 4) * 3;
```

Le résultat est 18.

Lorsque deux ou plusieurs opérateurs possèdent le même ordre de priorité, leur associativité détermine l'ordre dans lequel ils sont exécutés. L'associativité peut aller de la gauche vers la droite, comme de la droite vers la gauche. Par exemple, l'opérateur de multiplication a une associativité gauche-droite et les deux instructions suivantes sont donc équivalentes :

```
total = 2 * 3 * 4;
total = (2 * 3) * 4;
```

Un tableau de tous les opérateurs, de leur ordre de priorité et de leur associativité, apparaît dans l'Annexe B, *Priorité et associativité des opérateurs*, page 899.

Opérateurs numériques

Les opérateurs numériques additionnent, soustraient, multiplient, divisent et effectuent d'autres opérations arithmétiques.

L'emploi le plus courant de l'opérateur d'incrémentation est `i++` au lieu de l'opérateur `i = i+1`, qui est plus long. L'opérateur d'incrémentation peut s'utiliser avant ou après une opérande. Dans l'exemple suivant, `âge` est incrémenté en premier, puis à nouveau testé contre le nombre 30 :

```
if (++âge >= 30)
```

Dans l'exemple suivant, `âge` est incrémenté à la suite du test :

```
if (âge++ >= 30)
```

Le tableau suivant répertorie les opérateurs numériques d'ActionScript :

Opérateur	Opération effectuée
+	Addition
*	Multiplication
/	Division
%	Modulo (reste de division)
-	Soustraction
++	Incrémentation
--	Décrémentation

Opérateurs de comparaison

Les opérateurs de comparaison comparent les valeurs des expressions et renvoient une valeur booléenne (`true` ou `false`). Ces opérateurs sont surtout utilisés dans les boucles et les instructions conditionnelles. Dans l'exemple suivant, si la variable `score` correspond à `100`, un fichier SWF spécifique est chargé ; sinon, c'est un autre fichier SWF qui est chargé :

```
if (score > 100){
  loadMovieNum("gagnant.swf", 5);
} else {
  loadMovieNum("perdant.swf", 5);
}
```

Le tableau suivant répertorie les opérateurs de comparaison d'ActionScript :

Opérateur	Opération effectuée
<	Inférieur à
>	Supérieur à
<=	Inférieur ou égal à
>=	Supérieur ou égal à

Opérateurs de chaîne

L'opérateur + agit de manière spéciale sur les chaînes : il en concatène les deux opérandes. Par exemple, les instructions suivantes additionnent `"Félicitations,"` et `"Donna !"`:

```
"Félicitations, " + "Donna !"
```

Le résultat est `"Félicitations, Donna !"`. Si un seul opérande de l'opérateur + est une chaîne, Flash convertit l'autre opérande en chaîne.

Les opérateurs de comparaison >, >=, < et <= agissent également de manière particulière sur les chaînes. Ces opérateurs comparent deux chaînes pour déterminer celle qui apparaît en premier dans l'ordre alphabétique. Les opérateurs de comparaison ne comparent des chaînes que si les deux opérandes sont des chaînes. Si un seul opérande est une chaîne, ActionScript convertit les deux opérandes en nombres et effectue une comparaison numérique.

Opérateurs logiques

Les opérateurs logiques comparent des valeurs booléennes (`true` et `false`) et renvoient une troisième valeur booléenne. Par exemple, si les deux opérandes sont `true`, l'opérateur logique AND (`&&`) renvoie `true`. Si l'un des opérandes, ou les deux, est `true`, l'opérateur logique OR (`||`) renvoie `false`. Les opérateurs logiques sont souvent utilisés en complément des opérateurs de comparaison pour déterminer la condition d'une action `if`. Par exemple, dans le script suivant, si les deux expressions sont `true`, l'action if est exécutée :

```
if (i > 10 && _framesloaded > 50){
  play();
}
```

Le tableau suivant répertorie les opérateurs logiques d'ActionScript :

Opérateur	Opération effectuée
`&&`	AND logique
`||`	OR logique
`!`	NOT logique

Opérateurs au niveau du bit

Les opérateurs au niveau du bit manipulent (en interne) les nombres à virgule flottante pour les transformer en entiers 32 bits. L'opération exacte effectuée dépend de l'opérateur, mais toutes les opérations au niveau du bit évaluent chaque bit d'un entier 32 bits séparément pour calculer une nouvelle valeur.

Le tableau suivant répertorie les opérateurs au niveau du bit d'ActionScript :

Opérateur	Opération effectuée
&	AND au niveau du bit
|	OR au niveau du bit
^	XOR au niveau du bit
~	NOT au niveau du bit
<<	Décalage gauche
>>	Décalage droit
>>>	Décalage droit avec remplissage par zéros

Opérateurs d'égalité

Vous pouvez utiliser l'opérateur d'égalité (==) pour déterminer si les valeurs ou les identités de deux opérandes sont égales. Cette comparaison renvoie une valeur booléenne (`true` ou `false`). Si les opérandes sont des chaînes, des nombres ou des valeurs booléennes, ils sont comparés par valeur. Si les opérandes sont des objets ou des tableaux, ils sont comparés par référence.

Une erreur courante consiste à utiliser l'opérateur d'affectation pour contrôler l'égalité. Par exemple, le code suivant compare *x* à 2 :

```
if (x == 2)
```

Dans ce même exemple, l'expression `x = 2` est incorrecte, car elle ne compare pas les opérandes, mais affecte la valeur 2 à la variable `x`.

L'opérateur d'égalité stricte (===) est semblable à l'opérateur d'égalité, à une différence (importante) près : l'opérateur d'égalité stricte n'effectue pas de conversion de type. Si les deux opérandes sont de types différents, l'opérateur d'égalité stricte renvoie `false`. L'opérateur d'inégalité stricte (`!==`) renvoie l'inverse de l'opérateur d'égalité stricte.

Le tableau suivant répertorie les opérateurs d'égalité d'ActionScript :

Opérateur	Opération effectuée
==	Egalité
===	Egalité stricte
!=	Inégalité
!==	Inégalité stricte

Opérateurs d'affectation

Vous pouvez utiliser l'opérateur d'affectation (=) pour affecter une valeur à une variable, comme par exemple :

```
var motDePasse = "Sk8tEr";
```

Vous pouvez également utiliser l'opérateur d'affectation pour affecter plusieurs variables dans la même expression. Dans l'instruction suivante, la valeur `a` est affectée aux variables `b`, `c` et `d` :

```
a = b = c = d;
```

Vous pouvez aussi utiliser des opérateurs d'affectation composés pour combiner des opérations. Les opérateurs composés agissent sur les deux opérandes, puis affectent la nouvelle valeur au premier. Par exemple, les deux instructions suivantes sont équivalentes :

```
x += 15;
x = x + 15;
```

L'opérateur d'affectation peut également être utilisé au milieu d'une expression, comme illustré ci-dessous :

```
// s'il ne s'agit pas de vanille, afficher un message.
if ((goût =goûtGlace()) != "vanille") {
  trace ("Le goût est " + flavor + ", pas la vanille.");
}
```

Ce code équivaut au code suivant, qui est quelque peu plus long :

```
goût = goûtGlace();
if (goût != "vanille") {
  trace ("Le goût est " + flavor + ", pas la vanille.");
}
```

Le tableau suivant répertorie les opérateurs d'affectation d'ActionScript :

Opérateur	Opération effectuée
=	Affectation
+=	Addition et affectation
-=	Soustraction et affectation
*=	Multiplication et affectation
%=	Modulo et affectation
/=	Division et affectation
<<=	Décalage gauche au niveau du bit et affectation
>>=	Décalage droit au niveau du bit et affectation
>>>=	Décalage droit avec remplissage par zéros et affectation
^=	XOR au niveau du bit et affectation
|=	OR au niveau du bit et affectation
&=	AND au niveau du bit et affectation

Opérateurs point et accès tableau

Vous pouvez utiliser les opérateurs point (`.`) et accès tableau (`[]`) pour accéder aux propriétés des objets ActionScript intégrés ou personnalisés, telles que celles d'un clip.

L'opérateur point utilise le nom d'un objet dans sa partie gauche et le nom d'une propriété ou d'une variable dans sa partie droite. Le nom de propriété ou de variable ne peut pas être une chaîne ni une variable évaluée comme une chaîne ; il doit s'agir d'un identifiant. Les exemples suivants utilisent l'opérateur point :

```
année.mois = "juin";
année.mois.jour = 9;
```

Les opérateurs point et accès tableau jouent le même rôle, mais l'opérateur point prend un identifiant comme propriété alors que l'opérateur d'accès tableau évalue son contenu comme nom et accède ensuite à la valeur de cette propriété nommée. Par exemple, les expressions suivantes accèdent à la même variable `vitesse` dans le clip `fusée` :

```
fusée.vitesse;
fusée["vitesse"];
```

Vous pouvez utiliser l'opérateur d'accès tableau pour définir et récupérer dynamiquement les noms et les variables des occurrences. Par exemple, dans le code suivant, l'expression insérée dans l'opérateur `[]` est évaluée et le résultat de cette évaluation est utilisé comme nom de la variable à récupérer du clip `nom` :

```
nom["mc" + i]
```

Vous pouvez également utiliser la fonction `eval()`, comme dans l'exemple ci-dessous :

```
eval("mc" + i)
```

L'opérateur d'accès tableau peut également être utilisé dans la partie gauche d'une instruction d'affectation. Cela vous permet de définir dynamiquement les noms d'objet, de variable et d'occurrence, comme dans l'exemple suivant :

```
nom[index] = "Gary";
```

Pour créer des tableaux multidimensionnels dans ActionScript, vous construisez un tableau dont les éléments sont également des tableaux. Pour accéder aux éléments d'un tableau multidimensionnel, vous pouvez imbriquer l'opérateur accès tableau en lui-même, comme illustré ci-après :

```
var Echiquier = new Array();
for (var i=0; i<8; i++) {
  Echiquier.push(new Array(8));
}
function récupContenuPosition(ligne, colonne){
  Echiquier[ligne][colonne];
}
```

Vous pouvez contrôler qu'il ne manque pas d'opérateurs `[]` dans vos scripts ; consultez *Vérification de la syntaxe et de la ponctuation*, page 71.

Définition du chemin d'un objet

Pour utiliser une action afin de contrôler un clip ou un fichier SWF chargé, vous devez spécifier son nom et son adresse, qui constituent le *chemin cible*.

Dans ActionScript, un clip est identifié par son nom d'occurrence. Par exemple, dans l'instruction suivante, la propriété `_alpha` du clip nommé `étoile` est définie sur une visibilité de 50 % :

```
étoile._alpha = 50;
```

Pour donner un nom d'occurrence à un clip :

1 Sélectionnez le clip sur la scène.

2 Entrez un nom d'occurrence dans l'inspecteur des propriétés.

Pour identifier un fichier SWF chargé :

- Utilisez `_level`*X*, où *X* est le numéro du niveau spécifié dans l'action `loadMovie()` qui a chargé le fichier SWF.

  Par exemple, un fichier SWF chargé dans le niveau 5 a pour chemin cible `_level5`. Dans l'exemple suivant, un fichier SWF est chargé dans le niveau 5 et sa visibilité est définie sur `false` :

  ```
  onClipEvent(load) {
    loadMovieNum("monAnimation.swf", 5);
  }
  onClipEvent(enterFrame){
    _level5._visible = false;
  }
  ```

Pour entrer le chemin cible d'un fichier SWF :

- Dans le panneau Actions (Fenêtre > Développement > Actions), cliquez sur le bouton Insérer un chemin cible et sélectionnez un clip dans la liste qui apparaît.

Pour plus d'informations sur les chemins cibles, consultez « Chemins cibles absolus et relatifs » dans le guide Utilisation de Flash del'aide.

Utilisation de fonctions intégrées

Une fonction est un bloc de code ActionScript qui peut être réutilisé n'importe où dans un fichier SWF. Si vous transmettez des valeurs en tant que paramètres à une fonction, cette dernière agit en fonction de ces valeurs. Une fonction peut également renvoyer des valeurs.

Flash possède des fonctions intégrées qui permettent d'accéder à certaines informations et d'exécuter certaines tâches, comme l'obtention du numéro de version de Flash Player qui héberge le fichier SWF (`getVersion()`). Les fonctions appartenant à un objet sont appelées *méthodes*. Les fonctions qui n'appartiennent pas à un objet sont appelées *fonctions de niveau supérieur* et se trouvent dans la catégorie Fonctions du panneau Actions.

Chaque fonction possède ses propres caractéristiques, certaines fonctions vous obligeant à transmettre certaines valeurs. Si vous transmettez plus de paramètres qu'il n'est nécessaire à la fonction, les valeurs supplémentaires sont ignorées. Si vous ne transmettez pas un paramètre requis, les paramètres vides reçoivent le type de données `undefined`, ce qui peut provoquer des erreurs à l'exportation du script. Pour appeler une fonction, celle-ci doit se trouver dans l'image que la tête de lecture a atteinte.

Pour appeler une fonction, utilisez tout simplement son nom et transmettez les paramètres requis :

```
isNaN(someVar);
getTimer()
eval("someVar");
```

Pour plus d'informations sur une fonction, consultez son entrée dans le Chapitre 12, *Dictionnaire ActionScript*, page 215.

Création de fonctions

Vous pouvez définir des fonctions pour exécuter une série d'instructions sur des valeurs transmises. Vos fonctions peuvent également renvoyer des valeurs. Une fois définie, une fonction peut être appelée à partir de tout scénario, y compris celui d'un fichier SWF chargé.

Une fonction bien rédigée peut être considérée comme une « boîte noire ». Si elle contient des commentaires positionnés judicieusement au sujet de son entrée, sa sortie et son rôle, l'utilisateur de la fonction ne doit pas nécessairement comprendre tout son fonctionnement interne.

Définition d'une fonction

Les fonctions, tout comme les variables, sont associées au scénario du clip qui les définit, et vous devez utiliser un chemin cible pour les appeler. Tout comme dans le cas d'une variable, vous pouvez utiliser l'identificateur `_global` pour déclarer une fonction globale disponible pour tous les scénarios sans emploi d'un chemin cible. Pour définir une fonction globale, faites précéder son nom de l'identificateur `_global`, comme illustré ci-dessous :

```
_global.maFonction = function (x) {
   return (x*2)+3;
}
```

Pour définir une fonction de scénario, utilisez l'action `function` suivie du nom de la fonction, des paramètres qui doivent être transmis à la fonction et des instructions ActionScript qui indiquent ce que la fonction fait.

L'exemple suivant illustre une fonction nommée `aireDuCercle` et dotée du paramètre `rayon` :

```
function aireDuCercle(rayon) {
  return Math.PI * rayon * rayon;
}
```

Vous pouvez également définir une fonction en créant un *littéral de fonction*, une fonction sans nom qui est déclarée dans une expression au lieu d'une instruction. Vous pouvez utiliser un littéral de fonction pour définir une fonction, renvoyer sa valeur et l'affecter à une variable dans une expression, comme illustré dans l'exemple suivant :

```
aire = (function() {return Math.PI * rayon *rayon;})(5);
```

Lorsqu'une fonction est redéfinie, la nouvelle définition remplace l'ancienne.

Transmission de paramètres à une fonction

Les paramètres sont les éléments sur lesquels une fonction exécute son code. Dans cet ouvrage, les termes *paramètre* et *argument* sont interchangeables. Par exemple, la fonction suivante prend les paramètres `initiales` et `scoreFinal`:

```
function remplirScores(initiales, scoreFinal) {
  carteDeScore.affichage = initiales;
  carteDeScore.score = scoreFinal;
}
```

Lorsque la fonction est appelée, les paramètres requis doivent lui être transmis. La fonction substitue les valeurs transmises aux paramètres de la définition de la fonction. Dans cet exemple, `carteDeScore` est le nom d'occurrence d'un clip, `affichage` et `score` étant des champs de texte de saisie dans l'occurrence. L'appel de fonction suivant affecte la valeur `"JEB"` à la variable `display` et la valeur `45000` à la variable `affichage` :

```
remplirScores("JEB", 45000);
```

Le paramètre `initiales` de la fonction `remplirScores()` est similaire à une variable locale : il existe tant que la fonction est appelée et cesse d'exister à la sortie de la fonction. Si vous omettez des paramètres lors de l'appel d'une fonction, les paramètres omis sont transmis comme `undefined`. Si vous fournissez des paramètres supplémentaires dans un appel de fonction alors qu'ils ne sont pas requis par la déclaration de la fonction, ces paramètres sont ignorés.

Utilisation de variables dans une fonction

Les variables locales sont des outils qui simplifient grandement l'organisation du code et en facilitent la compréhension. Lorsqu'une fonction utilise des variables locales, elle peut les cacher à tous les autres scripts du fichier SWF. Les variables locales ont un domaine limité au corps de la fonction et sont détruites à la sortie de celle-ci. Tous les paramètres transmis à une fonction sont également traités comme des variables locales.

Vous pouvez également utiliser des variables globales et normales dans une fonction. Cependant, si vous modifiez des variables globales ou normales dans une fonction, il est judicieux d'utiliser des commentaires pour documenter ces modifications.

Renvoi de valeurs d'une fonction

Utilisez l'instruction `return` pour renvoyer les valeurs des fonctions. L'instruction `return` stoppe la fonction et la remplace par la valeur de l'action `return`. Les règles suivantes gouvernent l'utilisation de l'instruction `return` dans les fonctions :

- Si vous spécifiez un type de renvoi autre que void pour une fonction, vous devez inclure une instruction `return` dans la fonction.
- Si vous spécifiez un type de renvoi void, n'incluez pas d'instruction `return`.
- Si vous ne spécifiez pas de type de renvoi, l'incorporation d'une instruction `return` est facultative. Si vous n'en incluez pas, une chaîne vide est renvoyée.

Par exemple, la fonction suivante renvoie le carré du paramètre `x` et spécifie que la valeur renvoyée doit être du type Number :

```
function sqr(x):Number {
  return x * x;
}
```

Certaines fonctions effectuent une série de tâches sans renvoyer de valeur. Par exemple, la fonction suivante initialise une série de variables globales :

```
function initialize() {
  bateau_x = _global.bateau._x;
  bateau_y = _global.bateau._y;
  voiture_x = _global.voiture._x;
  voiture_y = _global.voiture._y;
}
```

Appel d'une fonction définie par l'utilisateur

Vous pouvez utiliser un chemin cible pour appeler une fonction d'un scénario, quel qu'il soit, à partir de n'importe quel autre scénario, y compris celui d'un fichier SWF chargé. Si une fonction a été déclarée au moyen de l'identificateur `_global`, il n'est pas nécessaire de l'appeler à l'aide d'un chemin cible.

Pour appeler une fonction, entrez le chemin cible du nom de la fonction, si nécessaire, puis transmettez les paramètres requis entre parenthèses. Par exemple, l'instruction suivante appelle la fonction `sqr()` du clip `MathLib` du scénario principal, lui transmet le paramètre 3 et enregistre le résultat dans la variable `temp` :

```
var temp = _root.MathLib.sqr(3);
```

L'exemple suivant utilise un chemin absolu pour appeler la fonction `initialiser()` qui a été définie dans le scénario principal et n'exige aucun paramètre :

```
_root.initialiser();
```

L'exemple suivant utilise un chemin relatif pour appeler la fonction `liste()` qui a été définie dans le clip `functionsClip` :

```
_parent.clipDeFonctions.liste(6);
```

CHAPITRE 3
Rédaction et débogage de scripts

Dans Macromedia Flash MX 2004 et Macromedia Flash MX Professionnel 2004, vous pouvez rédiger des scripts intégrés à votre fichier FLA ou stockés en externe sur votre ordinateur. (Si vous rédigez des fichiers de classe ActionScript 2.0, stockez chaque classe comme un fichier externe du même nom.) Pour rédiger des scripts intégrés, utilisez le panneau Actions et associez l'action à un bouton, à un clip ou à une image dans le scénario (consultez *Contrôle de l'exécution d'ActionScript*, page 60). Pour rédiger des fichiers de script externes, vous pouvez utiliser n'importe quel éditeur de texte ou de code. Dans Flash Professionnel, vous pouvez aussi utiliser la fenêtre de script intégrée. Pour plus d'informations, consultez *Utilisation du panneau Actions et de la fenêtre de script*, page 62.

Lorsque vous utilisez l'éditeur ActionScript, vous pouvez également rechercher les erreurs de syntaxe, mettre automatiquement le code en forme et utiliser des conseils pour compléter la syntaxe du code. De plus, la fonction d'équilibrage de la ponctuation vous aide à compléter les paires de parenthèses, de crochets ou d'accolades. Pour plus d'informations, consultez *Utilisation de l'éditeur ActionScript*, page 65.

Lorsque vous travaillez dans un document, nous ne pouvons que vous recommander de le tester assez fréquemment afin de vous assurer que sa lecture est aussi fluide que possible et qu'elle s'effectue de la manière prévue. Vous pouvez utiliser le testeur de bande passante pour simuler la lecture de votre document en fonction de différentes vitesses de connexion (consultez Test des performances de téléchargement des documents dans le manuel Utilisation de Flash de l'aide). Une version de débogage spéciale de Flash Player qui facilite la résolution des problèmes vous permet de tester vos scripts. Si vous utilisez de bonnes techniques de programmation dans votre code ActionScript, vos scripts seront plus faciles à dépanner si un problème imprévu est rencontré. Pour plus d'informations, consultez *Débogage de scripts*, page 73.

Contrôle de l'exécution d'ActionScript

Lorsque vous rédigez un script, vous utilisez le panneau Actions pour associer le script à une image d'un scénario, ou à un bouton ou un clip sur la scène. Les scripts associés à une image sont *exécutés* lorsque la tête de lecture entre dans cette image. Toutefois, il se peut que les scripts associés à la première image d'un fichier SWF se comportent différemment de ceux qui sont associés aux images suivantes. La première image est en effet rendue de manière incrémentielle (les objets sont tracés sur la scène au fur et à mesure de leur téléchargement dans Flash Player), ce qui peut influer sur le moment où les actions sont exécutées. Toutes les images qui suivent la première sont rendues en une seule fois, lorsque tous les objets qu'elles contiennent sont disponibles.

Les scripts associés à des clips ou des boutons sont exécutés lorsqu'un événement se produit. Un *événement* correspond à une occurrence dans le fichier SWF telle qu'un mouvement de souris, une pression sur une touche ou le chargement d'un clip. Vous pouvez utiliser ActionScript pour déterminer quand de tels événements se produisent et exécuter des scripts en fonction de l'événement. Pour plus d'informations, consultez le Chapitre 4, *Gestion d'événements*, page 89.

Pour exécuter une action selon qu'une condition existe ou non, ou pour répéter une action, vous pouvez utiliser les instructions `if`, `else`, `else if`, `for`, `while`, `do while`, `for..in` ou `switch`, qui sont décrites brièvement dans la suite de cette section.

Vérification d'une condition

Les instructions qui vérifient si une condition est `true` ou `false` commencent par le terme `if`. Si la condition existe, ActionScript exécute l'instruction qui suit. Si la condition n'existe pas, ActionScript passe à l'instruction suivante, à l'extérieur du bloc de code.

Pour optimiser les performances de votre code, vérifiez d'abord les conditions les plus probables.

Les instructions suivantes testent trois conditions. Le terme `else if` spécifie d'autres tests à effectuer si les conditions précédentes sont false.

```
if (motDePasse == null || email == null) {
  gotoAndStop("refuser");
} else if (motDePasse == iDutilisateur){
  gotoAndPlay("démarrerAnim");
}
```

Pour vérifier une condition parmi d'autres, vous pouvez utiliser l'instruction `switch`, plutôt que plusieurs instructions `else if`.

Répétition d'une action

ActionScript peut répéter une action un certain nombre de fois précisé ou tant qu'une condition spécifique existe. Utilisez les actions `while`, `do..while`, `for` et `for..in` pour créer des boucles.

Pour répéter une action tant qu'une condition existe :

- Utilisez l'instruction `while`.

Une boucle `while` évalue une expression et exécute le code dans le corps de la boucle si l'expression est `true`. L'expression est évaluée à nouveau après l'exécution de chaque instruction du corps. Dans l'exemple suivant, la boucle est exécutée quatre fois :

```
i = 4;
while (var i > 0) {
  mon_mc.duplicateMovieClip("nouveauMC" + i, i );
  i--;
}
```

Vous pouvez utiliser l'instruction `do..while` pour créer le même genre de boucle qu'avec une boucle `while`. Dans une boucle `do..while`, l'expression est évaluée à la fin du bloc de code et la boucle est toujours exécutée au moins une fois, comme dans l'exemple suivant :

```
i = 4;
do {
  mon_mc.duplicateMovieClip("nouveauMC" +i, i );
  i--;
} while (var i > 0);
```

Pour répéter une action en utilisant un compteur intégré :

- Utilisez l'instruction `for`.

La plupart des boucles utilisent un compteur d'un certain type pour contrôler le nombre de fois qu'une boucle est exécutée. Chaque exécution d'une boucle est appelée *itération*. Vous pouvez déclarer une variable et rédiger une instruction qui augmente ou diminue la variable chaque fois que la boucle est exécutée. Dans l'action `for`, le compteur et l'instruction qui l'incrémente font partie de l'action. Dans l'exemple suivant, la première expression (`var i = 4`) est l'expression initiale évaluée avant la première itération. La deuxième expression (`i > 0`) est la condition contrôlée à chaque fois avant l'exécution de la boucle. La troisième expression (`i--`) est appelée *post-expression* et est évaluée à chaque fois après l'exécution de la boucle.

```
for (var i = 4; i > 0; i--){
  monMC.duplicateMovieClip("nouveauMC" + i, i + 10);
}
```

Pour passer en boucle sur les enfants d'un clip ou d'un objet :

- Utilisez l'instruction `for..in`.

Les enfants sont composés d'autres clips, fonctions, objets et variables. L'exemple suivant utilise l'instruction `trace` pour envoyer les résultats dans le panneau de sortie :

```
monObjet = { nom:'Joe', âge:25, ville:'San Francisco' };
for (nomDeProp in monObjet) {
  trace("monObjet a la propriété : " + nomDeProp + ", avec la valeur : " +
  monObjet[nomDeProp]);
}
```

Cet exemple donne les résultats suivants dans le panneau de sortie :

```
monObjet a la propriété : nom, avec la valeur : Joe
monObjet a la propriété : âge, avec la valeur : 25
monObjet a la propriété : ville, avec la valeur : San Francisco
```

Vous pouvez souhaiter que votre script itère sur un type particulier d'enfants, par exemple, seulement sur les enfants d'un clip. Vous pouvez le faire avec `for..in` en conjonction avec l'opérateur `typeof`.

```
for (nom dans monClip) {
  if (typeof (monClip[nom]) == "clip") {
    trace("J'ai un clip enfant appelé " + nom);
  }
}
```

Pour plus d'informations sur chaque action, consultez les entrées correspondantes dans le Chapitre 12, *Dictionnaire ActionScript*, page 215.

Utilisation du panneau Actions et de la fenêtre de script

Vous pouvez intégrer des scripts Flash à votre fichier FLA ou les enregistrer dans des fichiers externes. Dans la mesure du possible, il est préférable d'enregistrer la majorité du code ActionScript dans des fichiers externes. Il est en effet plus simple d'utiliser du code dans plusieurs fichiers FLA. Ensuite, dans votre fichier FLA, créez un script comportant des instructions `#include` pour accéder au code stocké en externe. Utilisez le suffixe .as pour identifier vos scripts en tant que fichiers ActionScript (AS). (Si vous créez des fichiers de classe personnalisés, vous devez les enregistrer en tant que fichiers AS externes.)

Remarque : Lors de la publication, de l'exportation, du test ou du débogage d'un fichier FLA, le code ActionScript enregistré dans des fichiers externes est compilé dans un fichier SWF. Si vous modifiez un fichier externe, vous devez donc l'enregistrer et recompiler tout fichier FLA qui l'utilise.

Vous pouvez associer à des images et à des objets le code ActionScript que vous intégrez dans un fichier FLA. Dans la mesure du possible, associez le code ActionScript intégré à la première image du scénario. Votre code ne sera ainsi pas éparpillé et vous ne serez pas obligé de le rechercher dans tout le fichier FLA. Créez un calque appelé « Actions » et placez-y votre code. Ainsi, même si vous associez du code à d'autres images ou à des objets, il vous suffira de consulter un seul calque pour le retrouver.

Pour créer des scripts qui font partie de votre document, accédez directement à ActionScript via le panneau Actions. Pour créer des scripts externes, utilisez votre éditeur de texte préféré ou, dans Flash Professionnel, utilisez la fenêtre de script. Lorsque vous utilisez le panneau Actions ou la fenêtre de script, vous utilisez le même éditeur ActionScript et tapez votre code dans la fenêtre de script, sur le côté droit du panneau ou de la fenêtre. Pour éviter d'avoir à rédiger toutes les informations, vous pouvez également sélectionner ou faire glisser des actions de la boîte à outils Actions vers la fenêtre de script.

Pour afficher le panneau Actions, effectuez l'une des opérations suivantes :

- Choisissez Fenêtre > Panneaux de développement > Actions.
- Appuyez sur la touche F9.

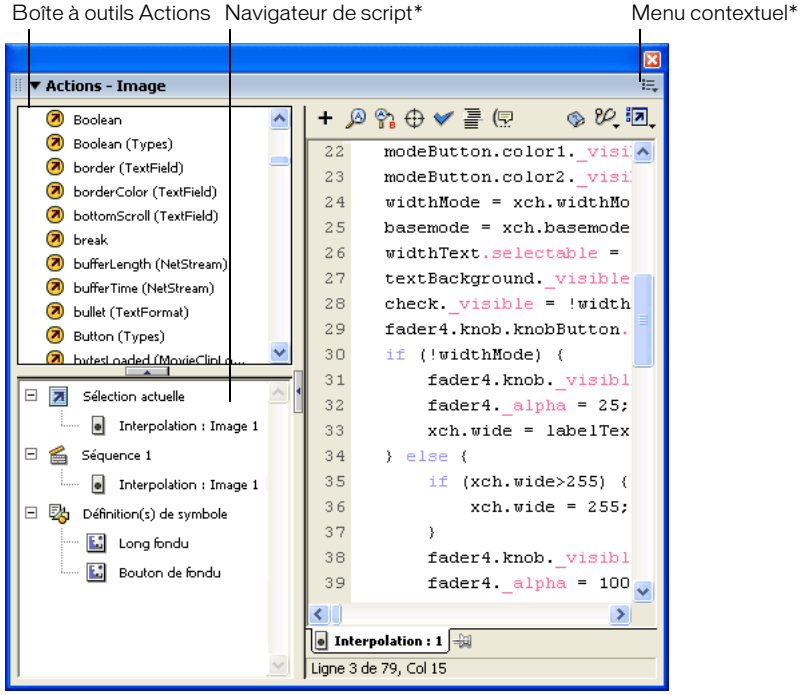
(Flash Professionnel uniquement) Pour afficher la fenêtre de script, effectuez l'une des opérations suivantes :

- Pour commencer à rédiger un nouveau script, choisissez Fichier > Nouveau > Fichier ActionScript.
- Pour ouvrir un script existant, choisissez Fichier > Ouvrir, puis ouvrez un fichier ActionScript (AS) existant.
- Pour modifier un script déjà ouvert, cliquez sur l'onglet du document qui affiche le nom du script. (Les onglets de document sont uniquement disponibles sous Microsoft Windows.)

A propos de l'environnement de l'éditeur ActionScript

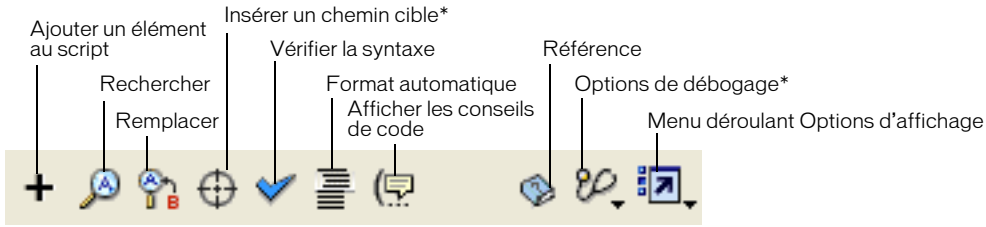
L'environnement de l'éditeur ActionScript se compose de deux sections. A droite figure la fenêtre de script, dans laquelle vous saisissez le code. A gauche se trouve la boîte à outils Actions, qui comprend une entrée pour chaque élément du langage ActionScript.

Dans le panneau Actions, la boîte à outils contient également un navigateur de script, qui est une représentation visuelle des emplacements du fichier FLA auxquels du code ActionScript est associé. Ce fichier vous permet de parcourir votre fichier FLA pour localiser le code ActionScript. Si vous cliquez sur un élément dans le navigateur de script, le script correspondant s'affiche dans la fenêtre de script et la tête de lecture se déplace jusqu'à la position appropriée dans le scénario. Si vous double-cliquez sur un élément dans le navigateur de script, le script est verrouillé (consultez *Gestion de scripts dans un fichier FLA*, page 64).



* Panneau Actions uniquement

Plusieurs boutons figurent également au-dessus de la fenêtre de script :

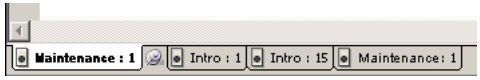


* Panneau Actions uniquement

Vous modifiez des actions, entrez des paramètres pour les actions ou supprimez des actions directement dans la fenêtre de script. Vous pouvez également double-cliquer sur un élément de la boîte à outils ou sur le bouton Ajouter (+) en haut de la fenêtre de script pour ajouter des actions dans la fenêtre de script.

Gestion de scripts dans un fichier FLA

Si vous disséminez le code dans un fichier FLA, vous pouvez *verrouiller* plusieurs scripts dans le panneau Actions pour faciliter le passage de l'un à l'autre. Dans l'illustration suivante, le script associé à l'emplacement en cours du scénario se trouve sur l'image 1 du calque Nettoyage. (L'onglet le plus à gauche suit toujours votre emplacement dans le scénario.) Ce script est également verrouillé (il est représenté par l'onglet le plus à droite). Deux autres scripts sont verrouillés : un sur l'image 1 et l'autre sur l'image 15 du calque Intro. Pour passer d'un script verrouillé à l'autre, cliquez sur les onglets correspondants ou utilisez les raccourcis clavier. Cette opération n'a aucune incidence sur votre position actuelle dans le scénario.



Conseil : Si le contenu affiché dans la fenêtre de script ne change pas de manière à refléter l'emplacement que vous sélectionnez dans le scénario, la fenêtre de script affiche probablement un script verrouillé. Cliquez sur l'onglet le plus à gauche dans la partie inférieure gauche de la fenêtre de script pour afficher le code ActionScript associé à votre emplacement sur le scénario.

Pour verrouiller un script :

1 Pointez sur le scénario afin que le script apparaisse dans un onglet situé en bas à gauche de la fenêtre de script dans le panneau Actions.

2 Effectuez l'une des opérations suivantes:

- Cliquez sur l'icône en forme de punaise qui figure à droite de l'onglet. (Si la punaise ressemble à l'icône la plus à gauche, le script est déjà verrouillé. Cliquez sur l'icône pour le déverrouiller.)
- Cliquez sur l'onglet avec le bouton droit de la souris (Windows) ou en appuyant sur la touche Contrôle (Macintosh) et choisissez Verrouiller le script.
- Choisissez Verrouiller le script dans le menu contextuel Options (dans le coin supérieur droit du panneau).

Pour déverrouiller un ou plusieurs scripts :

- Effectuez l'une des opérations suivantes:

  - Si un script verrouillé est affiché dans un onglet situé en bas à gauche de la fenêtre de script, dans le panneau Actions, cliquez sur l'icône en forme de punaise située à droite de l'onglet. (Si la punaise ressemble à l'icône la plus à gauche, le script est déjà déverrouillé. Cliquez sur l'icône pour le verrouiller.)
  - Cliquez du bouton droit de la souris (Windows) ou cliquez en appuyant sur la touche Contrôle (Macintosh) sur l'onglet et choisissez Fermer le script ou Fermer tous les scripts.

  - Choisissez Fermer le script ou Fermer tous les scripts dans le menu d'options contextuel (dans le coin supérieur droit du panneau).

Pour utiliser les raccourcis clavier avec les scripts verrouillés :

- Vous pouvez utiliser les raccourcis clavier suivants avec les scripts verrouillés :

Action	Raccourci clavier Windows	Raccourci clavier Macintosh
Verrouiller le script	Contrôle-= (signe égal)	Commande-=
Déverrouiller le script	Contrôle-- (signe moins)	Commande--
Déplacer le focus vers l'onglet de droite	Contrôle-Maj-. (point)	Commande-Maj-.
Déplacer le focus vers l'onglet de gauche	Contrôle-Maj-, (virgule)	Commande-Maj-,
Déverrouiller tous les scripts	Contrôle-Maj-- (moins)	Commande-Maj--

Utilisation de l'éditeur ActionScript

Flash MX 2004 et Flash MX Professionnel 2004 comportent plusieurs outils pour vous aider à rédiger du code conforme à la syntaxe. Ils vous donnent également la possibilité de définir des préférences de formatage du code et d'autres options. Ces fonctionnalités sont présentées dans la section suivante.

Mise en évidence de la syntaxe

Dans ActionScript, comme dans tout autre langage, la *syntaxe* est la manière dont les éléments sont assemblés afin de créer du sens. Vos scripts ne fonctionneront pas si vous utilisez une syntaxe ActionScript incorrecte.

Lorsque vous rédigez des scripts dans Flash MX 2004 et Flash MX Professionnel 2004, les commandes qui ne sont pas prises en charge par la version du lecteur que vous ciblez s'affichent en jaune dans la boîte à outils du panneau Actions. Par exemple, si la version de Flash Player SWF est définie sur Flash 6, le code ActionScript qui est uniquement pris en charge par Flash Player 7 apparaît en jaune dans la boîte à outils. (Pour plus d'informations sur la définition de la version de Flash Player SWF, consultez Définition des options de publication pour le format de fichier Flash SWF dans le guide Utilisation de Flash de l'aide.)

Vous pouvez également définir une préférence pour que les « codes-couleurs » de Flash s'intègrent à vos scripts, à mesure que vous les rédigez, afin de vous signaler les fautes de frappe. Par exemple, supposons que vous ayez défini votre préférence de coloration de la syntaxe de sorte à ce que les mots-clés s'affichent en vert foncé. Lors de la saisie, si vous entrez `var`, le mot `var` s'affiche en vert. Toutefois, si vous tapez `vae` par inadvertance, le mot `vae` restera en noir, vous indiquant ainsi que vous avez fait une faute de frappe.

Pour définir des préférences de coloration de la syntaxe en cours de frappe, effectuez l'une des opérations suivantes :

- Choisissez Edition > Préférences et définissez les paramètres Coloration de la syntaxe dans l'onglet ActionScript.
- Dans le menu contextuel Options (dans le coin supérieur droit du panneau Actions), choisissez Préférences et définissez les paramètres Coloration de la syntaxe dans l'onglet ActionScript.

Rédaction de code qui déclenche des conseils de code

Lorsque vous travaillez dans l'éditeur ActionScript (dans le panneau Actions ou la fenêtre de script), Flash peut détecter l'action que vous entrez et afficher un *conseil de code* (une info-bulle contant la syntaxe complète de l'action en cours ou un menu contextuel répertoriant des noms de propriétés ou de méthodes possibles). Des conseils de code apparaissent pour les paramètres, propriétés et événements lorsque vous tapez ou nommez strictement vos objets de sorte que l'éditeur ActionScript sache quels conseils de code afficher, comme l'indique la suite de cette section. Pour plus d'informations sur l'utilisation des conseils de code lorsque ceux-ci apparaissent, consultez *Utilisation des conseils de code*, page 68.

Remarque : Les conseils de code sont activés automatiquement pour les classes natives pour lesquelles vous n'avez pas besoin de créer ni de nommer un objet de la classe, telles que Math, Key, Mouse, etc.

Typage strict des objets pour déclencher des conseils de code

Lorsque vous utilisez ActionScript 2.0, vous pouvez définir strictement le type d'une variable qui est basée sur une classe intégrée, telle que Button, Array, etc. Si vous faites cela, l'éditeur ActionScript affiche des conseils de code relatifs à la variable. Par exemple, supposons que vous tapez ce qui suit :

```
var noms:Array = new Array();
noms.
```

Dès que vous tapez le point final (`.`), Flash affiche la liste des méthodes et propriétés disponibles pour les objets Array, car vous avez défini la variable comme étant de type tableau. Pour plus d'informations sur le typage des données, consultez *Typage strict des données*, page 40. Pour plus d'informations sur l'utilisation des conseils de code lorsque ceux-ci apparaissent, consultez *Utilisation des conseils de code*, page 68.

Utilisation de suffixes pour déclencher des conseils de code

Si vous utilisez ActionScript 1 ou que vous souhaitez afficher des conseils de code pour des objets que vous créez sans définir strictement leur type (consultez *Typage strict des objets pour déclencher des conseils de code*, page 66), vous devez ajouter un suffixe spécial au nom de chaque objet lors de sa création. Par exemple, les suffixes qui déclenchent des conseils de code pour les classes Array et Camera sont respectivement `_array` et `_cam`. Si vous tapez le code suivant :

```
var mon_array = new Array();
var ma_cam = Camera.get();
```

et que vous tapez le nom de la variable suivi d'un point, comme indiqué ci-dessous, des conseils de code pour les objets Array et Camera s'affichent respectivement.

```
mon_array.
ma_cam.
```

Pour les objets qui apparaissent sur la scène, entrez le suffixe dans le champ Nom de l'occurrence, dans l'inspecteur des propriétés. Par exemple, pour afficher des conseils de code pour des objets MovieClip, utilisez l'inspecteur des propriétés pour affecter des noms d'occurrence portant le suffixe `_mc` à tous les objets MovieClip. Lorsque vous taperez le nom d'une occurrence suivi d'un point, des conseils de code d'afficheront.

Même si les suffixes ne sont pas nécessaires au déclenchement des conseils de code lorsque vous définissez strictement le type d'un objet, il est recommandé de les utiliser de façon cohérente pour rendre vos scripts plus compréhensibles.

Le tableau suivant répertorie les suffixes requis pour le support des conseils de code automatiques :

Type d'objet	Suffixe de variable
Array	_array
Button	_btn
Camera	_cam
Color	_color
ContextMenu	_cm
ContextMenuItem	_cmi
Date	_date
Error	_err
LoadVars	_lv
LocalConnection	_lc
Microphone	_mic
MovieClip	_mc
MovieClipLoader	_mcl
PrintJob	_pj
NetConnection	_nc
NetStream	_ns
SharedObject	_so
Sound	_sound
String	_str
TextField	_txt

Type d'objet	Suffixe de variable
TextFormat	_fmt
Video	_video
XML	_xml
XMLNode	_xmlnode
XMLSocket	_xmlsocket

Pour plus d'informations sur l'utilisation des conseils de code lorsque ceux-ci apparaissent, consultez *Utilisation des conseils de code*, page 68.

Utilisation de commentaires pour déclencher des conseils de code

Vous pouvez également utiliser des commentaires ActionScript pour spécifier la classe d'un objet pour les conseils de code. L'exemple suivant indique à ActionScript que la classe de l'occurrence `Lobjet` est Object, et ainsi de suite. Si vous entrez le code `mc` suivi d'un point après ces commentaires, un conseil de code affiche la liste de méthodes et de propriétés MovieClip. Si vous entrez le code `leTableau` suivi d'un point, un conseil de code affiche une liste de méthodes et de propriétés Array, et ainsi de suite.

```
// Object Lobjet;
// Array leTableau;
// MovieClip mc;
```

Toutefois, Macromedia recommande d'utiliser le typage strict des données (consultez *Typage strict des objets pour déclencher des conseils de code*, page 66) ou les suffixes (consultez *Utilisation de suffixes pour déclencher des conseils de code*, page 66) plutôt que cette technique, car ces techniques permettent d'obtenir des conseils de code automatiquement et rendent votre code plus compréhensible.

Utilisation des conseils de code

Les conseils de code sont activés par défaut. Des préférences vous permettent de désactiver les conseils de code ou de déterminer la vitesse à laquelle ils s'affichent. Lorsque les conseils de code sont désactivés dans les préférences, il est toujours possible d'afficher un conseil de code pour une commande spécifique.

Pour définir des paramètres pour les conseils de code automatiques, effectuez l'une des opérations suivantes :

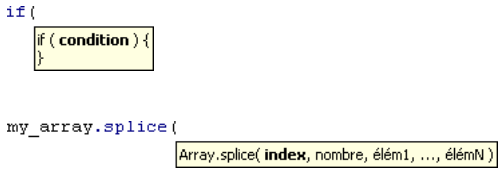
- Choisissez Edition > Préférences, puis, dans l'onglet ActionScript, activez ou désactivez l'option Conseils de code.
- Dans le panneau Actions, dans le menu contextuel Options (dans le coin supérieur droit du panneau), choisissez Préférences et, sous l'onglet ActionScript, activez ou désactivez Conseils de code.

Si vous activez les conseils de code, vous pouvez également spécifier un délai d'affichage, en secondes. Par exemple, si vous découvrez ActionScript, il pourrait être souhaitable de ne spécifier aucun délai, de sorte que les conseils de code s'affichent toujours immédiatement. Toutefois, si vous savez en général ce que vous voulez taper et que vous avez besoin de conseils uniquement lorsque vous utilisez des éléments de langage que vous ne connaissez pas, vous pouvez spécifier un délai de sorte que les conseils de code ne s'affichent pas lorsque vous ne le souhaitez pas.

Pour utiliser des conseils de code de type info-bulle :

1 Faites apparaître un conseil de code en tapant une parenthèse d'ouverture [(] après un élément qui nécessite l'utilisation des parenthèses, tel qu'un nom de méthode, une commande comme `if` ou `do while`, etc.

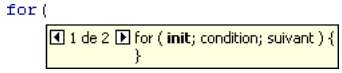
Le conseil de code apparaît.



Remarque : Si le conseil de code ne s'affiche pas, assurez-vous que l'option Conseils de code est activée dans l'onglet ActionScript. Pour afficher des conseils de code pour une variable ou un objet que vous avez créé(e), assurez-vous que vous avez nommé correctement la variable ou l'objet (consultez *Utilisation de suffixes pour déclencher des conseils de code*, page 66), ou que vous avez défini strictement son type (consultez *Typage strict des objets pour déclencher des conseils de code*, page 66).

2 Entrez une valeur pour le paramètre. S'il existe plusieurs paramètres, séparez les valeurs par des virgules.

Des commandes étendues telles que `gotoAndPlay()` ou `for` (il s'agit de fonctions ou de méthodes qui peuvent être invoquées selon différents jeux de paramètres) affichent un indicateur qui vous permet de choisir le paramètre que vous souhaitez définir. Cliquez sur les petits boutons fléchés ou appuyez sur Ctrl+Flèche gauche ou Ctrl+Flèche droite pour choisir le paramètre.



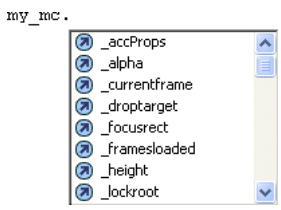
3 Pour annuler le conseil de code, effectuez l'une des opérations suivantes :

- Tapez une parenthèse de fermeture [)].
- Cliquez à l'extérieur de l'instruction.
- Appuyez sur la touche Echap.

Pour utiliser des conseils de code de type menu :

1 Faites apparaître le conseil de code en tapant un point après le nom de la variable ou de l'objet.

Le menu du conseil de code apparaît.



Remarque : Si le conseil de code ne s'affiche pas, assurez-vous que l'option Conseils de code est activée dans l'onglet ActionScript. Pour afficher des conseils de code pour une variable ou un objet que vous avez créé(e), assurez-vous que vous avez nommé correctement la variable ou l'objet (consultez *Utilisation de suffixes pour déclencher des conseils de code*, page 66), ou que vous avez défini strictement son type (consultez *Typage strict des objets pour déclencher des conseils de code*, page 66).

2 Pour naviguer dans les conseils de code, utilisez les touches Flèche Haut ou Flèche Bas.

3 Pour sélectionner un élément dans le menu, appuyez sur Entrée ou Tab, ou double-cliquez sur cet élément.

4 Pour annuler le conseil de code, effectuez l'une des opérations suivantes :

- Choisissez l'un des éléments du menu.
- Cliquez à l'extérieur de l'instruction.
- Tapez une parenthèse de fermeture [)] si vous avez déjà tapé une parenthèse d'ouverture.
- Appuyez sur la touche Echap.

Pour afficher manuellement un conseil de code :

1 Cliquez dans le code à l'endroit où les conseils de code peuvent s'afficher. En voici quelques exemples :

- Après le point qui suit une instruction ou une commande, à l'endroit où une propriété ou une méthode doit être entrée.
- Entre les parenthèses dans un nom de méthode

2 Effectuez l'une des opérations suivantes:

- Cliquez sur le bouton Afficher les conseils de code, situé au-dessus de la fenêtre de script.
- Appuyez sur Ctrl+Barre d'espace (Windows) ou sur Commande+Barre d'espace (Macintosh).
- Si vous travaillez dans le panneau Actions, ouvrez le menu contextuel (dans le coin supérieur droit de la barre de titre), puis choisissez Afficher les conseils de code.

Utilisation des touches de raccourci d'échappement

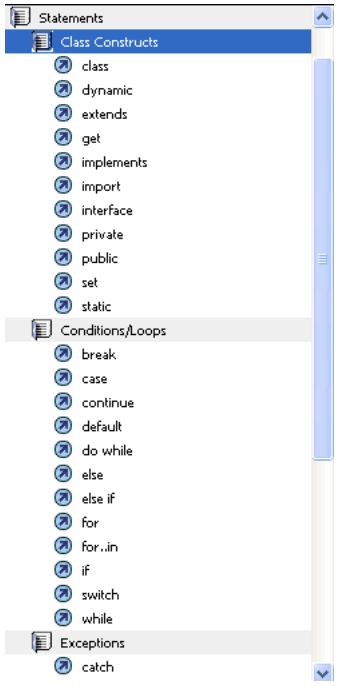
Vous pouvez ajouter plusieurs éléments à un script en utilisant des touches de raccourci (en appuyant sur la touche Echap et sur deux autres touches). (Il ne s'agit pas des mêmes raccourcis que les raccourcis clavier qui initialisent certaines commandes de menu.) Si vous saisissez Echap+d+o dans la fenêtre de script, le code suivant est inséré dans votre script, et le point d'insertion est placé immédiatement après le mot `while`, ce qui vous permet de commencer à taper la condition :

```
do {
} while ();
```

De même, si vous saisissez Echap+c+h, le code suivant est inséré dans votre script, et le point d'insertion est placé entre les parenthèses, ce qui vous permet de commencer à taper la condition :

```
catch () {
}
```

Si vous souhaitez connaître (ou revoir) les commandes qui bénéficient de touches de raccourci, vous pouvez les afficher en regard des éléments dans le panneau Actions.



Pour afficher ou masquer les touches de raccourci d'échappement :

- Dans le menu contextuel Options d'affichage, activez ou désactivez Afficher les touches de raccourci Echap.

Vérification de la syntaxe et de la ponctuation

Pour déterminer à l'avance si le code que vous avez rédigé fonctionne comme prévu, vous devez publier ou tester le fichier. Vous avez toutefois la possibilité d'effectuer une vérification rapide sans quitter le fichier FLA. Les erreurs de syntaxe sont signalées dans le panneau de sortie. (Lors de la vérification de la syntaxe, seul le script en cours est examiné ; les autres scripts que contient éventuellement le fichier FLA sont ignorés.) Vous pouvez également vérifier que les jeux de parenthèses, d'accolades ou de crochets (opérateurs d'accès tableau) encadrent correctement un bloc.

Pour vérifier la syntaxe, effectuez l'une des opérations suivantes :

- Cliquez sur le bouton Vérifier la syntaxe, situé au-dessus de la fenêtre de script.
- Dans le panneau Actions, affichez le menu contextuel (dans le coin supérieur droit du panneau) et choisissez Vérifiez la syntaxe.
- Appuyez sur Ctrl+T (Windows) ou sur Commande+T (Macintosh).

Pour vérifier l'équilibrage de la ponctuation :

1 Cliquez entre les accolades ({}), les opérateurs d'accès tableau ([]) ou les parenthèses (()) dans votre script.

2 Appuyez sur Ctrl+' (Windows) ou Commande+' (Macintosh) pour sélectionner le texte entre accolades, crochets ou parenthèses.

La mise en surbrillance vous permet de vérifier si la ponctuation d'ouverture possède une ponctuation de fermeture correspondante.

Formatage du code

Vous pouvez définir des paramètres pour déterminer si votre code est formaté et mis en retrait automatiquement ou manuellement. Vous pouvez également choisir l'affichage des numéros de ligne et le retour à la ligne automatique des lignes de code trop longues. Enfin, vous pouvez choisir d'utiliser ou non le mappage de police dynamique.

Pour définir des options de format :

1 Effectuez l'une des opérations suivantes :

- Dans le panneau Actions, dans le menu contextuel Options (dans le coin supérieur droit du panneau), choisissez Options de format automatique.
- (Flash Professionnel uniquement) Dans un fichier de script externe, choisissez Edition > Options de format automatique.

La boîte de dialogue Options de format automatique apparaît.

2 Faites votre choix parmi les options proposées. Pour visualiser l'effet de chaque sélection, examinez le panneau Afficher un aperçu.

Une fois les options de format automatique définies, vos paramètres s'appliquent automatiquement au code que vous rédigez, mais pas au code existant. Pour appliquer vos paramètres au code existant, vous devez procéder manuellement. Vous pouvez utiliser la procédure suivante pour mettre en forme du code formaté selon d'autres paramètres, importé d'un autre éditeur, etc.

Pour mettre du code en forme selon les paramètres Options de format automatique, effectuez l'une des opérations suivantes :

- Cliquez sur le bouton Format automatique, situé au-dessus de la fenêtre de script.
- Choisissez Format automatique dans le menu contextuel du panneau Actions.
- Appuyez sur Ctrl+Maj+F1 (Windows) ou sur Commande+Maj+F1 (Macintosh).

Pour utiliser la fonction de mappage de police dynamique :

- Le mappage de police dynamique est désactivé par défaut car il ralentit les performances pendant la programmation. Pour l'activer, sélectionnez Utiliser le mappage de police dynamique dans les préférences d'ActionScript.

  Il est recommandé d'activer le mappage de police dynamique pour travailler avec du texte multilingue entre autres.

Pour utiliser la fonction d'indentation automatique :

- L'indentation automatique est activée par défaut. Pour la désactiver, désactivez l'option Indentation automatique dans les préférences d'ActionScript.

  Lorsque l'indentation automatique est activée, le texte tapé après (ou { est automatiquement mis en retrait conformément à la valeur de taille de tabulation définie dans les préférences d'ActionScript. Pour mettre en retrait une autre ligne, sélectionnez la ligne souhaitée, puis appuyez sur la touche Tab. Pour supprimer le retrait, appuyez sur Maj+Tab.

Pour activer ou désactiver les numéros de ligne et le retour à la ligne automatique :

- Dans le menu contextuel Options d'affichage, activez ou désactivez Afficher les numéros de ligne et Retour à la ligne.

Débogage de scripts

Flash propose plusieurs outils qui permettent de tester le code ActionScript dans vos fichiers SWF. Le débogueur, qui est décrit dans la suite de cette section, vous permet de repérer des erreurs dans un fichier SWF en cours d'exécution dans Flash Player. Flash dispose également des outils de débogage complémentaires suivants :

- Le panneau de sortie, qui affiche des messages d'erreur et répertorie les variables et les objets (consultez *Utilisation du panneau de sortie*, page 82).
- L'instruction `trace`, qui envoie des notes de programmation et la valeur des expressions au panneau de sortie (consultez *Utilisation de l'instruction trace*, page 85).
- Les instructions `throw` et `try..catch..finally`, qui vous permettent de tester et de résoudre les erreurs d'exécution à partir de votre script.
- Des messages d'erreur de compilateur clairs, qui vous permettent de diagnostiquer et de résoudre les problèmes plus facilement (consultez l'Annexe A, *Messages d'erreur*, page 893).

Vous devez afficher votre fichier SWF dans une version spéciale de Flash Player appelée débogueur de Flash Player. Lorsque vous installez l'outil de programmation, le débogueur de Flash Player est automatiquement installé. Ainsi, si vous installez Flash et naviguez sur un site web avec du contenu Flash, ou effectuez un test d'animation, cela signifie que vous utilisez le débogueur de Flash Player. Vous pouvez également utiliser le programme d'installation situé dans le répertoire <app_dir>\Players\Debug\, ou lancer le débogueur de Flash Player autonome à partir du même répertoire.

Lorsque vous utilisez la commande Tester l'animation pour tester des animations qui implémentent des contrôles de clavier (tabulation, raccourcis claviers créés à l'aide de `Key.addListener()`, etc.), choisissez Contrôle > Désactiver les raccourcis clavier. La sélection de cette option empêche l'environnement auteur de « récupérer » les séquences de touches, et les laisse être transmises au lecteur. Par exemple, dans l'environnement auteur, Ctrl+U ouvre la boîte de dialogue Préférences. Si votre script affecte Ctrl+U à une action qui souligne du texte à l'écran, lorsque vous utilisez Tester l'animation, le fait d'appuyer sur Ctrl+U ouvre la boîte de dialogue Préférences au lieu d'exécuter l'action qui souligne le texte. Pour que la commande Ctrl+U puisse être transmise au lecteur, vous devez choisir Contrôle > Désactiver les raccourcis clavier.

Attention : La commande Tester l'animation échoue si n'importe quelle partie du chemin du fichier SWF contient des caractères ne pouvant pas être représentés à l'aide du système de codage MBCS. Par exemple, les chemins japonais ne fonctionnent pas dans un système anglais. Toutes les zones de l'application utilisant le lecteur externe sont soumises à cette restriction.

Le débogueur affiche une liste hiérarchique des clips actuellement chargés dans Flash Player. Il permet d'afficher et de modifier la valeur des variables et des propriétés pendant la lecture du fichier SWF et d'insérer des points d'arrêt grâce auxquels vous pouvez arrêter le fichier SWF et examiner le code ActionScript ligne par ligne.

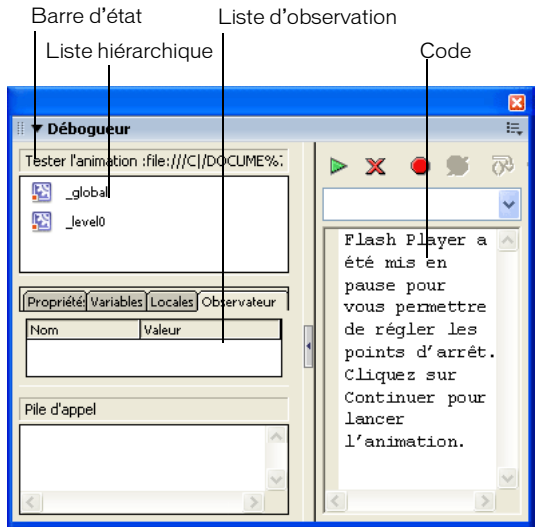
Vous pouvez utiliser le débogueur en mode de test sur des fichiers locaux, ou l'utiliser pour tester des fichiers sur un serveur web à un emplacement distant. Le débogueur permet d'insérer des points d'arrêt dans le code ActionScript, grâce auxquels vous pouvez arrêter l'animation et consulter le code ligne par ligne. Vous pouvez alors revenir aux scripts et les modifier afin d'obtenir les résultats souhaités.

Lorsque le débogueur est activé, sa barre d'état affiche l'URL ou le chemin d'accès local du fichier, indique si celui-ci est exécuté en mode de test ou depuis un site distant, et présente une vue en direct de la liste hiérarchique de clips. Lorsque vous ajoutez des clips au fichier ou en supprimez, la liste est immédiatement mise à jour. Vous pouvez redimensionner la liste hiérarchique en tirant la barre de séparation horizontale.

Pour activer le débogueur :

- Choisissez Contrôle > Déboguer l'animation.

  Cette commande entraîne l'ouverture du débogueur. Elle ouvre également le fichier SWF en mode de test.



Débogage d'un fichier SWF à partir d'un emplacement distant

Vous pouvez déboguer un fichier SWF à distance au moyen de la version autonome, de la version ActiveX ou du module de Flash Player. Lors de l'exportation d'un fichier SWF, vous pouvez activer le débogage dans le fichier et créer un mot de passe de débogage. Le débogueur n'est pas activé si vous n'activez pas le débogage.

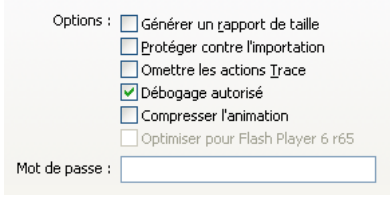
Pour que seuls certains utilisateurs puissent exécuter vos fichiers SWF dans le débogueur de Flash Player, vous pouvez les publier avec un mot de passe de débogage. Tout comme dans JavaScript ou HTML, les utilisateurs peuvent consulter les variables côté client dans ActionScript. Pour stocker les variables de façon sécurisée, vous devez les envoyer à une application côté serveur au lieu de les stocker dans votre fichier. Cependant, en tant que développeur Flash, vous ne voudrez peut-être pas révéler d'autres secrets professionnels, tels que des structures de clips. Vous pouvez donc utiliser un mot de passe de débogage pour protéger votre travail.

Lorsque vous exportez, publiez ou testez une animation, Flash crée un fichier SWD contenant des informations de débogage. Pour effectuer le débogage à distance, vous devez placer le fichier SWD dans le répertoire du serveur qui contient le fichier SWF.

Pour permettre le débogage à distance d'une animation Flash :

1 Choisissez Fichier > Paramètres de publication.

2 Sous l'onglet Flash de la boîte de dialogue Paramètres de publication, activez l'option Débogage autorisé.



3 Pour définir un mot de passe, entrez-le dans le champ Mot de passe.

Une fois un mot de passe défini, personne ne peut télécharger d'informations vers le débogueur sans entrer ce mot de passe. Cependant, aucun mot de passe n'est requis si vous ne remplissez pas le champ correspondant.

4 Fermez la boîte de dialogue Paramètres de publication, puis choisissez l'une des commandes suivantes :

- Contrôle > Déboguer l'animation
- Fichier > Exporter l'animation
- Fichier > Paramètres de publication > Publier

Flash crée un fichier de débogage portant l'extension .swd et l'enregistre conjointement avec le fichier SWF. Le fichier SWD contient des informations qui vous permettent d'utiliser des points d'arrêt et de faire défiler le code pas à pas.

5 Placez le fichier SWD dans le répertoire du serveur qui contient le fichier SWF.

Si le fichier SWD ne se trouve pas dans ce répertoire, vous pouvez toujours effectuer le débogage à distance, mais le débogueur ignore les points d'arrêt et il vous est impossible de faire défiler le code pas à pas.

6 Dans Flash, choisissez Fenêtre > Panneaux de développement > Débogueur.

- Dans le débogueur, dans le menu contextuel Options (dans le coin supérieur droit du panneau), choisissez Activer le débogage à distance.

Pour activer le débogueur à distance :

1 Ouvrez l'application auteur Flash.

2 Dans un navigateur ou dans le lecteur autonome, ouvrez le fichier SWF publié à partir de son emplacement distant.

La boîte de dialogue Débogage à distance apparaît.

Si cette boîte de dialogue ne s'affiche pas, cela signifie que Flash n'a pas trouvé le fichier SWD. Dans ce cas, cliquez sur le fichier SWF avec le bouton droit de la souris (Windows) ou en appuyant sur la touche Contrôle (Macintosh) pour afficher un menu contextuel, et choisissez Débogueur.

3 Dans la boîte de dialogue Débogage à distance, sélectionnez Hôte local ou Autre machine :

- L'option Hôte local est utilisée si le lecteur de débogage et l'application auteur Flash se trouvent sur le même ordinateur.
- Optez pour Autre machine si le lecteur de débogage et l'application auteur Flash ne se trouvent pas sur le même ordinateur. Entrez l'adresse IP de l'ordinateur qui exécute l'application auteur Flash.

4 Lorsqu'une connexion est établie, une boîte de dialogue s'affiche et vous invite à entrer un mot de passe. Entrez le mot de passe de débogage si vous en avez défini un.

La liste hiérarchique du fichier SWF apparaît dans le débogueur.

Affichage et modification de variables

L'onglet Variables du débogueur affiche les noms et valeurs des variables globales et de scénario du fichier SWF. Si vous modifiez la valeur d'une variable dans cet onglet, vous pouvez constater la modification dans le fichier SWF au cours de son exécution. Par exemple, pour tester la détection de collision dans un jeu, vous pouvez entrer la valeur de la variable afin de positionner une balle à l'emplacement correct près d'un mur.

L'onglet Locales du débogueur affiche les noms et les valeurs des variables locales disponibles lorsque le fichier SWF s'est arrêté à un point d'arrêt ou n'importe où ailleurs dans une fonction définie par l'utilisateur.

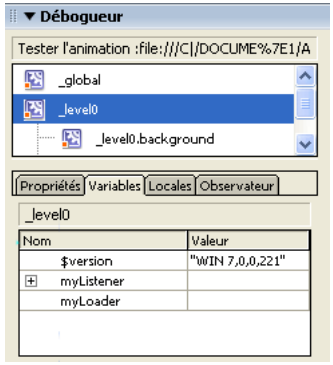
Pour afficher une variable :

1 Sélectionnez le clip contenant la variable de la liste.

Pour afficher les variables globales, sélectionnez le clip `_global` dans la liste.

2 Cliquez sur l'onglet Variables.

La liste hiérarchique est automatiquement mise à jour au cours de la lecture du fichier SWF. Un clip supprimé du fichier SWF au niveau d'une image spécifique est également supprimé (avec sa variable et son nom de variable) de la liste hiérarchique du débogueur. Toutefois, si vous marquez une variable pour la liste d'observation (consultez *Utilisation de la liste d'observation*, page 78), cette variable n'est pas supprimée.



Pour modifier la valeur d'une variable :

- Double-cliquez sur la valeur et entrez-en une nouvelle.

La valeur ne peut pas être une expression. Par exemple, vous pouvez utiliser `"Bonjour"`, `3523` ou `"http://www.macromedia.com"`, et vous ne pouvez pas utiliser `x + 2` ou `eval("nom:" +i)`. La valeur peut être une chaîne (n'importe quelle valeur comprise entre guillemets, un nombre ou une valeur booléenne (`true` ou `false`).

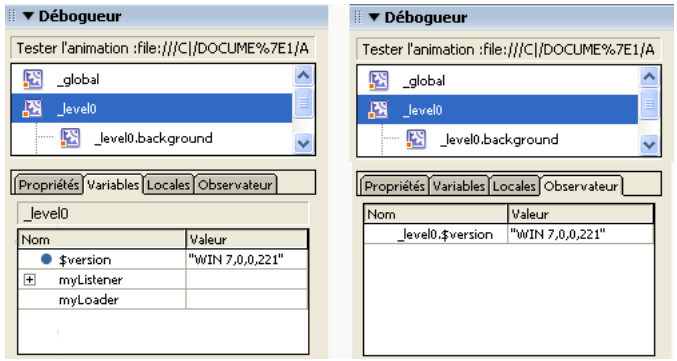
Remarque : Pour afficher la valeur d'une expression dans le panneau de sortie en mode de test d'animation, utilisez l'instruction `trace`. Pour plus d'informations, consultez *Utilisation de l'instruction trace*, page 85.

Utilisation de la liste d'observation

Pour facilement contrôler un ensemble de variables critiques, vous pouvez marquer celles qui doivent apparaître dans la liste d'observation. Cette liste affiche le chemin d'accès absolu de la variable et de la valeur. Vous pouvez également entrer une nouvelle valeur de variable dans la liste d'observation, comme dans le volet Variables.

Si vous ajoutez une variable locale à la liste d'observation, sa valeur ne s'affiche que lorsque le lecteur est arrêté sur une ligne de code ActionScript au niveau de laquelle cette variable est utilisée. Toutes les autres variables s'affichent pendant la lecture du fichier SWF. Si le débogueur ne trouve pas la valeur de la variable, elle est répertoriée comme Undefined.

La liste d'observation ne peut afficher que les variables, et non les propriétés ou les fonctions.



Variables marquées pour la liste d'observation et variables dans la liste d'observation

Pour ajouter des variables à la liste d'observation, effectuez l'une des opérations suivantes :

- Sous l'onglet Variables ou Locales, cliquez sur une variable sélectionnée avec le bouton droit de la souris (Windows) ou en appuyant sur la touche Contrôle (Macintosh), puis choisissez Observateur dans le menu contextuel. Un point bleu apparaît en regard de la variable.
- Sous l'onglet Observateur, cliquez sur une variable sélectionnée avec le bouton droit de la souris (Windows) ou en appuyant sur la touche Contrôle (Macintosh), puis choisissez Ajouter dans le menu contextuel. Entrez le chemin cible du nom de variable et sa valeur dans les champs.

Pour retirer des variables de la liste d'observation :

- Sous l'onglet Observateur, cliquez sur une variable sélectionnée avec le bouton droit de la souris (Windows) ou en appuyant sur la touche Contrôle (Macintosh), puis choisissez Supprimer dans le menu contextuel.

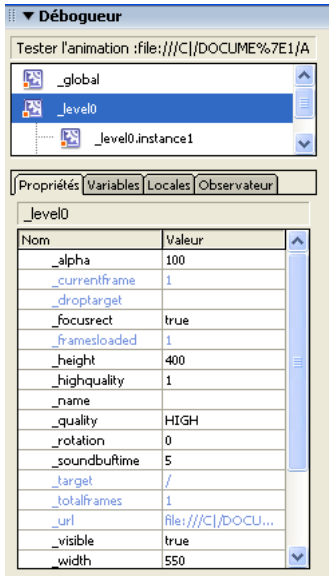
Affichage et modification des propriétés de clip

L'onglet Propriétés du débogueur affiche toutes les valeurs de propriétés des clips sur la scène. Vous pouvez modifier une valeur et constater l'effet de la modification sur le fichier SWF lors de son exécution. Certaines propriétés de clip sont en lecture seule et ne peuvent pas être modifiées.

Pour afficher les propriétés d'un clip dans le débogueur :

1 Sélectionnez un clip dans la liste.

2 Cliquez sur l'onglet Propriétés du débogueur.



Pour modifier la valeur d'une propriété :

- Double-cliquez sur la valeur et entrez-en une nouvelle.

La valeur ne peut pas être une expression. Par exemple, vous pouvez entrer `50` ou `"goutteDeau"` mais pas `x + 50`. La valeur peut être une chaîne (n'importe quelle valeur comprise entre guillemets, un nombre ou une valeur booléenne (`true` ou `false`). Vous ne pouvez pas entrer de valeurs d'objets ou tableaux (par exemple, `{id: "rogue"}` ou `[1, 2, 3]`) dans le débogueur.

Pour plus d'informations, consultez *Opérateurs de chaîne*, page 49 et *Utilisation d'opérateurs pour manipuler les valeurs des expressions*, page 48.

Remarque : Pour afficher la valeur d'une expression dans le panneau de sortie en mode de test d'animation, utilisez l'instruction `trace`. Pour plus d'informations, consultez *Utilisation de l'instruction trace*, page 85.

Définition et suppression de points d'arrêt

Un point d'arrêt vous permet d'interrompre un fichier SWF en cours de lecture dans Flash Player à une ligne précise de code ActionScript. Vous pouvez utiliser les points d'arrêt pour tester d'éventuels endroits problématiques du code. Par exemple, si vous avez rédigé un jeu d'instructions `if..else if` et que vous ne pouvez pas déterminer laquelle est en cours d'exécution, vous pouvez ajouter un point d'arrêt avant ces instructions et les faire défiler une par une dans le débogueur.

Vous pouvez définir des points d'arrêt dans le panneau Actions ou dans le débogueur. (Pour définir des points d'arrêt dans des scripts externes, vous devez utiliser le débogueur.) Les points d'arrêt définis dans le panneau Actions sont enregistrés dans le document Flash (fichier FLA). Les points d'arrêt définis dans le débogueur ne sont pas enregistrés dans le fichier FLA et ne sont valides que pour la session de débogage en cours.

Pour définir ou supprimer un point d'arrêt dans le panneau Actions, effectuez l'une des opérations suivantes :

- Cliquez dans la marge gauche. La présence d'un point rouge indique un point d'arrêt.
- Cliquez sur le bouton Options de débogage, situé au-dessus de la fenêtre de script.
- Cliquez avec le bouton droit (Windows) ou en appuyant sur la touche Contrôle (Macintosh) pour afficher le menu contextuel et choisissez Définir un point d'arrêt, Supprimer le point d'arrêt ou Supprimer tous les points d'arrêt.
- Appuyez sur Ctrl+Maj+B (Windows) ou sur Commande+Maj+B (Macintosh).

Remarque : Dans les versions précédentes de Flash, un clic dans la marge gauche de la fenêtre de script sélectionnait la ligne de code. Désormais il permet d'ajouter ou de supprimer un point d'arrêt. Pour sélectionner une ligne de code, cliquez en appuyant sur la touche Ctrl (Windows) ou Commande (Macintosh).

Pour définir ou supprimer des points d'arrêt dans le débogueur, effectuez l'une des opérations suivantes :

- Cliquez dans la marge gauche. La présence d'un point rouge indique un point d'arrêt.
- Cliquez sur le bouton Basculer le point d'arrêt ou Supprimer tous les points d'arrêt, au-dessus de l'affichage du code.
- Cliquez avec le bouton droit (Windows) ou en appuyant sur la touche Contrôle (Macintosh) pour afficher le menu contextuel et choisissez Définir un point d'arrêt, Supprimer le point d'arrêt ou Supprimer tous les points d'arrêt.
- Appuyez sur Ctrl+Maj+B (Windows) ou sur Commande+Maj+B (Macintosh).

  Lorsque Flash Player est interrompu à un point d'arrêt, vous pouvez entrer dans une ligne de code, l'ignorer ou en sortir. Un point d'arrêt fixé dans un commentaire ou sur une ligne vide du panneau Actions est ignoré.

Défilement de lignes de code

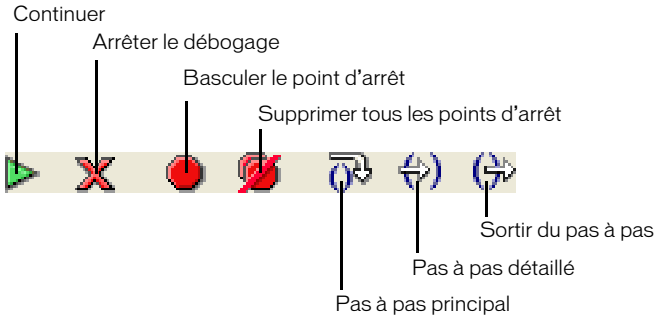
Flash Player est mis en pause lorsque vous commencez une session de débogage. Si vous définissez des points d'arrêt dans le panneau Actions, vous pouvez tout simplement cliquer sur le bouton Continuer pour lire le fichier SWF jusqu'à ce qu'il rencontre un de ces points. Par exemple, dans le code suivant, supposons qu'un point d'arrêt a été défini dans un bouton sur la ligne `maFonction();`:

```
on(press) {
  maFonction();
}
```

Lorsque vous cliquez sur le bouton, le point d'arrêt est atteint et Flash Player s'interrompt. Vous pouvez ensuite intervenir afin d'amener le débogueur à la première ligne de `maFonction()`, là où elle a été définie dans le document. Vous pouvez également faire défiler toute la fonction ou la quitter.

Si vous n'avez pas défini de points d'arrêt dans le panneau Actions, vous pouvez utiliser le menu de passage aux autres éléments du débogueur pour sélectionner n'importe quel script de l'animation. Vous pouvez ensuite y ajouter des points d'arrêt. Après avoir ajouté des points d'arrêt, vous devez cliquer sur le bouton Continuer pour lancer l'animation. Le débogueur s'arrête lorsqu'il atteint le point d'arrêt.

Lorsque vous faites défiler les lignes de code, les valeurs des variables et des propriétés changent dans la liste d'observation ainsi que dans les onglets Variables, Locales et Propriétés. Une flèche jaune sur le côté gauche de la fenêtre de code du débogueur indique la ligne sur laquelle le débogueur s'est arrêté. Utilisez les boutons suivants, placés en haut de la fenêtre de code :



Le bouton **Pas à pas détaillé** fait avancer le débogueur (indiqué par la flèche jaune) dans une fonction. Le bouton Pas à pas détaillé ne peut être utilisé qu'avec les fonctions définies par l'utilisateur.

Dans l'exemple suivant, si vous placez un point d'arrêt à la ligne 7 et que vous cliquez sur Pas à pas détaillé, le débogueur passe à la ligne 2, et un nouveau clic sur ce bouton passe à la ligne 3. Un clic sur Pas à pas détaillé pour des lignes ne comportant pas de fonction définie par l'utilisateur fait avancer le débogueur sur une ligne de code. Par exemple, si vous effectuez un arrêt à la ligne 2 et que vous choisissez Pas à pas détaillé, le débogueur passe à la ligne 3, comme dans l'exemple suivant :

```
1 function maFonction() {
2 x = 0;
3 y = 0;
4 }
5
6 mover = 1;
7 maFonction();
8 mover = 0;
```

Le bouton **Sortir du pas à pas** fait sortir le débogueur d'une fonction. Le bouton Sortir du pas à pas ne fonctionne que si vous êtes actuellement arrêté(e) sur une fonction définie par l'utilisateur. Il déplace la flèche jaune sur la ligne suivant celle au niveau de laquelle cette fonction a été appelée. Dans l'exemple ci-dessus, si vous placez un point d'arrêt à la ligne 3 et que vous cliquez sur Sortir du pas à pas, le débogueur passe à la ligne 8. Un clic sur Sortir du pas à pas sur une ligne qui ne se trouve pas dans une fonction définie par l'utilisateur équivaut à cliquer sur Continuer. Par exemple, si vous vous arrêtez à la ligne 6 et que vous cliquez sur Sortir du pas à pas, le lecteur continue à exécuter le script jusqu'à ce qu'il rencontre un point d'arrêt.

Le bouton **Pas à pas principal** fait avancer le débogueur sur une ligne de code. Le bouton Pas à pas principal déplace la flèche jaune sur la ligne suivante du script et ignore les fonctions définies par l'utilisateur. Dans l'exemple ci-dessus, si vous êtes arrêté(e) à la ligne 7 et que vous cliquez sur Pas à pas principal, vous passez directement à la ligne 8 et `maFonction()` est ignorée.

Le bouton **Continuer** quitte la ligne au niveau de laquelle le lecteur est arrêté et continue la lecture jusqu'à ce qu'un point d'arrêt soit atteint.

Le bouton **Arrêter le débogage** désactive le débogueur sans arrêter la lecture du fichier SWF dans Flash Player.

Utilisation du panneau de sortie

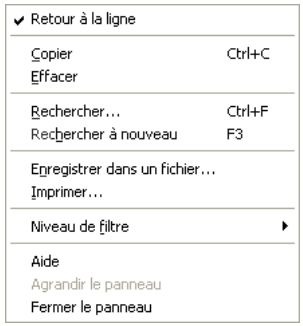
En mode de test d'animation, le panneau de sortie affiche des informations facilitant le dépannage de votre fichier SWF. Certaines informations, telles que les erreurs de syntaxe, sont automatiquement affichées. Vous pouvez afficher d'autres informations à l'aide des commandes Lister les objets et Lister les variables. (Consultez *Liste des objets d'un fichier SWF*, page 83 et *Liste des variables d'un fichier SWF*, page 84.)

Si vous utilisez l'instruction `trace` dans vos scripts, vous pouvez envoyer des informations spécifiques au panneau de sortie au cours de l'exécution du fichier SWF. Il peut s'agir de notes relatives à l'état du fichier SWF ou à la valeur d'une expression. Pour plus d'informations, consultez *Utilisation de l'instruction trace*, page 85.

Pour afficher le panneau de sortie, choisissez Fenêtre > Panneaux de développement > Sortie ou appuyez sur la touche F2.

Remarque : Si un script comporte des erreurs de syntaxe, le panneau de sortie apparaît automatiquement lorsque vous vérifiez la syntaxe ou que vous testez votre fichier SWF.

Pour utiliser le contenu du panneau de sortie, utilisez le menu d'options (dans le coin supérieur droit).



Liste des objets d'un fichier SWF

En mode de test d'animation, la commande Lister les objets affiche le niveau, l'image, le type d'objet (forme, clip ou bouton), les chemins cible et les noms d'occurrences de clips, boutons et champs de texte dans une liste hiérarchique. Cela est particulièrement utile pour rechercher le chemin cible et le nom de l'occurrence corrects. Contrairement au débogueur, la liste n'est pas automatiquement mise à jour au cours de la lecture du fichier SWF ; vous devez choisir la commande Lister les objets chaque fois que vous voulez envoyer les informations au panneau de sortie.

La commande Lister les objets ne dresse pas la liste de tous les objets de données ActionScript. Dans ce contexte, un objet est considéré comme une forme ou un symbole sur la scène.

Pour afficher une liste d'objets d'une animation :

1 Si l'animation n'est pas en cours d'exécution en mode de test d'animation, choisissez Contrôle > Tester l'animation.

2 Choisissez Déboguer > Lister les objets.

La liste des objets actuellement sur la scène s'affiche dans le panneau de sortie, comme illustré dans l'exemple suivant :

```
Level #0: Image=1 Etiquette="Scene_1"
  Bouton: Cible="_level0.monBouton"
    Forme:
  Clip: Image=1 Cible="_level0.monClip"
    Forme:
  Modifier le texte: Cible="_level0.monChampDeTexte" Texte="Ceci est un
  exemple."
```

Liste des variables d'un fichier SWF

En mode de test, la commande Lister les variables affiche la liste de toutes les variables actuelles que contient le fichier SWF. Cela est particulièrement utile pour rechercher la variable cible et le nom de la variable corrects. Contrairement au débogueur, la liste n'est pas automatiquement mise à jour au cours de la lecture du fichier SWF ; vous devez choisir la commande Lister les variables chaque fois que vous voulez envoyer les informations au panneau de sortie.

La commande Lister les variables affiche également les variables globales déclarées au moyen de l'identifiant `_global`. Les variables globales sont affichées en haut de la liste produite par l'option Lister les variables, dans une section nommée « Variables globales ». Chacune d'entre elles est précédée de la mention `_global`.

En outre, la commande Lister les variables affiche des propriétés de type lecture/définition, des propriétés créées par la méthode `Objet.addProperty()` et qui appellent des méthodes `get` ou `set`. Une propriété lecture/définition est affichée à côté des autres propriétés de l'objet auquel elle appartient. Pour distinguer aisément ces propriétés des variables ordinaires, la valeur d'une propriété lecture/définition porte le préfixe `[obtenir/placer]`. La valeur affichée pour une propriété lecture/définition est déterminée par l'évaluation de la fonction `get` de la propriété.

Pour afficher la liste des variables d'un fichier SWF :

1. Si le fichier SWF n'est pas en cours d'exécution en mode de test, choisissez Contrôle > Tester l'animation.
2. Choisissez Déboguer > Lister les variables.

   La liste de toutes les variables qui figurent actuellement dans le fichier SWF s'affiche dans le panneau de sortie, comme illustré dans l'exemple suivant :

```
Variables globales :
  Variable _global.MonTableauDeGlobales = [objet 1] [
    0:1,
    1:2,
    2:3
  ]
Level #0:
  Variable _level0.$version = "WIN 6,0,0,101"
  Variable _level0.VariableNormale = "Gary"
  Variable _level0.UnObjet = [objet 1] {
    maPropriété: [obtenir/placer] 3.14159
  }
```

Affichage des propriétés de champ de texte pour le débogage

Pour obtenir des informations de débogage sur les objets du champ de texte, vous pouvez utiliser la commande Déboguer > Lister les variables en mode tester l'animation. Le panneau de sortie utilise les conventions suivantes pour l'affichage des objets TextField :

- Si une propriété est introuvable sur l'objet, elle ne s'affiche pas.
- Pas plus de quatre propriétés s'affichent sur une ligne.
- Une propriété possédant une valeur de chaîne est affichée sur une ligne séparée.
- Si d'autres propriétés sont définies pour l'objet après le traitement des propriétés intégrées, elles s'ajoutent à l'affichage selon les règles des deuxième et troisième points précédents.
- Les propriétés de couleur s'affichent sous forme de nombres hexadécimaux (0x00FF00).

- Les propriétés sont affichées dans l'ordre suivant : `variable`, `text`, `htmlText`, `html`, `textWidth`, `textHeight`, `maxChars`, `borderColor`, `backgroundColor`, `textColor`, `border`, `background`, `wordWrap`, `password`, `multiline`, `selectable`, `scroll`, `hscroll`, `maxscroll`, `maxhscroll`, `bottomScroll`, `type`, `embedFonts`, `restrict`, `length`, `tabIndex`, `autoSize`.

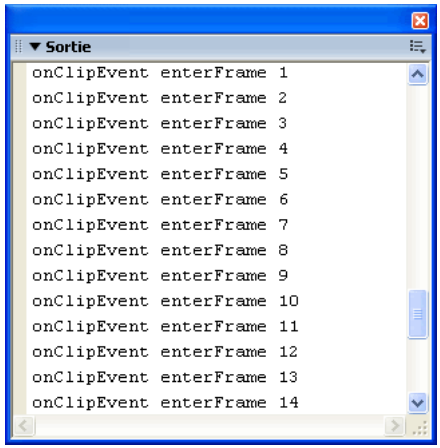
La commande Déboguer > Lister les objets en mode test répertorie les objets du champ de texte. Si un nom d'occurrence est spécifié pour un champ de texte, le panneau de sortie affiche le chemin cible complet, y compris le nom d'occurrence sous la forme suivante :

```
Target = "target path"
```

Utilisation de l'instruction trace

L'utilisation de l'instruction `trace` dans un script vous permet d'envoyer des informations au panneau de sortie. Par exemple, lors du test d'une animation ou séquence, vous pouvez envoyer des notes de programmation spécifiques au panneau ou afficher des résultats spécifiques lorsque vous cliquez sur un bouton ou en lisez une image. L'instruction `trace` est similaire à l'instruction JavaScript `alert`.

Lorsque vous utilisez l'instruction `trace` dans un script, vous pouvez utiliser des expressions en tant que paramètres. La valeur d'une expression s'affiche dans le panneau de sortie en mode de test, comme illustré dans l'exemple suivant :



L'instruction `trace` renvoie des valeurs qui sont affichées dans le panneau de sortie.

```
onClipEvent (enterFrame) {
  trace("onClipEvent enterFrame " + enterFrame++)
}
```

Mise à jour de Flash Player pour le test

Vous pouvez télécharger la version la plus récente de Flash Player à partir du site du centre de support Macromedia à l'adresse www.macromedia.com/go/flash_support_fr afin de l'utiliser pour tester vos fichiers SWF.

PARTIE II
Gestion des événements et création d'interactivité

Certains événements, tels que les clics de souris ou les pressions exercées sur les touches, peuvent être générés par l'utilisateur. D'autres peuvent être le résultat de certains autres processus, tels que le chargement d'un fichier XML sur le réseau. Le premier chapitre de cette section décrit les différents types d'événements dans Macromedia Flash et explique comment les traiter dans ActionScript. Le deuxième chapitre explique comment appliquer ces principes pour créer des présentations, des applications et des animations interactives simples.

CHAPITRE 4
Gestion d'événements

Un *événement* est une occurrence logicielle ou matérielle qui requiert une réponse d'une application Macromedia Flash. Par exemple, un événement tel qu'un clic de souris ou une pression sur une touche est appelé *événement utilisateur*, puisqu'il résulte d'une interaction directe avec l'utilisateur. Un événement généré automatiquement par Flash Player, tel que l'apparence initiale d'un clip sur la scène, est appelé *événement système*, car il n'est pas généré directement par l'utilisateur.

Pour que votre application réagisse à des événements, vous devez utiliser des *gestionnaires d'événement* (code ActionScript associé à un objet et à un événement particuliers). Par exemple, lorsqu'un utilisateur clique sur un bouton sur la scène, la tête de lecture peut passer à l'image suivante. De même, à l'issue du chargement d'un fichier XML sur le réseau, le contenu du fichier peut s'afficher dans un champ de texte.

ActionScript propose différentes méthodes de gestion des événements : méthodes de gestionnaire d'événement, des écouteurs d'événement, et des gestionnaires d'événement de bouton et de clip.

Utilisation de méthodes de gestionnaire d'événement

Une méthode de gestionnaire d'événement est une méthode de classe invoquée lorsqu'un événement se produit sur une occurrence de cette classe. Par exemple, la classe Button définit un gestionnaire d'événement `onPress`, invoqué lorsque l'utilisateur clique avec le bouton de la souris sur un objet Button. Contrairement aux autres méthodes de classe, vous n'invoquez pas directement un gestionnaire d'événement ; Flash Player l'invoque automatiquement lorsque l'événement approprié se produit.

Par défaut, les méthodes de gestionnaire d'événement ne sont pas définies : lorsqu'un événement particulier se produit, son gestionnaire d'événement correspondant est invoqué, mais votre application ne répond pas davantage à l'événement. Pour qu'elle réponde à l'événement, définissez une fonction au moyen de l'instruction function et affectez-la au gestionnaire d'événement approprié. La fonction que vous avez affectée au gestionnaire d'événement est ensuite automatiquement invoquée lorsque l'événement se produit.

Un gestionnaire d'événement se compose de trois éléments : l'objet auquel l'événement s'applique, le nom de la méthode de gestionnaire d'événement de l'objet et la fonction que vous avez affectée au gestionnaire d'événement. L'exemple suivant illustre la structure de base d'un gestionnaire d'événement.

```
object.eventMethod = function () {
  // Votre code, répondant à l'événement
}
```

Par exemple, supposons que vous avez un bouton `suivant_btn` sur la scène. Le code suivant affecte au gestionnaire d'événement `onPress` du bouton une fonction qui permet de faire avancer la tête de lecture jusqu'à l'image suivante dans le scénario.

```
suivant_btn.onPress = function ()
  nextFrame();
}
```

Dans le code ci-dessus, la fonction `nextFrame()` est affectée directement à `onPress`. Vous pouvez également affecter une référence de fonction (nom) à une méthode de gestionnaire d'événement et définir la fonction ultérieurement.

```
// Assigne une référence de fonction à la méthode de gestionnaire d'événement
  onPress du bouton
suivant_btn.onPress = goNextFrame;

// Définit la fonction doSubmit()
function goNextFrame() {
  nextFrame();
}
```

Notez que vous affectez la référence de fonction et non la valeur renvoyée de la fonction au gestionnaire d'événement `onPress`.

```
// Incorrect !
suivant_btn.onPress = goNextFrame();
// Correct.
suivant_btn.onPress = goNextFrame;
```

Certains gestionnaires d'événement reçoivent des paramètres transmis qui donnent des informations sur l'événement qui s'est produit. Par exemple, le gestionnaire d'événement `TextField.onSetFocus` est invoqué lorsqu'une occurrence de champ de texte parvient au focus clavier. Ce gestionnaire d'événement reçoit une référence à l'objet de champ de texte qui avait précédemment le focus clavier.

Par exemple, le code suivant insère du texte dans le champ de texte qui vient juste de perdre le focus clavier.

```
nomDutilisateur_txt.onSetFocus = function(oldFocus_txt) {
  ancienFocus_txt.text = "Je viens de perdre le focus clavier";
}
```

Les classes ActionScript suivantes définissent des gestionnaires d'événement : Button, ContextMenu, ContextMenuItem, Key, LoadVars, LocalConnection, Mouse, MovieClip, MovieClipLoader, Selection, SharedObject, Sound, Stage, TextField, XML et XMLSocket. Pour plus d'informations sur les gestionnaires d'événement qu'elles proposent, consultez les entrées correspondantes dans le Chapitre 12, *Dictionnaire ActionScript*, page 215.

Vous pouvez également affecter des fonctions à des gestionnaires d'événement pour des objets que vous créez lors de l'exécution. Le code suivant, par exemple, crée une nouvelle occurrence de clip (`nouveauClip_mc`) et affecte une fonction au gestionnaire d'événement `onPress` du clip.

```
_root.attachMovie("idDeSymbole", "nouveauClip_mc", 10);
nouveauClip_mc.onPress = function () {
  trace("Vous m'avez pressé");
}
```

Pour plus d'informations, consultez *Création de clips à l'exécution*, page 132.

Utilisation des écouteurs d'événement

Les écouteurs d'événement permettent à un objet, appelé *objet d'écoute*, de recevoir des événements générés par un autre objet, appelé *objet diffuseur*. L'objet diffuseur enregistre l'objet d'écoute afin de recevoir des événements générés par le diffuseur. Par exemple, vous enregistrez un objet de clip pour recevoir des notifications `onResize` de la scène, ou une occurrence de bouton peut recevoir des notifications `onChanged` d'un objet de champ de texte. Vous pouvez enregistrer plusieurs objets d'écoute pour recevoir des événements d'un seul diffuseur, et vous pouvez enregistrer un seul objet d'écoute pour recevoir des événements de plusieurs diffuseurs.

Les écouteurs d'événement utilisent le même modèle que les gestionnaires d'événement (consultez *Utilisation de méthodes de gestionnaire d'événement*, page 89), à deux différences près :

- L'objet auquel vous affectez le gestionnaire d'événement n'est pas l'objet qui émet l'événement.
- Appelez une méthode spéciale de l'objet diffuseur, `addListener()`, qui enregistre l'objet d'écoute pour recevoir ses événements.

Pour utiliser les écouteurs d'événement, créez un objet d'écoute avec une propriété qui porte le même nom que l'événement en cours de création par l'objet diffuseur. Affectez ensuite une fonction à l'écouteur d'événement qui répond en quelque sorte à l'événement. Enfin, appelez `addListener()` sur l'objet qui diffuse l'événement, en lui transmettant le nom de l'objet d'écoute. Le code suivant présente le modèle d'écouteur d'événement.

```
objetDecoute.eventName = function(){
  // votre code
};
objetDiffuseur.addListener(listenerObject);
```

L'objet d'écoute spécifié peut être tout objet, tel qu'une occurrence de bouton ou de clip sur la scène ou une occurrence de n'importe quelle classe ActionScript. Le nom de l'événement est un événement qui se produit sur *`objetDiffuseur`*, qui ensuite diffuse l'événement à *`objetDecoute`*. Vous pouvez enregistrer plusieurs écouteurs sur un diffuseur d'événement.

L'exemple suivant illustre comment utiliser l'écouteur d'événement `Selection.onSetFocus` pour créer un gestionnaire de focus simple pour un groupe de champs de saisie de texte. Dans ce cas, la bordure du champ de texte qui reçoit le focus clavier est activée (affichée) et la bordure du champ de texte qui a perdu le focus est désactivée.

Pour créer un gestionnaire de focus simple avec des gestionnaires d'événement :

1. En utilisant l'outil Texte, créez un champ de texte sur la scène.
2. Sélectionnez le champ de texte et, dans l'Inspecteur des propriétés, choisissez Entrée dans le menu contextuel Type de texte, puis choisissez l'option Afficher la bordure autour du texte.
3. Créez un autre champ de saisie de texte, sous le premier.

Assurez-vous que l'option Afficher la bordure autour du texte n'est pas activée pour ce champ de texte. Si nécessaire, continuez à créer des champs de saisie de texte.

4 Sélectionnez l'image 1 dans le scénario et ouvrez le panneau Actions (Fenêtre > Panneaux de développement > Actions).

5 Pour créer un objet qui écoute les notifications de focus d'une classe Selection, entrez le code suivant dans le panneau Actions :

```
var ecouteurDeFocus = new Object();
ecouteurDeFocus.onSetFocus = function(ancienFocus_txt, nouveauFocus_txt) {
  ancienFocus_txt.border = false;
  nouveauFocus_txt.border = true;
}
```

Ce code crée un nouvel objet ActionScript (générique) appelé `focusListener`. Cet objet définit pour lui-même une propriété `onSetFocus`, à laquelle il affecte une fonction. La fonction prend en compte deux paramètres : une référence au champ de texte qui a perdu le focus et une référence au champ de texte qui a reçu le focus. La fonction définit la propriété `border` du champ de texte qui a perdu le focus à `false` et la propriété `border` du champ de texte qui a reçu le focus à `true`.

6 Pour enregistrer l'objet `focusListener` pour recevoir des événements de l'objet Selection, ajoutez le code suivant au panneau Actions :

```
Selection.addListener(focusListener);
```

7 Testez l'animation (Contrôle > Tester l'animation), cliquez dans le premier champ de texte, puis appuyez sur Tab pour passer d'un champ à l'autre.

Pour désenregistrer un objet d'écoute de sorte qu'il ne reçoive plus d'événements, appelez la méthode `removeListener()` de l'objet diffuseur, en lui transmettant le nom de l'objet d'écoute.

```
objetDiffuseur.removeListener(listenerObject);
```

Les écouteurs d'événement sont disponibles pour les objets des classes ActionScript suivantes : Key, Mouse, MovieClipLoader, Selection, TextField, et Stage. Pour obtenir une liste des écouteurs d'événement disponibles pour chaque classe, consultez les entrées de ces classes dans le Chapitre 12, *Dictionnaire ActionScript*, page 215.

Utilisation de gestionnaires d'événement de bouton et de clip

Vous pouvez associer des gestionnaires d'événement directement à une occurrence de bouton ou de clip au moyen des gestionnaires `onClipEvent()` et `on()`. Le gestionnaire `onClipEvent()` traite les événements de clips alors que le gestionnaire `on()` traite ceux des boutons. Vous pouvez également utiliser `on()` avec des clips pour créer des clips qui reçoivent des événements de bouton. Pour plus d'informations, consultez *Création de clips avec états de bouton*, page 94.

Pour utiliser le gestionnaire `on()` ou `onClipEvent()`, associez-le directement à une occurrence d'un bouton ou d'un clip sur la scène et spécifiez l'événement que vous souhaitez gérer pour cette occurrence. Par exemple, le gestionnaire d'événement `on()` suivant est exécuté lorsque l'utilisateur clique sur le bouton auquel il est associé.

```
on(press) {
  trace("Merci de m'avoir pressé.");
}
```

Vous pouvez spécifier deux événements, voire plus, pour chaque gestionnaire `on()`, séparés par des virgules. Le code ActionScript dans un gestionnaire est exécuté lorsque l'un des événements spécifiés par le gestionnaire se produit. Par exemple, le gestionnaire `on()` suivant, associé à un bouton, est exécuté lorsque le pointeur de la souris passe au-dessus puis en dehors du bouton.

```
on(rollOver, rollOut) {
  trace("Vous êtes passé au-dessus ou en dehors");
}
```

Vous pouvez associer plusieurs gestionnaires à un objet si vous souhaitez exécuter différents scripts lorsque différents événements se produisent. Par exemple, vous pouvez associer les gestionnaires `onClipEvent()` suivants à la même occurrence de clip. Le premier est exécuté au premier chargement du clip (ou lorsque celui-ci apparaît sur la scène), le second est exécuté lorsque le clip est purgé de la scène.

```
onClipEvent(load) {
  trace("chargé");
}
onClipEvent (unload) {
  trace("purgé");
}
```

Pour obtenir la liste complète des événements pris en charge par les gestionnaires d'événement `on()` et `onClipEvent()`, consultez `on()`, page 660 et `onClipEvent()`, page 661.

La gestion d'événement avec `on()` et `onClipEvent()` ne provoque pas de conflit avec la gestion d'événement via des méthodes de gestionnaire d'événement que vous définissez. Imaginons par exemple que votre fichier SWF comporte un bouton associé à un gestionnaire `on(press)` qui déclenche la lecture du fichier SWF. Ce même bouton possède également une méthode `onPress`, pour laquelle vous définissez une fonction qui indique une rotation à un objet sur la scène. Lorsque l'utilisateur clique sur le bouton, la lecture du fichier SWF commence et l'objet tourne sur lui-même. Selon vos préférences, vous pouvez utiliser `on()` et `onClipEvent()`, des méthodes de gestionnaire d'événement ou ces deux types de gestion d'événement. Toutefois, le domaine des variables et des objets dans les gestionnaires `on()` et `onClipEvent()` n'est pas le même que celui du gestionnaire d'événement et des écouteurs d'événement. Pour plus d'informations, consultez *Domaine du gestionnaire d'événement*, page 94.

Vous pouvez associer `onClipEvent()` et `on()` uniquement à des occurrences de clips qui ont été placées sur la scène au cours de la programmation. Il est impossible d'associer `onClipEvent()` ou `on()` à des occurrences de clips créées à l'exécution (avec la méthode `attachMovie`, par exemple). Pour associer des gestionnaires d'événement à des objets créés à l'exécution, utilisez des méthodes de gestionnaire d'événement ou des écouteurs d'événement. Pour plus d'informations, consultez *Utilisation de méthodes de gestionnaire d'événement*, page 89 et *Utilisation des écouteurs d'événement*, page 91.

Création de clips avec états de bouton

Lorsque vous associez un gestionnaire `on()` à un clip ou que vous affectez une fonction à l'un des gestionnaires d'événement souris MovieClip pour une occurrence de clip, le clip répond à des événements souris de la même façon qu'un bouton. Vous pouvez également créer des états de bouton automatiques (Haut, Dessus et Abaissé) dans un clip, en ajoutant les étiquettes d'image `_up`, `_over` et `_down` au scénario du clip.

Lorsque l'utilisateur déplace la souris au-dessus d'un clip ou qu'il clique sur ce dernier, la tête de lecture passe à l'image avec l'étiquette d'image appropriée. Pour désigner la zone active qu'utilise un clip, utilisez la propriété `hitArea` de la classe MovieClip.

Pour créer des états de bouton dans un clip :

1. Sélectionnez, dans le scénario d'un clip, une image à utiliser comme état de bouton (Haut, Dessus ou Abaissé).
2. Entrez une étiquette d'image dans l'inspecteur des propriétés (`_up`, `_over` ou `_down`).
3. Pour ajouter d'autres états de bouton, répétez les étapes 1 et 2.
4. Pour que votre clip réponde aux événements souris, effectuez l'une des opérations suivantes :
   - Associez un gestionnaire d'événement `on()` à l'occurrence du clip, comme indiqué dans *Utilisation de gestionnaires d'événement de bouton et de clip*, page 92.
   - Affectez une fonction à l'un des gestionnaires d'événement souris de l'objet du clip (`onPress`, `onRelease`, etc.), comme indiqué dans *Utilisation de méthodes de gestionnaire d'événement*, page 89.

Domaine du gestionnaire d'événement

Le domaine, ou contexte, des variables et commandes que vous déclarez et exécutez dans un gestionnaire d'événement varie selon le type de gestionnaire d'événement que vous utilisez : gestionnaires d'événement ou écouteurs d'événement, gestionnaires `on()` et `onClipEvent()`.

Les fonctions affectées aux méthodes de gestionnaire d'événement et aux écouteurs d'événement (comme toutes les fonctions ActionScript que vous rédigez) définissent un domaine de variable locale, contrairement aux gestionnaires `on()` et `onClipEvent()`.

Par exemple, soit les deux gestionnaires d'événement suivants : le premier est un gestionnaire d'événement `onPress` associé à un clip appelé `clip_mc`, le second est un gestionnaire `on()` associé à la même occurrence de clip.

```
// Associé au scénario de clip parent de clip_mc :
clip_mc.onPress = function () {
  var couleur; // variable de fonction locale
  couleur = "bleu";
}
// gestionnaire on() associé à clip_mc :
on(press) {
  var couleur; // aucun domaine de variable locale
  couleur = "bleu";
}
```

Bien que les deux gestionnaires d'événement contiennent le même code, les résultats sont différents. Dans le premier cas, la variable `couleur` est locale par rapport à la fonction définie pour `onPress`. Dans le second, dans la mesure où le gestionnaire `on()` ne définit pas de domaine de variable locale, la variable a un domaine limité au scénario du clip `clip_mc`.

Dans le cas des gestionnaires d'événement `on()` associés à des boutons, plutôt qu'à des clips, le domaine des variables (à l'instar des appels de fonction et de méthode) est limité au scénario qui contient l'occurrence de bouton.

Le gestionnaire d'événement `on()` suivant, par exemple, génère différents résultats selon qu'il est associé à un clip ou à un bouton. Dans le premier exemple, l'appel de la fonction `play()` lance la tête de lecture du scénario qui contient le bouton ; dans le second, l'appel de la fonction `play()` démarre le scénario du clip auquel le gestionnaire est associé.

```
// Associé à un bouton
on(press) {
  play(); // exécute le scénario parent
}
// Associé à un clip
on(press) {
  play(); // exécute le scénario du clip
}
```

Autrement dit, lorsqu'il est associé à un objet de bouton, l'appel de la méthode `play()` s'applique au scénario qui contient le bouton, c'est-à-dire au scénario parent du bouton. En revanche, lorsque ce même gestionnaire est associé à un objet de clip, la méthode `play()` s'applique au clip qui contient le gestionnaire.

Dans la définition d'une fonction de gestionnaire d'événement ou d'écouteur d'événement, la fonction `play()` s'appliquerait au scénario contenant la définition. Imaginons par exemple que la fonction de gestionnaire d'événement `MovieClip.onPress` suivante soit déclarée dans le scénario contenant l'instance de clip `monClip`.

```
// Fonction définie dans le scénario du clip :
monClip.onPress = function () {
  play(); // exécute le scénario contenant la définition de la fonction
}
```

Pour exécuter le clip qui définit le gestionnaire d'événement `onPress`, vous devez faire explicitement référence à ce clip à l'aide du mot-clé `this`, comme suit :

```
monClip.onPress = function () {
  this.play(); // exécute le scénario du clip qui définit le gestionnaire
  onPress
}
```

Domaine du mot-clé « this »

Le mot-clé `this` fait référence à l'objet dans le domaine en cours d'exécution. Selon le type de technique de gestionnaire d'événement que vous utilisez, `this` renvoie à différents objets.

Dans une fonction de gestionnaire d'événement ou d'écouteur d'événement, `this` fait référence à l'objet qui définit la méthode du gestionnaire d'événement ou de l'écouteur d'événement. Dans le code suivant, par exemple, `this` renvoie à `monClip`.

```
// gestionnaire d'événement onPress() associé à _level0.monClip :
monClip.onPress = function () {
  trace(this); // affiche "_level0.monClip"
}
```

Dans un gestionnaire `on()` associé à un clip, `this` renvoie au clip auquel le gestionnaire `on()` est associé.

```
// Associé au clip "monClip"
on(press) {
  trace(this); // affiche "_level0.monClip"
}
```

Dans un gestionnaire `on()` associé à un bouton, `this` renvoie au scénario qui contient le bouton.

```
// Associé à un bouton du scénario principal
on(press) {
  trace(this); // affiche "_level0"
}
```

CHAPITRE 5
Création d'interactivité avec ActionScript

Dans une animation simple, Macromedia Flash Player lit les scènes et les images du fichier SWF de façon séquentielle. Dans un fichier SWF interactif, votre public utilise le clavier et la souris pour atteindre les différentes parties du fichier SWF, pour déplacer des objets, entrer des informations dans des formulaires et effectuer bien d'autres opérations interactives.

Utilisez ActionScript pour créer des scripts qui indiquent à Flash Player l'action à exécuter lorsqu'un événement survient. Certains événements pouvant déclencher un script se produisent lorsque la tête de lecture atteint une image, qu'un clip est chargé ou purgé, ou que l'utilisateur clique sur un bouton ou appuie sur une touche.

Les scripts peuvent prendre la forme d'une commande unique (pour ordonner la fin de la lecture d'un fichier SWF, par exemple) ou d'une série de commandes et d'instructions (pour évaluer une condition avant d'effectuer une action, par exemple). De nombreuses commandes ActionScript sont simples et permettent de créer des contrôles de base pour un fichier SWF. D'autres actions exigent une certaine connaissance des langages de programmation et sont destinées à un développement avancé.

A propos des événements et de l'interaction

Chaque clic de souris ou pression sur une touche entraîne la génération d'un événement. Ces types d'événements portent généralement le nom d'*événements utilisateur*, car ils sont générés en réponse à une action de l'utilisateur final. Vous pouvez utiliser ActionScript pour répondre à ces événements ou les *gérer*. Par exemple, lorsqu'un utilisateur clique sur un bouton, il peut être intéressant d'envoyer la tête de lecture vers une autre image du fichier SWF ou de charger une nouvelle page dans le navigateur.

Dans un fichier SWF, les boutons, clips et champs de texte génèrent tous des événements auxquels vous pouvez répondre. ActionScript offre trois manières de gérer les événements : les méthodes des gestionnaires d'événement, les écouteurs d'événements et les gestionnaires `on()` et `onClipEvent()`. Pour plus d'informations sur les événements et les gestionnaires d'événement, consultez le Chapitre 4, *Gestion d'événements*, page 89.

Contrôle de la lecture d'un fichier SWF

Les fonctions ActionScript suivantes vous permettent de contrôler la tête de lecture dans le scénario et de charger une nouvelle page web dans une fenêtre de navigateur :

- Les fonctions `gotoAndPlay()` et `gotoAndStop()` envoient la tête de lecture vers une image ou une scène. Ce sont des fonctions générales que vous pouvez appeler à partir de tout script. Vous pouvez également utiliser les méthodes `MovieClip.gotoAndPlay()` et `MovieClip.gotoAndStop()` pour naviguer dans le scénario d'un objet clip spécifique.
- Les actions `play()` et `stop()` entraînent la lecture et l'arrêt des animations.
- L'action `getURL()` permet d'atteindre une URL différente.

Déplacement vers une image ou une scène

Pour atteindre une image ou une scène spécifique du fichier SWF, vous pouvez utiliser les fonctions globales `gotoAndPlay()` ou `gotoAndStop()` ou les méthodes équivalentes `gotoAndPlay()` et `gotoAndStop()` de la classe MovieClip. Chaque fonction ou méthode vous permet de spécifier l'image de la scène en cours à atteindre. Si le document contient plusieurs scènes, vous pouvez en plus spécifier celle qui est concernée.

L'exemple suivant utilise la fonction globale `gotoAndPlay()` dans le gestionnaire d'événement `onRelease` d'un objet bouton pour envoyer la tête de lecture du scénario contenant le bouton vers l'image 10.

```
jump_btn.onRelease = function () {
  gotoAndPlay(10);
}
```

Dans l'exemple suivant, la méthode `MovieClip.gotoAndStop()` envoie le scénario d'un clip nommé `categories_mc` vers l'image 10 puis s'arrête. Lorsque vous utilisez les méthodes MovieClip `gotoAndPlay()` et `gotoAndStop()`, vous devez indiquer l'occurrence à laquelle elles s'appliquent.

```
jump_btn.onPress = function () {
  categories_mc.gotoAndStop(10);
}
```

Lecture et arrêt de clips

Sauf indication contraire, une fois démarré, un fichier SWF exécute toutes les images du scénario. Vous pouvez arrêter ou démarrer un fichier SWF à l'aide des fonctions globales `play()` et `stop()` ou des méthodes MovieClip équivalentes. Ainsi, la fonction ou la méthode `stop()` vous permet d'arrêter un fichier SWF à la fin d'une scène avant de passer à la scène suivante. Une fois un fichier SWF arrêté, il doit être redémarré au moyen de `play()`.

Vous pouvez utiliser les fonctions `play()` et `stop()` ou les méthodes MovieClip pour contrôler le scénario principal ou le scénario de tout clip ou de tout fichier SWF chargé. Le clip que vous souhaitez contrôler doit posséder un nom d'occurrence et être présent dans le scénario.

Le gestionnaire `on(press)` suivant associé à un bouton lance la tête de lecture dans le fichier SWF ou le clip contenant l'objet bouton.

```
// Associé à une occurrence de bouton
on(press) {
  // Lit le scénario contenant le bouton
  play();
}
```

Ce même code donnerait un résultat différent s'il était associé à un clip plutôt qu'à un bouton. Lorsqu'elles sont associées à un objet bouton, les instructions faites dans un gestionnaire `on()` sont par défaut appliquées au scénario contenant le bouton. En revanche, lorsqu'elles sont associées à un clip, les instructions définies dans un gestionnaire `on()` sont appliquées au clip contenant le gestionnaire.

Le gestionnaire `on()` suivant, par exemple, arrête le scénario du clip auquel il est associé, et non le scénario contenant le clip.

```
on(press) {
  stop();
}
```

Il en va de même pour les gestionnaires `onClipEvent()` associés à des clips. Le code suivant, par exemple, arrête le scénario du clip contenant le gestionnaire `onClipEvent()` lors du premier chargement du clip ou de sa première apparition sur la scène.

```
onClipEvent(load) {
  stop();
}
```

Déplacement vers une URL différente

Pour ouvrir une page web dans une fenêtre de navigateur ou pour transmettre des données à une autre application sur une URL définie, utilisez la fonction globale `getURL()` ou la méthode `MovieClip.getURL()`. Vous pouvez, par exemple, lier un bouton à un nouveau site web ou envoyer des variables de scénario à un script CGI pour traitement comme s'il s'agissait d'un formulaire HTML. Vous pouvez également spécifier une fenêtre cible, exactement comme pour le ciblage d'une fenêtre à l'aide d'une balise d'ancrage HTML (`<a></a>`).

Par exemple, le code suivant ouvre la page d'accueil macromedia.com dans une fenêtre de navigateur vide lorsque l'utilisateur clique sur l'occurrence de bouton `homepage_btn`.

```
homepage_btn.onRelease = function () {
  getURL("http://www.macromedia.com", _blank);
}
```

Vous pouvez également envoyer des variables avec l'URL, en utilisant `GET` ou `POST`. Cette opération est utile si la page que vous chargez provient d'un serveur d'applications, telle une page ColdFusion Server (CFM), et attend la réception de variables. Supposons par exemple que vous souhaitiez charger une page CFM nommée addUser.cfm qui attend deux variables de formulaire, `name` et `age`. Pour ce faire, vous pouvez créer un clip nommé `variables_mc` qui définit ces deux variables de la façon suivante :

```
variables_mc.name = "Francois";
variables_mc.age = 32;
```

Le code suivant charge ensuite addUser.cfm dans une fenêtre de navigateur vide, puis transmet `variables_mc.name` et `variables_mc.age` à la page CFM dans l'en-tête `POST`.

```
variables_mc.getURL("addUser.cfm", "_blank", "POST");
```

Pour plus d'informations, consultez `getURL()`, page 434.

Création d'interactivité et d'effets visuels

Pour créer de l'interactivité et d'autres effets visuels, vous devez comprendre les techniques suivantes :

- *Création d'un pointeur de souris personnalisé*
- *Obtention de la position de la souris*
- *Capture des pressions sur les touches*
- *Définition des valeurs des couleurs*
- *Création de commandes audio*
- *Détection des collisions*
- *Création d'un outil de dessin de ligne simple*

Création d'un pointeur de souris personnalisé

Un pointeur de souris standard est la représentation visuelle par le système d'exploitation de la position de la souris de l'utilisateur. Si vous remplacez le pointeur de souris standard par un pointeur réalisé dans Flash, vous pouvez intégrer plus étroitement les mouvements de la souris de l'utilisateur dans le fichier SWF. L'exemple de cette section utilise un pointeur personnalisé qui ressemble à une grande flèche. Cependant, la puissance de cette fonctionnalité réside dans la possibilité de donner au pointeur n'importe quelle apparence (par exemple, un ballon de football qui doit franchir la ligne de but ou un rouleau de tissu que l'utilisateur tire au-dessus d'un canapé afin de modifier sa couleur).

Pour créer un pointeur personnalisé, vous devez créer son clip sur la scène. Ensuite, dans ActionScript, masquez le pointeur standard et associez le clip aux mouvements du curseur personnalisé. Pour masquer le pointeur standard, utilisez la méthode `Mouse.hide()` de la classe Mouse intégrée. Pour utiliser un clip comme pointeur personnalisé, utilisez `startDrag()` l'action.

Pour créer un pointeur personnalisé :

1. Créez un clip à utiliser comme pointeur et placez une occurrence du clip sur la scène.
2. Sélectionnez l'occurrence de clip sur la scène.
3. Sélectionnez Fenêtre > Panneaux de développement > Actions pour ouvrir le panneau Actions si ce dernier n'est pas déjà visible.
4. Tapez ce qui suit dans le panneau Actions :

```
onClipEvent(load){
  Mouse.hide();
  startDrag(this, true);
}
onClipEvent(mouseMove) {
  updateAfterEvent();
}
```

Le premier gestionnaire `onClipEvent()` masque le pointeur de la souris lors de la première apparition du clip sur la scène. Le second gestionnaire appelle `updateAfterEvent()` chaque fois que l'utilisateur déplace la souris.

La fonction `updateAfterEvent()` actualise l'écran dès que l'événement spécifié s'est produit, sans attendre le dessin de l'image suivante, ce qui correspond au comportement par défaut. Pour plus d'informations, consultez `updateAfterEvent()`, page 850.

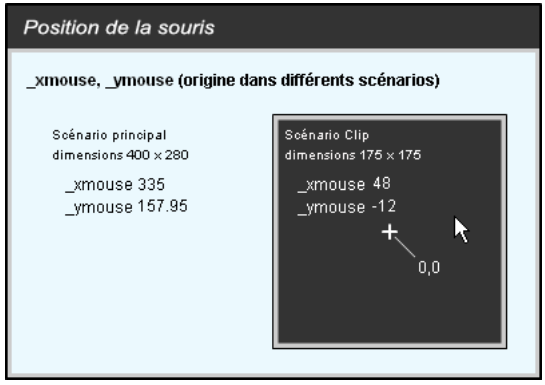
5 Sélectionnez Contrôle > Tester l'animation pour tester votre pointeur personnalisé.

Les boutons fonctionnent encore lorsque vous utilisez un pointeur personnalisé. Il est conseillé de placer le pointeur personnalisé sur le calque supérieur du scénario, de sorte qu'il s'affiche au premier plan, par-dessus les boutons et autres objets, lorsque vous le déplacez dans le fichier SWF. De plus, l'extrémité d'un pointeur personnalisé est le point d'alignement du clip que vous utilisez comme pointeur. Pour qu'elle corresponde à une partie spécifique du clip, définissez donc les coordonnées du point d'alignement de façon à ce qu'ils coïncident avec ce point.

Pour plus d'informations sur les méthodes de la classe Mouse, consultez l'entrée *Classe Mouse* dans le Chapitre 12, *Dictionnaire ActionScript*, page 215.

Obtention de la position de la souris

Vous pouvez utiliser les propriétés `_xmouse` et `_ymouse` pour déterminer la position du pointeur de la souris (curseur) dans un fichier SWF. Chaque scénario possède une propriété `_xmouse` et `_ymouse` qui renvoie l'emplacement de la souris dans son système de coordonnées. La position est toujours donnée par rapport au point d'alignement. Dans le scénario principal (`_level0`), le point d'alignement se trouve dans le coin supérieur gauche.



Les propriétés _xmouse et _ymouse dans le scénario principal et dans le scénario d'un clip

Les procédures suivantes représentent deux manières d'obtenir la position de la souris.

Pour obtenir la position actuelle de la souris dans le scénario principal :

1 Créez deux champs de texte dynamiques et nommez-les `x_pos` et `y_pos`.

2 Sélectionnez Fenêtre > Panneaux de développement > Actions pour ouvrir le panneau Actions si ce dernier n'est pas déjà visible.

3 Pour renvoyer la position de la souris dans le scénario principal, ajoutez le code suivant à une image du fichier SWF `_level0` :

```
x_pos = _root._xmouse;
y_pos = _root._ymouse;
```

Les variables `x_pos` et `y_pos` sont utilisées en tant que conteneurs pour stocker les valeurs des positions de la souris. Vous pouvez utiliser ces variables dans n'importe quel script de votre document. Dans le gestionnaire `onClipEvent()` suivant, les valeurs de `x_pos` et `y_pos` sont mises à jour à chaque fois que l'utilisateur déplace la souris.

```
onClipEvent(mouseMove) {
  x_pos = _root._xmouse;
  y_pos = _root._ymouse;
}
```

Pour obtenir la position actuelle de la souris dans un clip :

1 Créez un clip.

2 Sélectionnez l'occurrence de clip sur la scène. Utilisez l'inspecteur des propriétés pour le nommer `monClip`.

3 Sélectionnez Fenêtre > Panneaux de développement > Actions pour ouvrir le panneau Actions si ce dernier n'est pas déjà visible.

4 Utilisez le nom de l'occurrence de clip pour renvoyer la position de la souris dans le scénario principal.

Par exemple, vous pouvez placer l'instruction suivante dans tout scénario du fichier SWF `_level0` pour renvoyer la position `_ymouse` dans l'occurrence `monClip` :

```
x_pos = _root.monClip._xmouse
y_pos = _root.monClip._ymouse
```

Le code renvoie les positions `_xpos` et `_ypos` de la souris par rapport au point d'alignement.

5 Sélectionnez Contrôle > Tester l'animation pour tester l'animation.

Vous pouvez également déterminer la position de la souris dans un clip en utilisant les propriétés `_xmouse` et `_ymouse` dans un événement de clip, comme illustré dans le code suivant :

```
onClipEvent(enterFrame){
  xmousePosition = this._xmouse;
  ymousePosition = this._ymouse;
}
```

Pour plus d'informations sur les propriétés `_xmouse` et `_ymouse`, consultez `MovieClip._xmouse`, page 613 et `MovieClip._ymouse`, page 614.

Capture des pressions sur les touches

Vous pouvez utiliser les méthodes de la classe intégrée Key pour détecter la dernière touche sur laquelle l'utilisateur a appuyé. La classe Key ne requiert aucune fonction constructeur ; appelez simplement ses méthodes sur la classe même, comme illustré dans l'exemple suivant :

```
Key.getCode();
```

Vous pouvez obtenir des codes virtuels ou des valeurs ASCII (American Standard Code for Information Interchange) des pressions sur les touches :

- Pour obtenir le code réel de la dernière touche enfoncée par l'utilisateur, utilisez la méthode `getCode()`.
- Pour obtenir la valeur ASCII de la dernière touche enfoncée par l'utilisateur, utilisez la méthode `getAscii()`.

Un code est affecté à chaque touche physique du clavier. Ainsi, la touche Flèche gauche est associée au code virtuel 37. L'utilisation d'un code vous permet de vous assurer que les commandes de votre fichier SWF sont les mêmes sur chaque clavier, quelle que soit la langue ou la plate-forme utilisée.

Des valeurs ASCII sont affectées aux 127 premiers caractères de chaque jeu de caractères. Les valeurs ASCII fournissent des informations sur un caractère affiché à l'écran. Par exemple, la lettre « A » et la lettre « a » ont des valeurs ASCII différentes.

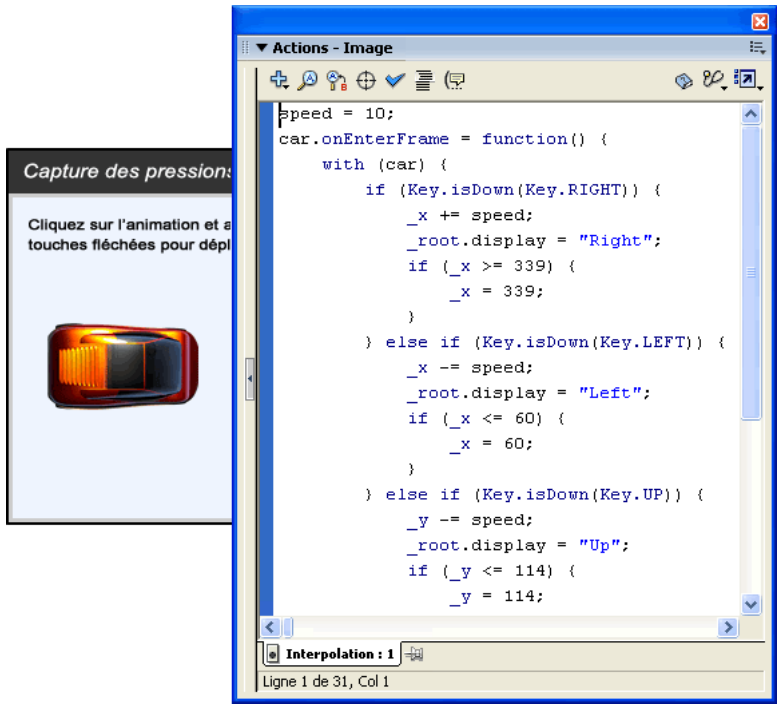
Pour décider des touches que vous allez utiliser et déterminer leurs codes, utilisez l'une des approches suivantes :

- La liste des codes de touches est publiée dans Annexe C, *Touches du clavier et valeurs de code correspondantes*, page 901.
- Utilisez une constante de classe Key. (Dans la boîte à outils Actions, cliquez sur la catégorie Classes intégrées, sur Animation, sur Touche et sur Constantes.)
- Affectez le gestionnaire `onClipEvent()` suivant à un clip, puis choisissez Contrôle > Tester l'animation et appuyez sur la touche souhaitée.

```
onClipEvent(keyDown) {
  trace(Key.getCode());
}
```

Le code de la touche s'affiche dans le panneau de sortie.

Les méthodes de classe Key sont fréquemment placées dans un gestionnaire d'événement. Dans l'exemple suivant, l'utilisateur déplace la voiture à l'aide des touches de direction. La méthode `Key.isDown()` indique si la touche enfoncée est la flèche orientée vers la droite, la gauche, le haut ou le bas. Le gestionnaire d'événement, `onEnterFrame`, détermine la valeur de `Key.isDown(codeDeTouche)` à partir des instructions `if`. En fonction de la valeur, il indique à Flash Player de mettre à jour la position de la voiture et d'afficher la direction.



La saisie avec les touches du clavier entraîne le déplacement de la voiture.

La procédure suivante indique comment capturer les pressions sur les touches pour déplacer un clip vers le haut, le bas, à gauche ou à droite de la scène, en fonction de la touche fléchée utilisée (haut, bas, gauche ou droite). Le clip est confiné dans une zone arbitraire d'une largeur de 400 pixels et d'une hauteur de 300 pixels. Par ailleurs, un champ de texte indique le nom de la touche enfoncée.

Pour créer un clip activé via le clavier :

1 Sur la scène, créez un clip qui se déplacera en réponse aux touches fléchées du clavier.

Dans cet exemple, le nom de l'occurrence de clip est `voiture`.

2 Sur la scène, créez un champ de texte dynamique qui sera mis à jour en fonction de la direction de la voiture. Utilisez l'inspecteur des propriétés pour lui donner le nom d'occurrence `display_txt`.

Remarque : Ne confondez pas le nom des variables avec celui des occurrences. Pour plus d'informations, consultez *A propos des noms d'occurrence et de variable de champ de texte*, page 142.

3 Sélectionnez l'image 1 dans le scénario et, le cas échéant, choisissez Fenêtre > Panneaux de développement > Actions pour ouvrir le panneau Actions.

4 Pour déterminer la distance parcourue par la voiture à l'écran après chaque pression sur une touche, définissez une variable `distance` et fixez sa valeur initiale à 10.

```
var distance = 10;
```

5 Pour créer le gestionnaire d'événement du clip voiture qui vérifie quelle touche de direction (gauche, droite, bas ou haut) est enfoncée, ajoutez le code suivant au panneau Actions :

```
voiture.onEnterFrame = function() {

}
```

6 Ajoutez une instruction `with` au corps du gestionnaire `onEnterFrame` et spécifiez `voiture` comme objet de l'instruction `with`.

Le code devrait avoir cette forme :

```
var distance = 10;
voiture.onEnterFrame = function() {
    with (voiture) {
    }
}
```

7 Pour vérifier si la touche de direction vers la droite est enfoncée et déplacer le clip voiture en conséquence, ajoutez du code au corps de l'instruction `with`. Le code devrait avoir cette forme :

```
distance = 10;
voiture.onEnterFrame = function() {
  with (voiture) {
    if (Key.isDown(Key.RIGHT)) {
      _x += distance;
      if (_x >= 400) {
        _x = 400;
      }
      _root.display_txt.text = "Droite";
    }
  }
}
```

Si la touche fléchée vers la droite est enfoncée, la propriété _x de la voiture est augmentée de la valeur spécifiée dans la variable `distance`. L'instruction `if` suivante teste si la valeur de la propriété _x du clip est supérieure ou égale à 400 (`if(_x >=400)`) ; si tel est le cas, sa position est fixée à 400. De plus, le mot *Droite* doit apparaître dans le fichier SWF.

8 Adaptez ce code pour vérifier si les autres touches de direction (gauche, haut ou bas) ont été enfoncées. Le code devrait avoir cette forme :

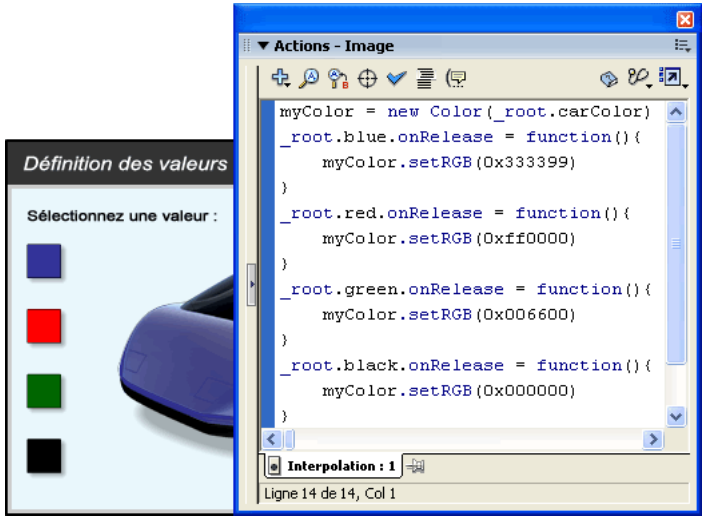
```
var distance = 10;
voiture.onEnterFrame = function() {
  with (voiture) {
    if (Key.isDown(Key.RIGHT)) {
      _x += distance;
      if (_x >= 400) {
        _x = 400;
      }
      _root.display_txt.text = "Droite";
    } else if (Key.isDown(Key.LEFT)) {
        _x -= distance;
        if (_x < 0) {
          _x = 0;
        }
        _root.display_txt.text = "Gauche";
    } else if (Key.isDown(Key.UP)) {
    _y -= distance;
      if (_y < 0) {
        _y = 0 ;
      }
    _root.display_txt.text = "Haut";
    } else if (Key.isDown(Key.DOWN)) {
      _y += distance;
      if (_y > 300) {
        _y = 300;
      }
    _root.display_txt.text = "Bas";
    }
  }
}
```

9 Choisissez Contrôle > Tester l'animation pour tester le fichier.

Pour plus d'informations sur les méthodes de la classe Key, consultez l'entrée *Classe Key* dans le Chapitre 12, *Dictionnaire ActionScript*, page 215.

Définition des valeurs des couleurs

Vous pouvez utiliser les méthodes de la classe Color intégrée pour définir la couleur d'un clip. La méthode `setRGB()` affecte des valeurs RVB (rouge, vert, bleu) hexadécimales au clip. L'exemple suivant utilise `setRGB()` pour changer la couleur d'un objet en réponse aux actions de l'utilisateur.



L'action de bouton crée un objet Color et change la couleur de la voiture en réponse aux actions de l'utilisateur.

Pour définir la valeur de couleur d'un clip :

1. Sélectionnez un clip sur la scène.
2. Dans l'inspecteur des propriétés, entrez `couleurDeVoiture` comme nom d'occurrence.
3. Créez un bouton nommé `puce de couleur`, placez quatre occurrences de ce bouton sur la scène et nommez-les `rouge`, `vert`, `bleu` et `noir`.
4. Sélectionnez l'image 1 dans le scénario principal, puis choisissez Fenêtre > Panneaux de développement > Actions.
5. Pour créer un objet Color qui cible le clip `couleurDeVoiture`, ajoutez le code suivant au panneau Actions :

   ```
   maCouleur = new Color(_root.couleurDeVoiture);
   ```

6. Pour que le bouton bleu colore en bleu le clip `couleurDeVoiture`, ajoutez le code suivant au panneau Actions :

   ```
   _root.bleu.onRelease = function(){
     maCouleur.setRGB(0x0000ff)
   }
   ```

   La valeur hexadécimale 0x0000ff représente le bleu. Le tableau suivant présente les autres couleurs que vous pouvez utiliser et leur valeur hexadécimale :

7 Répétez l'étape 6 pour les autres boutons (rouge, vert et noir) pour changer la couleur du clip à la couleur correspondante. Le code doit maintenant ressembler à ce qui suit :

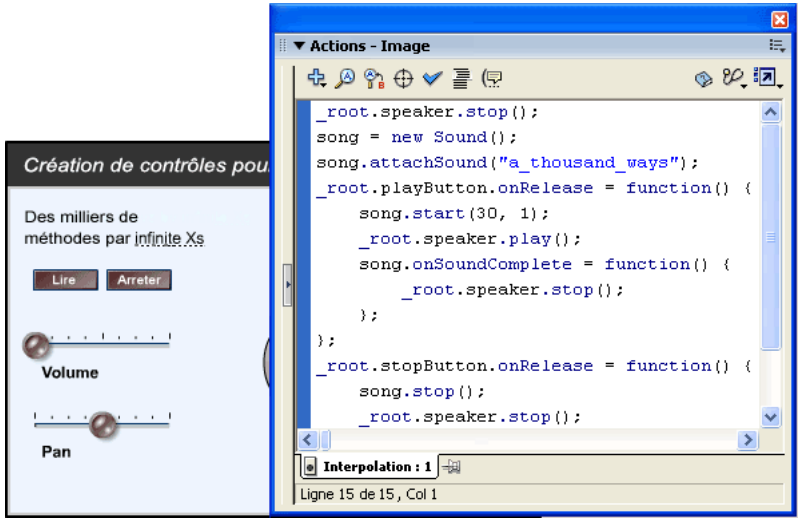
```
maCouleur = new Color(_root.couleurDeVoiture)
_root.bleu.onRelease = function(){
  maCouleur.setRGB(0x0000ff)
}
_root.rouge.onRelease = function(){
  maCouleur.setRGB(0xff0000)
}
_root.vert.onRelease = function(){
  maCouleur.setRGB(0x00ff00)
}
_root.noir.onRelease = function(){
  maCouleur.setRGB(0x000000)
}
```

8 Sélectionnez Contrôle > Tester l'animation pour changer la couleur du clip.

Pour plus d'informations sur les méthodes de la classe Color, consultez l'entrée *Classe Color* dans le Chapitre 12, *Dictionnaire ActionScript*, page 215.

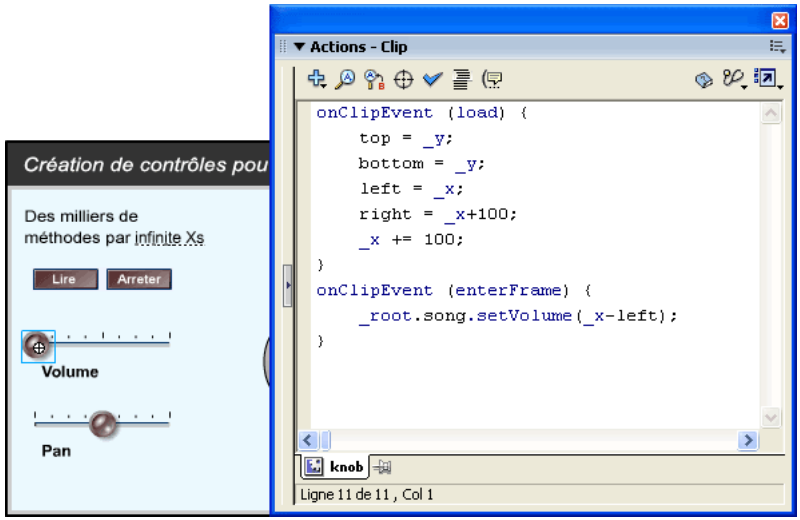
Création de commandes audio

La classe Sound intégrée permet de contrôler les sons d'un fichier SWF. Pour utiliser les méthodes de la classe Sound, vous devez d'abord créer un objet Sound. Faites ensuite appel à la méthode `attachSound()` pour insérer un son de la bibliothèque dans un fichier SWF pendant sa lecture.



Une chanson est lue lorsque l'utilisateur relâche le bouton de lecture.

La méthode `setVolume()` de la classe Sound contrôle le volume et la méthode `setPan()` règle la balance gauche et droite d'un son.



La méthode setVolume() est appelée lorsque l'utilisateur fait glisser le potentiomètre de volume.

Les procédures suivantes expliquent comment créer des contrôles audio semblables à ceux qui sont présentés ci-dessus.

Pour associer un son à un scénario :

1 Sélectionnez Fichier > Importer pour importer un son.

2 Sélectionnez le son dans la bibliothèque, puis cliquez dessus avec le bouton droit de la souris (Windows) ou en appuyant sur la touche Contrôle (Macintosh), et choisissez Liaison.

3 Activez les options Exporter pour ActionScript et Exporter dans la première image, puis affectez au son l'identifiant `au_clair_de_la_lune`.

4 Ajoutez un bouton sur la scène et appelez-le `boutonDeLecture`.

5 Ajoutez un bouton sur la scène et appelez-le `boutonDarrêt`.

6 Ajoutez un clip sur la scène et appelez-le `hautParleur`.

7 Sélectionnez l'image 1 dans le scénario principal, puis choisissez Fenêtre > Panneaux de développement > Actions. Ajoutez le code suivant au panneau Actions

```
hautParleur.stop();
chanson = new Sound();
chanson.onSoundComplete = function() {
  hautParleur.stop();
};
chanson.attachSound("au_clair_de_la_lune");
boutonDeLecture.onRelease = function() {
  chanson.start();
  hautParleur.play();
```

```
};
boutonDarrêt.onRelease = function () {
  chanson.stop();
  hautParleur.stop();
}
```

Ce code commence par arrêter le clip du haut-parleur. Il crée ensuite un objet Sound (`chanson`) et lui associe le son dont l'identifiant de liaison correspond à `au_clair_de_la_lune`. Il définit ensuite un gestionnaire `onSoundComplete` pour l'objet chanson, qui arrête le clip `haut-parleur` une fois que le son est terminé. Enfin, les gestionnaires `onRelease` associés aux objets `playButton` et `stopButton` démarrent et arrêtent le son à l'aide des méthodes `Sound.start()` et `Sound.stop()`, et lisent et arrêtent le clip `haut-parleur`.

8 Sélectionnez Contrôle > Tester l'animation pour entendre le son.

Pour créer une commande de volume réglable :

1 Faites glisser un bouton sur la scène.

2 Sélectionnez le bouton et choisissez Modification > Convertir en symbole. N'oubliez pas de sélectionner le comportement du clip.

Cela crée un clip avec le bouton sur sa première image.

3 Sélectionnez le clip et choisissez Edition > Modifier la sélection.

4 Sélectionnez le bouton et choisissez Fenêtre > Panneaux de développement > Actions.

5 Entrez les actions suivantes :

```
on(press) {
    startDrag(this, false, gauche, haut, droite, bas);
}
on (release) {
    stopDrag();
}
```

Les paramètres de `startDrag()`, `gauche`, `haut`, `droite` et `bas`, sont des variables définies dans une action de clip.

6 Choisissez Edition > Modifier le document pour revenir au scénario principal.

7 Sélectionnez le clip sur la scène.

8 Entrez les actions suivantes :

```
onClipEvent(load){
    top = _y;
    bottom = _y;
    left = _x;
    right = _x+100;
    _x += 100;
}
onClipEvent (enterFrame) {
    _parent.song.setVolume(_x-left);
}
```

9 Sélectionnez Contrôle > Tester l'animation pour utiliser la commande de volume.

Pour créer une commande de balance réglable :

1 Faites glisser un bouton sur la scène.

2 Sélectionnez le bouton et choisissez Insertion > Convertir en symbole. Sélectionnez le clip adéquat.

3 Sélectionnez le clip et choisissez Edition > Modifier le symbole.

4 Sélectionnez le bouton et choisissez Fenêtre > Panneaux de développement > Actions.

5 Entrez les actions suivantes :

```
on(press) {
  startDrag ("", false, left, top, right, bottom);
  dragging = true;
}
on (release, releaseOutside) {
  stopDrag ();
  dragging = false;
}
```

Les paramètres de `startDrag()`, `gauche`, `haut`, `droite` et `bas`, sont des variables définies dans une action de clip.

6 Choisissez Edition > Modifier le document pour revenir au scénario principal.

7 Sélectionnez le clip sur la scène.

8 Entrez les actions suivantes :

```
onClipEvent(load) {
  top=_y;
  bottom=_y;
  left=_x-50;
  right=_x+50;
  center=_x;
}

onClipEvent(enterFrame){
  if (dragging==true){
    _parent.setPan((_x-center)*2);
  }
}
```

9 Sélectionnez Contrôle > Tester l'animation pour utiliser le curseur de balance.

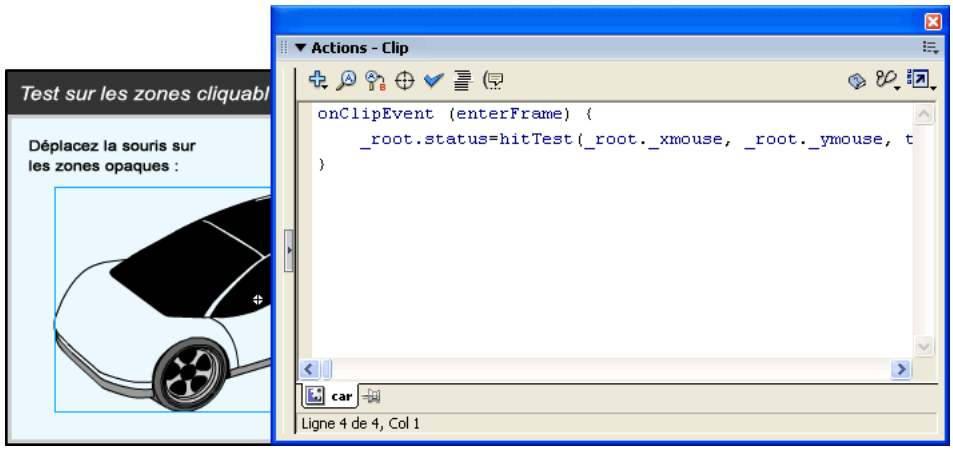
Pour plus d'informations sur les méthodes de la classe Sound, consultez l'entrée *Classe Sound* dans le Chapitre 12, *Dictionnaire ActionScript*, page 215.

Détection des collisions

La méthode `hitTest()` de la classe MovieClip détecte les collisions dans un fichier SWF. Elle vérifie si un objet est entré en collision avec un clip et renvoie une valeur booléenne (`true` ou `false`).

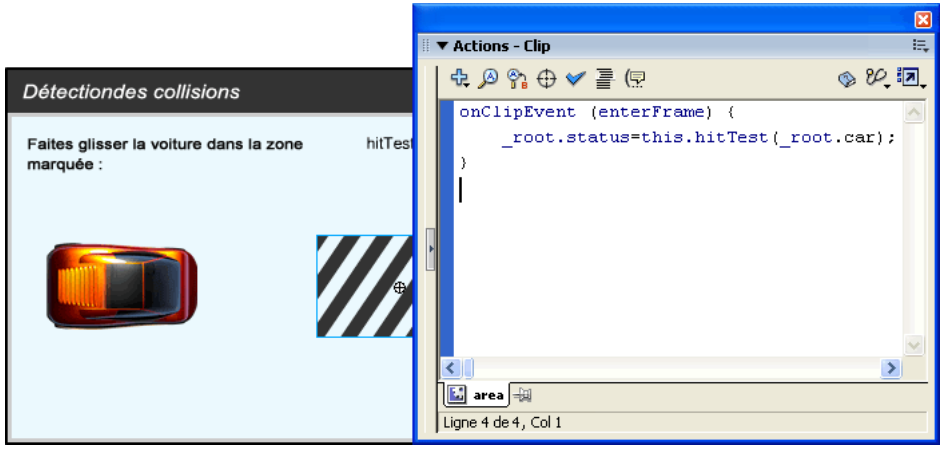
Il existe deux situations où il peut être utile de savoir si une collision s'est produite : pour tester si l'utilisateur a atteint une zone statique précise de la scène, et pour déterminer si un clip en a atteint un autre. La méthode `hitTest()` permet de déterminer ces résultats.

Vous pouvez utiliser les paramètres de `hitTest()` pour définir les coordonnées *x* et *y* d'une zone réactive sur la scène ou utiliser le chemin cible d'un autre clip comme zone réactive. Lorsque vous définissez *x* et *y*, `hitTest()` renvoie la valeur `true` si le point identifié par (*x*,*y*) n'est pas transparent. Lorsqu'une cible est transmise à `hitTest()`, les cadres de délimitation des deux clips sont comparés. S'ils se chevauchent, `hitTest()` renvoie la valeur `true`. S'ils ne se croisent pas, `hitTest()` renvoie la valeur `false`.



« True » apparaît dans le champ de texte lorsque le pointeur de la souris se trouve au-dessus de la carrosserie de la voiture.

Vous pouvez également utiliser `hitTest()` pour tester une collision entre deux clips.



« True » apparaît dans le champ de texte lorsqu'un clip en touche un autre.

Les procédures suivantes indiquent comment détecter les collisions, en utilisant une voiture comme exemple.

Pour détecter la collision entre un clip et un point de la scène :

1 Créez un clip sur la scène et entrez `champ` comme nom d'occurrence dans l'inspecteur des propriétés.

2 Créez un champ de texte dynamique sur la scène et entrez `état` comme nom d'occurrence dans l'inspecteur des propriétés.

3 Sélectionnez, dans le scénario, la première image du calque 1.

4 Sélectionnez Fenêtre > Panneaux de développement > Actions pour ouvrir le panneau Actions si ce dernier n'est pas déjà visible.

5 Entrez le code suivant dans le panneau Actions :

```
box.onEnterFrame = function () {
  état.text = this.hitTest(_xmouse, _ymouse, true);
}
```

6 Choisissez Contrôle > Tester l'animation et passez la souris sur le clip pour tester la collision.

La valeur `true` est affichée lorsque la souris se trouve au-dessus d'un pixel non transparent.

Pour détecter la collision entre deux clips :

1 Faites glisser deux clips jusqu'à la scène et affectez-leur les noms d'occurrence `voiture` et `zone`.

2 Créez un champ de texte dynamique sur la scène et entrez `état` comme nom d'occurrence dans l'inspecteur des propriétés.

3 Sélectionnez, dans le scénario, la première image du calque 1.

4 Sélectionnez Fenêtre > Panneaux de développement > Actions pour ouvrir le panneau Actions si ce dernier n'est pas déjà visible.

5 Entrez le code suivant dans le panneau Actions :

```
zone.onEnterFrame = function () {
  état.text=this.hitTest(voiture);
}
voiture.onPress = function (){
  this.startDrag(false);
  updateAfterEvent();
}
voiture.onRelease = function () {
  this.stopDrag();
}
```

6 Choisissez Contrôle > Tester l'animation et faites glisser le clip pour tester la collision.

Lorsque le cadre de délimitation de la voiture touche celui de la zone, l'état devient `true`.

Pour plus d'informations, consultez *`MovieClip.hitTest()`* dans le Chapitre 12, *Dictionnaire ActionScript*, page 215.

Création d'un outil de dessin de ligne simple

Vous pouvez utiliser les méthodes de la classe MovieClip pour dessiner des formes et des remplissages sur la scène en cours de lecture. Vous pouvez ainsi créer des outils de dessin pour les utilisateurs et tracer des formes dans le fichier SWF en réponse à des événements. Les méthodes de dessin sont `beginFill()`, `beginGradientFill()`, `clear()`, `curveTo()`, `endFill()`, `lineTo()`, `lineStyle()` et `moveTo()`. Vous pouvez appliquer ces méthodes à toute occurrence de clip (par exemple, `monClip.lineTo()`) ou à un niveau (`_root.curveTo()`).

Les méthodes `lineTo()` et `curveTo()` vous permettent respectivement de dessiner des lignes et des courbes. La méthode `lineStyle()` permet de spécifier une couleur et une épaisseur de ligne et un paramètre alpha pour une ligne ou une courbe. La méthode de dessin `moveTo()` place la position de dessin actuelle aux coordonnées *x* et *y* de scène spécifiées.

Les méthodes `beginFill()` et `beginGradientFill()` remplissent respectivement un chemin fermé avec une couleur de remplissage unie ou dégradée, tandis que `endFill()` applique le remplissage spécifié dans le dernier appel à `beginFill()` ou `beginGradientFill()`. La méthode `clear()` efface ce qui a été dessiné dans l'objet clip spécifié.

Pour plus d'informations, consultez `MovieClip.beginFill()`, page 551, `MovieClip.beginGradientFill()`, page 551, `MovieClip.clear()`, page 555, `MovieClip.curveTo()`, page 558, `MovieClip.endFill()`, page 561, `MovieClip.lineTo`, page 575, `MovieClip.lineStyle()`, page 574 et `MovieClip.moveTo`, page 582.

Pour créer un outil de dessin de ligne simple :

1 Dans un nouveau document, créez un bouton sur la scène et entrez `clear_btn` comme nom d'occurrence dans l'inspecteur des propriétés.

2 Sélectionnez l'image 1 dans le scénario et, le cas échéant, choisissez Fenêtre > Panneaux de développement > Actions pour ouvrir le panneau Actions.

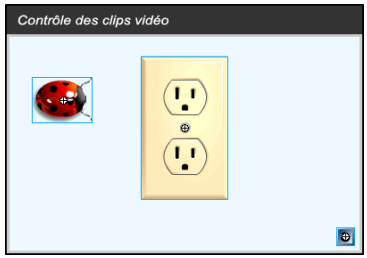
3 Dans le panneau Actions, entrez le code suivant :

```
_root.onMouseDown = function() {
    _root.lineStyle(5, 0xFF0000, 100);
    _root.moveTo(_root._xmouse, _root._ymouse);
    isDrawing = true;
};
_root.onMouseMove = function() {
    if (isDrawing == true) {
        _root.lineTo(_root._xmouse, _root._ymouse);
        updateAfterEvent();
    }
};
_root.onMouseUp = function() {
    isDrawing = false;
};
clear_btn.onRelease = function() {
    _root.clear();
};
```

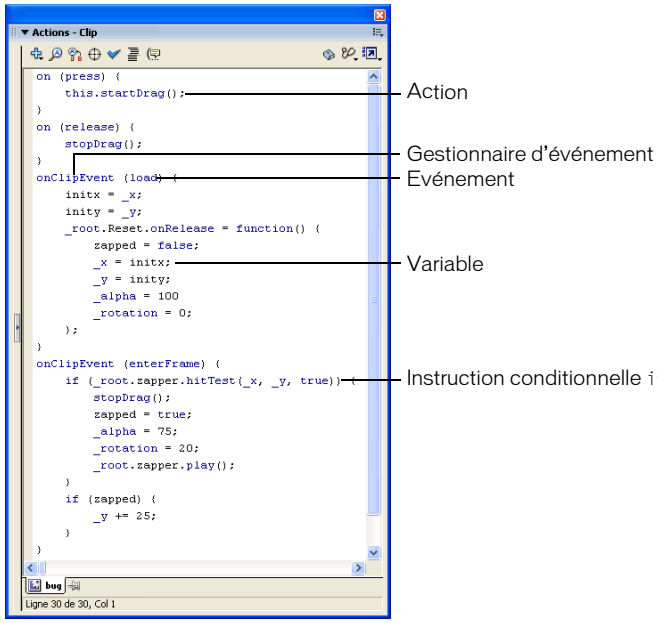
4 Sélectionnez Contrôle > Tester l'animation pour tester l'animation. Cliquez et faites glisser la souris pour dessiner une ligne sur la scène. Cliquez sur le bouton pour effacer ce que vous avez dessiné.

Structure d'un exemple de script

Lorsqu'un utilisateur fait glisser la coccinelle vers la prise électrique dans le fichier d'exemple zapper.swf (disponible dans le guide Utilisation de Flash de l'aide), la coccinelle tombe et la prise tremble. Le scénario principal ne comprend qu'une image et contient trois objets : la coccinelle, la prise et un bouton de réinitialisation. Chacun de ces objets est une occurrence de clip.



Le fichier SWF contient un script associé à l'occurrence `bug`, comme illustré dans le panneau Actions suivant :



Panneau Actions et script associé à l'occurrence bug

Le nom de l'occurrence de coccinelle est `bug` et le nom de l'occurrence de prise est `zapper`. Dans le script, la référence à la coccinelle est `this`, car le script est associé à la coccinelle et le mot réservé `this` désigne l'objet qui le contient.

Il existe deux gestionnaires `onClipEvent()` avec deux événements différents : `load` et `enterFrame`. Les actions contenues dans l'instruction `onClipEvent(load)` sont exécutées une seule fois, au chargement du fichier SWF. Les actions contenues dans l'instruction `onClipEvent(enterFrame)` sont exécutées à chaque fois que la tête de lecture entre dans une image. Même dans un fichier SWF composé d'une image unique, la tête de lecture entre plusieurs fois dans cette image et le script est exécuté à plusieurs reprises. Les actions suivantes ont lieu dans chaque gestionnaire `onClipEvent()` :

onClipEvent(load) Deux variables, `initx` et `inity`, sont définies de manière à stocker les positions *x* et *y* initiales de l'occurrence de clip `bug`. Une fonction est définie et affectée à l'événement `onRelease` de l'occurrence `Reset`. Cette fonction est appelée à chaque clic de la souris sur le bouton de réinitialisation. La fonction replace la coccinelle en position de départ sur la scène, réinitialise ses valeurs alpha et de rotation et redéfinit la variable `zapped` sur la valeur `false`.

onClipEvent(enterFrame) Une instruction conditionnelle `if` utilise la méthode `hitTest` pour vérifier si l'occurrence de la coccinelle touche l'occurrence de la prise (`_root.zapper`). Cette évaluation peut avoir deux types de résultats, `true` ou `false` :

```
onClipEvent(load){
  initx = _x;
  inity = _y;
  _root.Reset.onRelease = function() {
    zapped = false;
    _x = initx;
    _y = inity;
    _alpha = 100
    _rotation = 0;
  };
}
```

Si la méthode `hitTest()` renvoie la valeur `true`, la méthode `stopDrag()` est appelée, la variable `zapper` est définie sur la valeur `true`, les propriétés alpha et rotation sont modifiées et l'exécution de l'occurrence `zapped` est lancée.

Si la méthode `hitTest()` renvoie la valeur `false`, le code entre `{}` qui suit immédiatement l'instruction `if` n'est pas exécuté.

Deux gestionnaires `on()` sont associés à l'occurrence `bug` avec deux événements différents : `press` et `release`. Les actions de l'instruction `on(press)` sont exécutées une fois le bouton de la souris appuyé au-dessus de l'occurrence `bug`. Les actions de l'instruction `on(release)` sont exécutées après le relâchement du bouton de la souris au-dessus de l'occurrence `bug`. Les actions suivantes se déroulent à l'intérieur de chaque gestionnaire `onClipEvent()` :

on(press) Une action `startDrag()` rend la coccinelle déplaçable. Le script étant associé à l'occurrence `bug`, le mot-clé `this` indique que c'est l'occurrence `bug` qui peut être déplacée :

```
on(press) {
  this.startDrag();
}
```

on(release) Une action `stopDrag()` arrête l'action de déplacement :

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

Pour lire le fichier SWF, consultez le Guide de référence ActionScript, dans l'aide.

PARTIE III
Utilisation des objets et des classes

Cette section présente le modèle d'objet d'exécution Macromedia Flash ainsi que ses fonctionnalités, puis met l'accent sur l'utilisation des clips et du texte. Elle décrit également la manière de créer vos propres classes et interfaces à l'aide d'ActionScript 2.0.

CHAPITRE 6
Utilisation des classes intégrées

En plus des principaux éléments de langage, constructions (boucles `for` et `while`, par exemple) et types de données (nombres, chaînes et tableaux) ActionScript décrits précédemment (consultez *Notions de base du langage ActionScript*, page 27), ActionScript fournit également plusieurs classes intégrées ou *types de données complexes*. Ces classes vous offrent de nombreuses options et fonctionnalités de scripts.

Certaines de ces classes reposent sur la spécification ECMAScript et sont appelées *classes ActionScript de base*. Elles comprennent les classes Array, Boolean, Date et Math. Pour plus d'informations, consultez *Classes de base*, page 121.

Les autres classes intégrées ActionScript sont spécifiques à Macromedia Flash et au modèle d'objet Flash Player. La distinction entre les classes ActionScript de base et les classes spécifiques à Flash est similaire à celle qui existe entre les classes JavaScript de base et côté client. Tout comme les classes JavaScript côté client permettent de contrôler l'environnement du client (le navigateur web et le contenu des pages web), les classes spécifiques à Flash permettent de contrôler l'apparence et le comportement d'une application Flash à l'exécution.

Ce chapitre présente les classes intégrées ActionScript, passe en revue les tâches qu'elles permettent d'effectuer et propose des exemples de code. Pour un aperçu de ces classes, consultez *Aperçu des classes intégrées*, page 120. Pour un aperçu de l'utilisation des classes et objets en programmation orientée objet, consultez *A propos des classes et des occurrences*, page 119.

A propos des classes et des occurrences

En programmation orientée objet, une *classe* définit une catégorie d'objet. Une classe décrit les propriétés (données) et le comportement (méthodes) d'un objet, comme un calque d'architecte décrit les propriétés d'un immeuble. Pour utiliser les propriétés et méthodes définies par une classe, vous devez tout d'abord créer une *occurrence* de cette classe. La relation entre une occurrence et sa classe est similaire à la relation entre une maison et le calque d'architecte correspondant.

Création d'un nouvel objet

Pour créer une occurrence de classe ActionScript, utilisez l'opérateur `new` pour appeler la fonction constructeur de la classe. La fonction constructeur porte toujours le même nom que la classe dont elle renvoie une occurrence, que vous attribuez généralement à une variable.

Le code suivant crée par exemple un nouvel objet Sound :

```
var chanson:Sound= new Sound();
```

Dans certains cas, il n'est pas nécessaire de créer une occurrence de classe pour l'utiliser. Pour plus d'informations, consultez *A propos des membres de classe (membres statiques)*, page 120.

Accès aux propriétés des objets

Utilisez l'opérateur point (`.`) pour accéder à la valeur d'une propriété d'objet. Entrez le nom de l'objet à gauche du point et le nom de la propriété à droite. Par exemple, dans l'instruction suivante, `monObjet` représente l'objet et `nom` la propriété :

```
monObjet.nom
```

Le code suivant crée un objet TextField, puis définit sa propriété `autoSize` sur `true`.

```
var mon_txt = new TextField();
mon_txt.autoSize = true;
```

Vous pouvez également utiliser l'opérateur d'accès tableau (`[]`) pour accéder aux propriétés d'un objet. Pour plus d'informations, consultez *Opérateurs point et accès tableau*, page 52.

Appel des méthodes d'un objet

Vous pouvez appeler une méthode d'objet en utilisant l'opérateur point (`.`) suivi de la méthode. Par exemple, le code suivant crée un objet Sound et appelle sa méthode `setVolume()` :

```
monSon = new Sound(this);
monSon.setVolume(50);
```

A propos des membres de classe (membres statiques)

Certaines classes intégrées ActionScript possèdent des *membres de classe* (ou *membres statiques*). Pour invoquer les membres de classe (propriétés et méthodes) ou y accéder, vous utilisez le nom de la classe, et non l'occurrence de celle-ci. Il ne faut donc pas créer une occurrence de la classe pour utiliser ces propriétés et méthodes.

Par exemple, toutes les propriétés de la classe Math sont statiques. Le code suivant appelle la méthode `max()` de la classe Math pour déterminer lequel de deux nombres est le plus grand.

```
var plusGrandNombre = Math.max(10, 20);
```

Aperçu des classes intégrées

Cette section répertorie toutes les classes ActionScript et contient une brève description de chaque classe et des références croisées menant aux autres sections pertinentes de la documentation.

Classes de base

Les classes ActionScript de base sont celles qui sont directement empruntées à ECMAScript. Ces classes résident dans le sous-dossier Classes intégrées > de base de la boîte à outils Actions.

Classe	Description
Arguments	Tableau contenant les valeurs transmises comme paramètres pour toute fonction. Consultez l'entrée *Classe Arguments* dans le Chapitre 12, *Dictionnaire ActionScript*, page 215.
Array	La classe Array contient les méthodes et propriétés réservées aux objets Array. Consultez l'entrée *Classe Array* dans le Chapitre 12, *Dictionnaire ActionScript*, page 215.
Boolean	La classe Boolean est une enveloppe pour les valeurs booléennes (`true` ou `false`). Consultez l'entrée *Classe Boolean* dans le Chapitre 12, *Dictionnaire ActionScript*, page 215.
Button	La classe Button fournit des méthodes et des propriétés permettant d'utiliser les objets Button. Consultez l'entrée *Classe Button* dans le Chapitre 12, *Dictionnaire ActionScript*, page 215.
Date	La classe Date donne accès à des valeurs de date et d'heure relatives à l'heure universelle (GMT) ou au système d'exploitation sur lequel Flash Player s'exécute. Consultez l'entrée *Classe Date* dans le Chapitre 12, *Dictionnaire ActionScript*, page 215.
Error	La classe Error contient des informations sur les erreurs survenant dans vos scripts. En règle générale, vous utilisez l'instruction `throw` pour générer une condition d'erreur, que vous pouvez ensuite gérer à l'aide d'une instruction `try..catch..finally`. Consultez `try..catch..finally` et les entrées *Classe Error* dans le Chapitre 12, *Dictionnaire ActionScript*, page 215.
Function	La classe Function est la représentation sous forme de classe de toutes les fonctions ActionScript, y compris les fonctions Action Script natives et celles que vous définissez. Consultez l'entrée *Classe Function* dans le Chapitre 12, *Dictionnaire ActionScript*, page 215.
Math	La classe Math vous permet d'accéder à des constantes et fonctions mathématiques et de les manipuler. Toutes les propriétés et méthodes de la classe Math sont statiques et doivent être appelées à l'aide de la syntaxe `Math.méthode(paramètre)` ou `Math.constante`. Consultez l'entrée *Classe Math* dans le Chapitre 12, *Dictionnaire ActionScript*, page 215.
Number	La classe Number est une enveloppe pour le type de données primitif de nombre. Consultez l'entrée *Classe Number* dans le Chapitre 12, *Dictionnaire ActionScript*, page 215.
Object	La classe Object est à la racine de la hiérarchie de classes ActionScript. Toutes les autres classes héritent de ses méthodes et propriétés. Consultez l'entrée *Classe Object* dans le Chapitre 12, *Dictionnaire ActionScript*, page 215.
String	La classe String est une enveloppe pour le type de données primitif de chaîne, ce qui vous permet d'utiliser les méthodes et les propriétés de l'objet String pour manipuler les types primitifs de valeurs de chaîne. Consultez l'entrée *Classe String* dans le Chapitre 12, *Dictionnaire ActionScript*, page 215.

Classes spécifiques à Flash Player

Les tableaux suivants répertorient les classes spécifiques à Flash Player et au modèle d'exécution Flash. Ces classes sont généralement divisées en quatre catégories : les classes Movie (qui autorisent un contrôle général des fichiers SWF et de Flash Player), les classes Media (pour l'utilisation de son et de vidéo), les classes Client-serveur (pour l'utilisation de XML et d'autres sources de données externes) et les classes Authoring (qui permettent de contrôler l'environnement de programmation Flash).

Remarque : Cette catégorisation détermine l'emplacement des classes dans la boîte à outils Actions ; elle n'a aucune incidence sur leur utilisation.

Classes Movie

Les classes Movie permettent de contrôler la plupart des éléments visuels dans un fichier SWF, y compris les clips, les champs de texte et les boutons. Elles résident dans le sous-dossier Classes intégrées > Animation de la boîte à outils Actions.

Classe	Description
Accessibility	La classe Accessibility gère la communication entre les fichiers SWF et les applications de lecture d'écran. Conjointement avec la propriété globale `_accProps`, les méthodes de cette classe permettent de contrôler les propriétés accessibles des clips, des boutons et des champs de texte lors de l'exécution. Consultez `_accProps` et *Classe Accessibility* dans le Chapitre 12, *Dictionnaire ActionScript*, page 215.
Button	Tout bouton d'un fichier SWF est une occurrence de la classe Button. La classe Button fournit des méthodes, propriétés et gestionnaires d'événements pour l'utilisation des boutons. Consultez l'entrée *Classe Button* dans le Chapitre 12, *Dictionnaire ActionScript*, page 215.
Color	La classe Color vous permet d'extraire et de définir les valeurs des couleurs RVB des objets d'un clip. Pour plus d'informations, consultez l'entrée *Classe Color* dans le Chapitre 12, *Dictionnaire ActionScript*, page 215. Vous trouverez un exemple d'utilisation de la classe Color pour modifier les couleurs d'un clip à la section *Définition des valeurs des couleurs*, page 106.
ContextMenu	La classe ContextMenu vous permet d'agir sur le contenu du menu contextuel de Flash Player. Vous pouvez associer des objets ContextMenu séparés à des objets MovieClip, Button ou TextField à l'aide de la propriété `menu` de ces classes. Vous avez aussi tout loisir d'utiliser la classe ContextMenuItem pour ajouter des éléments de menu personnalisés à un objet ContextMenu. Consultez les entrées *Classe ContextMenu* et *Classe ContextMenuItem* dans le Chapitre 12, *Dictionnaire ActionScript*, page 215.
ContextMenuItem	La classe ContextMenuItem vous permet de créer des éléments de menu dans le menu contextuel de Flash Player. Pour ajouter les éléments de menu que vous créez à l'aide de cette classe au menu contextuel de Flash Player, vous utilisez la classe ContextMenu. Consultez les entrées *Classe ContextMenu* et *Classe ContextMenuItem* dans le Chapitre 12, *Dictionnaire ActionScript*, page 215.

Classe	Description
Key	La classe Key propose des méthodes et propriétés permettant d'obtenir des informations sur le clavier et les touches qui sont enfoncées. Pour plus d'informations, consultez l'entrée *Classe Key* dans le Chapitre 12, *Dictionnaire ActionScript*, page 215. Vous trouverez un exemple de capture des pressions sur les touches en vue de créer un fichier SWF interactif à la section *Capture des pressions sur les touches*, page 102.
LocalConnection	La classe LocalConnection permet à deux fichiers SWF qui s'exécutent sur un même ordinateur de communiquer. Consultez l'entrée *Classe LocalConnection* dans le Chapitre 12, *Dictionnaire ActionScript*, page 215.
Mouse	La classe Mouse vous permet de contrôler la souris dans un fichier SWF. Vous pouvez l'utiliser pour masquer ou afficher le pointeur de la souris, par exemple. Pour plus d'informations, consultez l'entrée *Classe Mouse* dans le Chapitre 12, *Dictionnaire ActionScript*, page 215. Vous trouverez un exemple d'utilisation de la classe Mouse à la section *Création d'un pointeur de souris personnalisé*, page 100.
MovieClip	Chaque clip d'une animation Flash est une occurrence de la classe MovieClip. Utilisez les méthodes et propriétés de cette classe pour contrôler les objets de clip. Consultez le Chapitre 7, *Utilisation des clips*, page 127 et l'entrée *Classe MovieClip* dans le Chapitre 12, *Dictionnaire ActionScript*, page 215.
MovieClipLoader	La classe MovieClipLoader vous permet de suivre la progression du téléchargement des fichiers SWF et JPEG à l'aide d'un mécanisme d'écouteur d'événement. Consultez *Préchargement des fichiers SWF et JPEG*, page 210 et l'entrée *Classe MovieClipLoader* dans le Chapitre 12, *Dictionnaire ActionScript*, page 215.
PrintJob	La classe PrintJob vous permet d'imprimer du contenu rendu dynamiquement, ainsi que des documents de plusieurs pages. Consultez l'entrée *Classe PrintJob* dans le Chapitre 12, *Dictionnaire ActionScript*, page 215 et Utilisation de la classe PrintJob d'ActionScript, dans le guide Utilisation de Flash de l'aide.
Selection	La classe Selection vous permet d'extraire et de définir le focus, les étendues de sélection et les points d'insertion de champs de texte. Consultez l'entrée *Classe Selection* dans le Chapitre 12, *Dictionnaire ActionScript*, page 215.
SharedObject	La classe SharedObject vous permet de stocker des données localement sur l'ordinateur client. Consultez l'entrée *Classe SharedObject* dans le Chapitre 12, *Dictionnaire ActionScript*, page 215.
Stage	La classe Stage fournit des informations sur les dimensions, l'alignement et le mode d'échelle d'un fichier SWF, et signale les événements de redimensionnement Stage. Consultez l'entrée *Classe Stage* dans le Chapitre 12, *Dictionnaire ActionScript*, page 215.
System	La classe System fournit des informations sur Flash Player et sur le système sur lequel il s'exécute (la résolution d'affichage et la langue système en cours, par exemple). Elle vous permet aussi d'afficher ou de masquer le panneau Paramètres Flash Player et de modifier les paramètres de sécurité du fichier SWF. Consultez l'entrée *Classe System* dans le Chapitre 12, *Dictionnaire ActionScript*, page 215.

Classe	Description
TextField	La classe TextField vous permet de contrôler les champs dynamiques et de saisie de texte. Consultez le Chapitre 8, *Utilisation du texte*, page 141 et l'entrée *Classe TextField* dans le Chapitre 12, *Dictionnaire ActionScript*, page 215.
TextField.StyleSheet	La classe TextField.StyleSheet (qui fait partie de la classe TextField) vous permet de créer et d'appliquer des styles de texte CSS à des textes au format HTML ou XML. Consultez *Formatage de texte avec les feuilles de style en cascade*, page 145 et l'entrée *Classe TextField.StyleSheet* dans le Chapitre 12, *Dictionnaire ActionScript*, page 215.
TextFormat	La classe TextFormat vous permet de définir le formatage des caractères et des paragraphes dans les objets TextField. Consultez *Utilisation de la classe TextFormat*, page 144 et l'entrée *Classe TextFormat* dans le Chapitre 12, *Dictionnaire ActionScript*, page 215.

Media classes

Les classes Media permettent de contrôler le son et la vidéo lors de la lecture d'un fichier SWF, ainsi que d'accéder au microphone et à la caméra de l'utilisateur, le cas échéant. Ces classes résident dans le sous-dossier Classes intégrées > Support de la boîte à outils Actions.

Classe	Description
Camera	La classe Camera vous permet d'accéder à la caméra de l'utilisateur, s'il en a installé une. Utilisé de concert avec Flash Communication Server MX, votre fichier SWF peut capturer, diffuser et enregistrer les images et les données vidéo de la caméra d'un utilisateur. Consultez l'entrée *Classe Camera* dans le Chapitre 12, *Dictionnaire ActionScript*, page 215.
Microphone	La classe Microphone vous permet d'accéder au microphone de l'utilisateur, s'il en a installé un. Utilisé de concert avec Flash Communication Server MX, votre fichier SWF peut diffuser et enregistrer les sons du microphone d'un utilisateur. Consultez l'entrée *Classe Microphone* dans le Chapitre 12, *Dictionnaire ActionScript*, page 215.
NetConnection	La classe NetConnection vous permet d'établir une connexion locale en flux continu pour la lecture de fichiers Flash Video (FLV) à partir d'une adresse HTTP ou du système de fichiers local. Pour plus d'informations, consultez l'entrée *Classe NetConnection* dans le Chapitre 12, *Dictionnaire ActionScript*, page 215. Pour plus d'informations sur la lecture de fichiers FLV sur Internet, consultez *Lecture dynamique des fichiers FLV externes*, page 209.
NetStream	La classe NetStream vous permet de contrôler la lecture des fichiers FLV. Pour plus d'informations, consultez l'entrée *Classe NetStream* dans le Chapitre 12, *Dictionnaire ActionScript*, page 215. Pour plus d'informations sur la lecture de fichiers FLV sur Internet, consultez *Lecture dynamique des fichiers FLV externes*, page 209.

Classe	Description
Sound	La classe Sound vous permet de contrôler les sons dans un fichier SWF. Pour plus d'informations, consultez l'entrée *Classe Sound* dans le Chapitre 12, *Dictionnaire ActionScript*, page 215. Vous trouverez un exemple d'utilisation de la classe Sound pour créer des commandes de balance et de réglage du volume à la section *Création de commandes audio*, page 107.
Video	La classe Video vous permet d'afficher les objets vidéo d'un fichier SWF. Consultez l'entrée *Classe Video* dans le Chapitre 12, *Dictionnaire ActionScript*, page 215.

Classes Client-serveur

Le tableau suivant répertorie les classes vous permettant d'envoyer et de recevoir des données provenant de sources externes, ou de communiquer avec des serveurs d'applications via FTP, HTTP ou HTTPS.

Remarque : Dans Flash Player 7, un fichier SWF ne peut charger des données qu'à partir du domaine qui l'a servi. Pour plus d'informations, consultez *Fonctions de sécurité de Flash Player*, page 199 et *A propos de l'autorisation de chargement de données inter-domaines*, page 201.

Ces classes résident dans le sous-dossier Classes intégrées > Client/Serveur de la boîte à outils Actions.

Classe	Description
LoadVars	La classe LoadVars peut se substituer à l'action `loadVariables()` pour le transfert de variables entre un fichier SWF et un serveur en paires nom-valeur. Consultez *Utilisation de la classe LoadVars*, page 190 et l'entrée *Classe LoadVars* dans le Chapitre 12, *Dictionnaire ActionScript*, page 215.
XML	La classe XML étend la classe XMLNode et fournit des méthodes, des propriétés et des gestionnaires d'événement pour l'utilisation de données au format XML, y compris pour le chargement et l'analyse de code XML externe, la création de documents XML et la navigation dans les arborescences de documents XML. Consultez *Utilisation de la classe XML*, page 192 et l'entrée *Classe XML* dans le Chapitre 12, *Dictionnaire ActionScript*, page 215.
XMLNode	La classe XMLNode représente un nœud unique dans une arborescence de documents XML. Elle constitue la superclasse de la classe XML. Consultez l'entrée *Classe XMLNode* dans le Chapitre 12, *Dictionnaire ActionScript*, page 215.
XMLSocket	La classe XMLSocket vous permet de créer une connexion socket durable avec un autre ordinateur pour des transferts de données à faible temps d'attente, comme ceux que requièrent les applications de dialogue en ligne en temps réel. Consultez *Utilisation de la classe XMLSocket*, page 195 et l'entrée *Classe XMLSocket* dans le Chapitre 12, *Dictionnaire ActionScript*, page 215.

Classes Authoring

Les classes Authoring sont uniquement disponibles dans l'environnement de programmation Flash. Ces classes résident dans le sous-dossier Classes intégrées > Programmation de la boîte à outils Actions.

Classe	Description
CustomActionsg	La classe CustomActions vous permet de gérer toute action personnalisée enregistrée dans l'outil de programmation. Consultez l'entrée *Classe CustomActions* dans le Chapitre 12, *Dictionnaire ActionScript*, page 215.
Aperçu en direct	La fonction Aperçu en direct (qui, bien qu'elle ne soit pas une classe, est répertoriée dans la catégorie Classes intégrées de la boîte d'outils Actions) propose une fonction unique, `onUpdate`, réservée aux développeurs de composants. Consultez `onUpdate` dans le Chapitre 12, *Dictionnaire ActionScript*, page 215.

CHAPITRE 7
Utilisation des clips

Les clips sont des fichiers SWF miniatures autonomes exécutés indépendamment les uns des autres et du scénario les contenant. Par exemple, si le scénario principal contient une seule image et qu'un clip contenu dans cette image comporte 10 images, chaque image du clip est lue lorsque vous lisez le fichier SWF principal. Un clip peut lui-même contenir d'autres clips, ou *clips imbriqués*. Les clips imbriqués de cette manière sont organisés hiérarchiquement : le *clip parent* contient un ou plusieurs *clips enfants*.

Chaque occurrence de clip a un nom, appelé *nom d'occurrence*, qui l'identifie de façon unique en tant qu'objet pouvant être contrôlé à l'aide d'ActionScript. De façon plus spécifique, le nom d'occurrence identifie l'occurrence comme un objet du type de classe MovieClip. Utilisez les propriétés et méthodes de la classe MovieClip pour contrôler l'apparence et le comportement des clips à l'exécution.

Vous pouvez considérer les clips comme des objets autonomes qui répondent à des événements, envoient des messages à d'autres objets de clip, conservent leur état et gèrent leurs clips enfants. Les clips constituent ainsi la base de *l'architecture basée sur les composants* de Macromedia Flash MX 2004 et Macromedia Flash MX Professionnel 2004. Les composants disponibles dans le panneau Composants (Fenêtre > Panneaux de développement > Composants) sont en fait des clips sophistiqués qui ont été conçus et programmés pour apparaître et se comporter d'une certaine manière. Pour plus d'informations sur la création de composants, consultez le guide *Utilisation des composants*.

A propos du contrôle des clips à l'aide d'ActionScript

Vous pouvez utiliser les fonctions ActionScript globales ou les méthodes de la classe MovieClip pour accomplir des tâches sur les clips. Certaines méthodes de MovieClip accomplissent les mêmes tâches que les fonctions du même nom, alors que d'autres méthodes MovieClip, comme `hitTest()` et `swapDepths()`, ne possèdent pas de noms de fonction correspondants.

L'exemple suivant illustre la différence entre l'emploi d'une méthode et d'une fonction. Les deux instructions dupliquent l'occurrence `mon_mc`, nomment le nouveau clip `nouveauClip` et le placent à une profondeur de 5.

```
mon_mc.duplicateMovieClip("nouveauClip", 5);
duplicateMovieClip("mon_mc", "nouveauClip", 5);
```

Lorsqu'une fonction et une méthode présentent des comportements similaires, vous pouvez choisir l'une ou l'autre pour contrôler des clips. Le choix dépend de vos préférences et de votre familiarité avec la rédaction de scripts dans ActionScript. Que vous utilisiez une fonction ou une méthode, le scénario cible doit être chargé dans Flash Player lorsque la fonction ou la méthode est appelée.

Pour utiliser une méthode, vous l'invoquez en utilisant le chemin cible du nom d'occurrence, suivi d'un point, puis du nom et des paramètres de la méthode, comme dans l'exemple suivant :

```
monClip.play();
clipParent.clipEnfant.gotoAndPlay(3);
```

Dans la première instruction, la méthode `play()` place la tête de lecture dans l'occurrence `monClip`. Dans la deuxième instruction, la méthode `gotoAndPlay()` envoie la tête de lecture dans `clipEnfant` (qui est un enfant de l'occurrence `clipParent`) à l'image 3 et continue à déplacer la tête de lecture.

Les fonctions globales qui contrôlent un scénario possèdent un paramètre *`cible`* qui permet de définir le chemin cible de l'occurrence que vous voulez contrôler. Par exemple, dans le script suivant, `startDrag()` cible l'occurrence `curseurPerso` et la rend déplaçable :

```
on (press) {
  startDrag("curseurPerso");
}
```

Les fonctions suivantes ciblent des clips : `loadMovie()`, `unloadMovie()`, `loadVariables()`, `setProperty()`, `startDrag()`, `duplicateMovieClip()` et `removeMovieClip()`. Pour utiliser ces fonctions, vous devez entrer un chemin cible dans le paramètre *`cible`* de la fonction pour indiquer la cible de la fonction.

Les méthodes MovieClip suivantes peuvent contrôler des clips ou des niveaux chargés et n'ont pas de fonctions équivalentes : `MovieClip.attachMovie()`, `MovieClip.createEmptyMovieClip()`, `MovieClip.createTextField()`, `MovieClip.getBounds`, `MovieClip.getBytesLoaded()`, `MovieClip.getBytesTotal()`, `MovieClip.getDepth()`, `MovieClip.getInstanceAtDepth()`, `MovieClip.getNextHighestDepth()`, `MovieClip.globalToLocal()`, `MovieClip.localToGlobal()`, `MovieClip.hitTest()`, `MovieClip.setMask()`, `MovieClip.swapDepths()`.

Pour plus d'informations sur ces fonctions et méthodes, consultez le Chapitre 12, *Dictionnaire ActionScript*, page 215.

Appel de plusieurs méthodes sur un seul clip

Vous pouvez utiliser l'instruction `with` pour appeler un clip une seule fois, puis exécuter une série de méthodes sur ce clip. L'instruction `with` fonctionne sur tous les objets ActionScript (tels que Array, Color et Sound), et non seulement sur les clips.

L'instruction `with` prend un objet comme paramètre. L'objet que vous spécifiez est ajouté à la fin du chemin cible courant. Toutes les actions imbriquées dans une instruction `with` sont exécutées à l'intérieur du nouveau chemin cible. Par exemple, dans le script suivant, l'instruction `with` est transmise à l'objet `beignet.confiture` pour changer les propriétés de `confiture` :

```
with (beignet.confiture){
  _alpha = 20;
  _xscale = 150;
  _yscale = 150;
}
```

Le script se comporte comme si les instructions dans l'instruction `with` étaient appelées depuis le scénario de l'occurrence `confiture`. Le code ci-dessus est équivalent au code suivant :

```
beignet.confiture._alpha = 20;
beignet.confiture._xscale = 150;
beignet.confiture._yscale = 150;
```

Le code ci-dessus est également équivalent au code suivant :

```
with (beignet){
  confiture._alpha = 20;
  confiture._xscale = 150;
  confiture._yscale = 150;
}
```

Chargement et déchargement de fichiers SWF supplémentaires

Pour lire d'autres fichiers SWF sans fermer Flash Player, ou pour passer d'un fichier SWF à l'autre sans charger une autre page HTML, vous pouvez utiliser la fonction globale `loadMovie()` ou la méthode `loadMovie()` de la classe MovieClip. Vous pouvez également utiliser `loadMovie()` pour envoyer des variables à un script CGI, qui génère un fichier SWF en tant que fichier CGI. Lorsque vous chargez un fichier SWF, vous pouvez spécifier comme cible un niveau ou un clip, dans lequel sera chargé le fichier SWF. Si vous chargez un fichier SWF dans une cible, le fichier SWF chargé hérite des propriétés du clip ciblé. Une fois l'animation chargée, vous pouvez modifier les propriétés.

La méthode `unloadMovie()` supprime un fichier SWF précédemment chargé par `loadMovie()`. En purgeant explicitement les fichiers SWF avec `unloadMovie()`, vous assurez une transition fluide entre les fichiers SWF et vous pouvez alléger la quantité de mémoire requise par Flash Player.

Utilisez `loadMovie()` pour effectuer les opérations suivantes :

- Lire une séquence de bandeaux publicitaires sous forme de fichiers SWF, en plaçant une fonction `loadMovie()` à la fin de chaque fichier SWF pour charger le fichier SWF suivant.
- Développer une interface arborescente permettant à l'utilisateur de choisir parmi différents fichiers SWF.
- Construire une interface de navigation avec des contrôles de navigation dans le niveau 0 chargeant d'autres niveaux. Le chargement de niveaux produit une transition plus douce que le chargement de nouvelles pages HTML dans un navigateur.

Pour plus d'informations sur le chargement d'animations, consultez *Chargement de fichiers SWF et JPEG externes*, page 206.

Spécification d'un scénario racine pour les fichiers SWF chargés

La propriété ActionScript _root spécifie ou renvoie une référence au scénario racine d'un fichier SWF. Si un fichier SWF possède plusieurs niveaux, le scénario 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. Cependant, le scénario spécifié par _root peut changer si le fichier SWF est exécuté de façon indépendante (à son propre niveau) ou s'il a été chargé dans une occurrence de clip par un appel loadMovie().

Prenez par exemple un fichier nommé conteneur.swf qui possède une occurrence de clip nommée cible_mc dans son scénario principal. Le fichier conteneur.swf déclare une variable nommée nomDutilisateur dans son scénario principal. Le même script charge ensuite un autre fichier nommé contenu.swf dans l'occurrence de clip cible_mc.

```
// Dans conteneur.swf:
_root.nomDutilisateur = "Tom";
cible_mc.loadMovie("contenu.swf");
```

Le fichier SWF chargé, contenu.swf, déclare également une variable nommée nomDutilisateur dans son scénario racine.

```
// Dans contenu.swf:
_root.nomDutilisateur = "Marie";
```

Lorsque contenu.swf est chargé dans le clip dans conteneur.swf, la valeur de nomDutilisateur associée au scénario racine du fichier SWF hôte (conteneur.swf) prend la valeur « Marie ». Ceci peut entraîner le mauvais fonctionnement du code dans conteneur.swf (ainsi que dans contenu.swf).

Pour obliger _root à évaluer systématiquement le scénario du fichier SWF chargé, et non le scénario racine réel, utilisez la propriété _lockroot. Cette propriété peut être définie par le fichier SWF en cours de chargement ou par le fichier SWF chargé. Lorsque _lockroot est défini sur true sur une occurrence de clip, le clip agira comme _root pour tout fichier SWF qui y sera chargé. Lorsque _lockroot est défini sur true au sein d'un fichier SWF, le fichier SWF en question agira comme sa propre racine, quel que soit l'autre fichier SWF effectuant le chargement. N'importe quel clip, et n'importe quel nombre de clips, peut définir _lockroot sur true. Cette propriété est false par défaut.

L'auteur de conteneur.swf peut par exemple associer le code suivant au clip cible_mc :

```
// Associé au clip cible_mc :
onClipEvent(load) {
  this._lockroot = true;
}
```

Ceci garantit que les références à _root dans contenu.swf (ou dans n'importe quel fichier SWF chargé dans cible_mc) feront référence à leur propre scénario, non au scénario racine réel de conteneur.swf.

De la même façon, l'auteur de contenu.swf peut ajouter le code suivant à son scénario principal.

```
// Dans contenu.swf :
this._lockroot = true;
```

Ceci garantit que, quel que soit l'endroit où contenu.swf est chargé, toute référence à _root fera référence à son propre scénario principal et non à celui du fichier SWF hôte.

Pour plus d'informations, consultez MovieClip._lockroot, page 580.

Chargement de fichiers JPEG dans des clips

Vous pouvez utiliser la fonction `loadMovie()` ou la méthode MovieClip sur un même nom pour charger des fichiers image JPEG dans une occurrence de clip. Vous pouvez également utiliser la fonction `loadMovieNum()` pour charger un fichier JPEG dans un niveau.

Lorsque vous chargez une image dans un clip, son coin supérieur gauche est placé au point d'alignement du clip. Ce point se trouvant souvent au centre du clip, il se peut que l'image chargée ne soit pas centrée. De même, lorsque vous chargez une image dans un scénario principal, son coin supérieur gauche est placé dans le coin supérieur gauche de la scène. L'image chargée hérite de la rotation et de l'échelle du clip, mais le contenu initial du clip est supprimé.

Pour plus d'informations, consultez *Chargement de fichiers SWF et JPEG externes*, page 206, `loadMovie()`, page 468, `MovieClip.loadMovie`, page 577 et `loadMovieNum()`, page 469.

Modification de la position et de l'apparence d'un clip

Pour modifier les propriétés d'un clip pendant sa lecture, vous pouvez rédiger une instruction affectant une valeur à la propriété ou utiliser la fonction `setProperty()`. Par exemple, le code suivant fixe la rotation de l'occurrence `mc` à 45 :

```
mc._rotation = 45;
```

Cela équivaut au code suivant, qui utilise la fonction `setProperty()` :

```
setProperty("mc", _rotation, 45);
```

Certaines propriétés, appelées *propriétés en lecture seule*, ont des valeurs que vous pouvez lire mais pas définir. Ces propriétés sont identifiées comme telles dans le dictionnaire ActionScript. Les propriétés suivantes sont en lecture seule : `_currentframe`, `_droptarget`, `_framesloaded`, `_parent`, `_target`, `_totalframes`, `_url`, `_xmouse` et `_ymouse`.

Vous pouvez rédiger des instructions pour définir des propriétés qui ne sont pas en lecture seule. L'instruction suivante définit la propriété `_alpha` de l'occurrence de clip `roue`, qui est un enfant de l'occurrence `voiture` :

```
voiture.roue._alpha = 50;
```

En outre, vous pouvez rédiger des instructions qui récupèrent la valeur d'une propriété de clip. Par exemple, l'instruction suivante récupère la valeur de la propriété `_xmouse` dans le scénario du niveau actuel et donne cette valeur à la propriété `x` de l'occurrence `curseurPerso` :

```
onClipEvent(enterFrame){
  curseurPerso._x = _root._xmouse;
}
```

Cela équivaut au code suivant, qui utilise la fonction `getProperty()` :

```
onClipEvent(enterFrame){
  curseurPerso._x = getProperty(_root, _xmouse);
}
```

Les propriétés `_x`, `_y`, `_rotation`, `_xscale`, `_yscale`, `_height`, `_width`, `_alpha` et `_visible` sont affectées par les transformations effectuées sur le parent du clip et transforment le clip et tous ses enfants. Les propriétés `_focusrect`, `_highquality`, `_quality` et `_soundbuftime` sont globales et appartiennent uniquement au scénario de niveau 0. Toutes les autres propriétés appartiennent à chaque clip ou niveau chargé.

Pour obtenir la liste des propriétés de clips, consultez *Propriétés de la classe MovieClip*, page 546.

Déplacement des clips

Vous pouvez utiliser la fonction globale `startDrag()` ou la méthode `MovieClip.startDrag()` pour rendre un clip déplaçable. Par exemple, vous pouvez créer un clip pouvant être déplacé pour les jeux, les fonctions glisser-déposer, les interfaces personnalisables, les barres de défilement et les curseurs de défilement.

Un clip reste déplaçable jusqu'à sont arrêt explicite par `stopDrag()` ou jusqu'à ce qu'un autre clip soit ciblé avec `startDrag()`. Vous ne pouvez déplacer qu'un seul clip à la fois.

Pour créer des comportements plus complexes avec les opérations glisser-déposer, vous pouvez évaluer la propriété `_droptarget` du clip en cours de déplacement. Par exemple, vous pouvez examiner la propriété `_droptarget` pour voir si le clip a été déplacé vers un clip spécifique (tel qu'un clip « poubelle »), puis déclencher une autre action. Pour plus d'informations, consultez `startDrag()`, page 734 ou `MovieClip.startDrag()`, page 604.

Création de clips à l'exécution

Vous pouvez créer des occurrences de clip dans l'environnement auteur de Flash, mais également à l'exécution. ActionScript propose trois manières de créer des clips à l'exécution :

- en créant une nouvelle occurrence de clip vide ;
- en dupliquant une occurrence de clip existante ;
- en associant une occurrence de symbole de bibliothèque de clip à la scène.

Chaque occurrence de clip créée à l'exécution doit avoir un nom d'occurrence et une valeur de profondeur (ordre d'empilement ou ordre *z*). La profondeur que vous spécifiez détermine la façon dont le nouveau clip recouvre les autres clips sur le même scénario. Pour plus d'informations, consultez *Gestion des profondeurs de clip*, page 135.

Création d'un clip vide

Pour créer un clip vide sur la scène, utilisez la méthode `createEmptyMovieClip()` de la classe MovieClip. Cette méthode crée un clip en tant qu'enfant du clip qui l'a appelée. Le point d'alignement d'un clip vide nouvellement créé se situe dans le coin supérieur gauche.

Par exemple, le code suivant permet de créer un nouveau clip enfant nommé `nouveau_mc` à une profondeur de 10 dans le clip nommé `parent_mc`.

```
parent_mc.createEmptyMovieClip("nouveau_mc", 10);
```

Le code suivant permet de créer un nouveau clip nommé `image_mc` sur le scénario racine du fichier SWF dans lequel le script est exécuté, puis d'appeler `loadMovie()` pour y charger un fichier JPEG externe.

```
_root.createEmptyMovieClip("canevas_mc", 10);
canevas_mc.loadMovie("fleurs.jpg");
```

Pour plus d'informations, consultez `MovieClip.createEmptyMovieClip()`, page 555.

Duplication ou suppression d'un clip

Pour dupliquer ou supprimer des occurrences de clip, utilisez les fonctions globales `duplicateMovieClip()` ou `removeMovieClip()`, ou encore les méthodes de la classe MovieClip du même nom. La méthode `duplicateMovieClip()` crée une nouvelle occurrence d'une occurrence de clip existante, lui affecte un nouveau nom d'occurrence et lui donne une profondeur, ou ordre *z*. Un clip dupliqué commence toujours à l'image 1 même si le clip initial se trouvait dans une autre image lors de la duplication, et il se situe toujours à la tête de tous les clips prédéfinis placés dans le scénario.

Pour supprimer un clip que vous avez créé avec `duplicateMovieClip()`, utilisez `removeMovieClip()`. Un clip dupliqué est également supprimé si le clip parent est supprimé.

Pour plus d'informations, consultez `duplicateMovieClip()`, page 411 et `removeMovieClip()`, page 686.

Association d'un symbole de clip à la scène

La dernière manière de créer des occurrences de clip à l'exécution est d'utiliser `attachMovie()`. La méthode `attachMovie()` associe une occurrence de symbole de clip comprise dans la bibliothèque du fichier SWF à la scène. Le nouveau clip devient un clip enfant du clip l'ayant associé.

Pour utiliser ActionScript pour associer un symbole de clip à partir de la bibliothèque, vous devez exporter le symbole pour ActionScript et lui affecter un identifiant de liaison unique. Pour ce faire, utilisez la boîte de dialogue Propriétés de liaison.

Par défaut, tous les clips exportés pour être utilisés avec ActionScript sont chargés avant la première image du fichier SWF les contenant. Ce chargement peut entraîner un retard avant la lecture de la première image. Lorsque vous affectez un identifiant de liaison à un élément, vous pouvez également spécifier si cet élément doit être ajouté avant la première image. S'il ne l'est pas, vous devez en inclure une occurrence sur une autre image du fichier SWF, sinon l'élément ne sera pas exporté dans le fichier SWF.

Pour affecter un identifiant de liaison à un clip :

1. Choisissez Fenêtre > Bibliothèque pour ouvrir le panneau Bibliothèque.
2. Sélectionnez un clip dans le panneau Bibliothèque.
3. Dans le panneau Bibliothèque, choisissez Liaison dans le menu d'options.

   La boîte de dialogue Propriétés de liaison apparaît.
4. Pour Liaison, activez l'option Exporter pour ActionScript.
5. Pour Identifiant, entrez l'identifiant du clip.

   Par défaut, l'identifiant est identique au nom du symbole.
6. Vous pouvez également affecter une classe ActionScript 2.0 au symbole de clip. Pour plus d'informations, consultez *Affectation d'une classe à un symbole de clip*, page 138.
7. Si vous ne voulez pas que le clip soit chargé avant la première image, désactivez l'option Exporter dans la première image.

   Si vous désactivez cette option, placez une occurrence du clip sur l'image du scénario où vous souhaitez qu'elle soit disponible. Par exemple, si le script que vous écrivez ne fait pas référence au clip avant l'image 10, placez une occurrence du symbole à cette image ou juste avant celle-ci dans le scénario.

8 Cliquez sur OK.

Une fois que vous avez affecté un identifiant de liaison à un clip, vous pouvez associer une occurrence du symbole à la scène au moment de l'exécution à l'aide de la méthode `attachMovie()`.

Pour associer un clip à un autre clip :

1 Affectez un identifiant de liaison à un symbole de bibliothèque de clip comme décrit ci-dessus.

2 Le panneau Actions étant ouvert (Fenêtre > Panneaux de développement > Actions), sélectionnez une image dans le scénario.

3 Dans la fenêtre de script du panneau Actions, tapez le nom du clip ou du niveau auquel vous souhaitez associer le nouveau clip. Par exemple, pour associer le clip au scénario principal, tapez `_root`.

4 Dans la boîte à outils Actions (du côté gauche du panneau Actions), cliquez sur la catégorie Classes intégrées, puis sur Animation, puis sur MovieClip, puis sur Méthodes et double-cliquez sur `attachMovie()`.

5 A l'aide des conseils de code qui apparaissent comme un guide, entrez les valeurs des paramètres suivants :

- Pour `nomID`, spécifiez l'identifiant que vous avez saisi dans la boîte de dialogue Propriétés de liaison.
- Pour `nouveauNom`, entrez un nom d'occurrence pour le clip associé afin de pouvoir le cibler.
- Pour `profondeur`, entrez le niveau dans lequel l'animation dupliquée sera associée au clip. Chaque animation associée a un ordre d'empilement qui lui est propre, le niveau 0 étant le niveau de l'animation d'origine. Les clips associés sont toujours au-dessus du clip d'origine. Voici un exemple :

```
monClip.attachMovie("calif", "california", 10);
```

Pour plus d'informations, consultez `MovieClip.attachMovie()`, page 550.

Ajout de paramètres aux clips créés dynamiquement

Lorsque vous créez ou dupliquez dynamiquement un clip à l'aide de `MovieClip.attachMovie()` et `MovieClip.duplicateMovie()`, vous pouvez remplir le clip avec des paramètres provenant d'un autre objet. Le paramètre *`objetInit`* de `attachMovie()` et `duplicateMovie()` permet aux clips créés dynamiquement de recevoir des paramètres de clip. Le paramètre *`objetInit`* est facultatif.

Pour plus d'informations, consultez `MovieClip.attachMovie()`, page 550 et `MovieClip.duplicateMovieClip`, page 560.

Pour remplir un clip créé dynamiquement avec des paramètres provenant d'un objet spécifié, effectuez l'une des opérations suivantes :

- Utilisez la syntaxe suivante avec `attachMovie()` :

```
monClip.attachMovie(nomIdentifiant, nouveauNom, profondeur [, objetInit])
```

- Utilisez la syntaxe suivante avec `duplicateMovie()` :

```
monClip.duplicateMovie(nomIdentifiant, nouveauNom, profondeur [, objetInit])
```

Le paramètre *`objetInit`* spécifie le nom de l'objet dont vous souhaitez utiliser les paramètres pour remplir le clip créé dynamiquement.

Pour remplir un clip avec des paramètres en utilisant `attachMovie()` :

1. Dans un nouveau document Flash, créez un symbole de clip en choisissant Insertion > Nouveau symbole. Tapez `dynamique` dans la zone de texte Nom du symbole et sélectionnez le comportement de clip.
2. A l'intérieur du symbole, créez une zone de texte dynamique sur la scène avec `nom_txt` comme nom d'occurrence.
3. Sélectionnez la première image du scénario du clip et ouvrez le panneau Actions (Fenêtre > Panneaux de développement > Actions).
4. Créez une nouvelle variable nommée `nom` et affectez sa valeur à la propriété `text` de `nom_txt`, comme suit :

```
var nom:String;
nom_txt.text = nom;
```

5. Choisissez Edition > Modifier le document pour revenir au scénario principal.
6. Sélectionnez le symbole de clip dans la bibliothèque et choisissez Propriétés de liaison dans le menu D'options du panneau Bibliothèque.

   La boîte de dialogue Propriétés de liaison apparaît.
7. Choisissez l'option Exporter pour ActionScript et cliquez sur OK.
8. Sélectionnez la première image du scénario principal et ajoutez le code suivant à la fenêtre de script du panneau Actions :

```
_root.attachMovie("dynamique", "nomNouveauClip", 10, {name:"Eric"});
```

9. Testez l'animation (Contrôle > Tester l'animation). Le nom que vous avez spécifié dans l'appel `attachMovie()` apparaît dans le champ de texte du nouveau clip.

Gestion des profondeurs de clip

Chaque clip possède son propre ordre *z* qui détermine la façon dont les objets se recouvrent à l'intérieur du fichier SWF parent ou du clip. A chaque clip est associée une valeur de profondeur, qui détermine s'il sera rendu devant ou derrière les autres clips dans le même scénario de clip. Lorsque vous créez un clip à l'exécution à l'aide de `MovieClip.attachMovie()`, `MovieClip.duplicateMovieClip` ou `MovieClip.createEmptyMovieClip()`, vous devez systématiquement spécifier une profondeur pour le nouveau clip comme paramètre de méthode. Par exemple, le code suivant associe un nouveau clip au scénario d'un clip nommé `conteneur_mc` avec une valeur de profondeur de 10.

```
conteneur_mc.attachMovie("symbolID", "clip_1", 10);
```

Ceci crée un nouveau clip avec une profondeur de 10 dans l'ordre *z* de `conteneur_mc`.

Par exemple, le code suivant associe deux nouveaux clips à `conteneur_mc`. Le premier clip, nommé `clip_1`, sera rendu derrière `clip_2`, car une valeur de profondeur inférieure lui a été attribuée.

```
conteneur_mc.attachMovie("symbolID", "clip_1", 10);
container_mc.attachMovie("symbolID", "clip_2", 15);
```

La valeur de profondeur pour les clips peut aller de -16384 à 1048575.

La classe MovieClip fournit plusieurs méthodes permettant de gérer les profondeurs de clip : Consultez `MovieClip.getNextHighestDepth()`, page 567, `MovieClip.getInstanceAtDepth()`, page 566, `MovieClip.getDepth()`, page 566 et `MovieClip.swapDepths()`, page 606.

Définition de la prochaine profondeur maximale disponible

Pour déterminer quelle est la prochaine profondeur maximale disponible dans un clip, utilisez `MovieClip.getNextHighestDepth()`. La valeur entière renvoyée par cette méthode indique la prochaine profondeur disponible qui sera rendue devant tous les autres objets dans le clip.

Le code suivant crée un nouveau clip, avec une valeur de profondeur de 10, sur le scénario du clip nommé `menus_mc`. Il détermine ensuite la prochaine profondeur maximale disponible dans le même clip, puis crée un nouveau clip à cette profondeur.

```
menus_mc.attachMovie("menuClip","file_menu", 10);
var prochaineProfondeur = menus_mc.getNextHighestDepth();
menus_mc.attachMovie("menuClip", "edit_menu", prochaineProfondeur);
```

Dans ce cas, la variable nommée `prochaineProfondeur` contient la valeur 11, étant donné qu'il s'agit de la prochaine profondeur maximale disponible pour le clip `menus_mc`.

Pour obtenir la profondeur maximale actuellement occupée, soustrayez 1 à la valeur renvoyée par `getNextHighestDepth()`, comme illustré dans la section suivante (consultez *Définition de l'occurrence à une profondeur spécifique*, page 136).

Définition de l'occurrence à une profondeur spécifique

Pour définir l'occurrence à une profondeur spécifique, utilisez `MovieClip.getInstanceAtDepth()`. Cette méthode renvoie une référence à l'occurrence de MovieClip se trouvant à la profondeur indiquée.

Le code suivant combine `getNextHighestDepth()` et `getInstanceAtDepth()` pour déterminer le clip se trouvant à la plus grande profondeur occupée du scénario racine.

```
var profMaxOccupée = _root.getNextHighestDepth() - 1;
var occurrenceAProfMax = _root.getInstanceAtDepth(profMaxOccupée);
```

Pour plus d'informations, consultez `MovieClip.getInstanceAtDepth()`, page 566.

Définition de la profondeur d'une occurrence

Pour déterminer la profondeur d'une occurrence de clip, utilisez `MovieClip.getDepth()`.

Le code suivant itère sur tous les clips du scénario principal d'un fichier SWF et affiche le nom d'occurrence et la valeur de profondeur de chaque clip dans le panneau de sortie.

```
for(each in _root) {
  var obj = _root[each];
  if(obj instanceof MovieClip) {
    var profObjet = obj.getDepth();
    trace(obj._name + ":" + profObjet)
  }
}
```

Pour plus d'informations, consultez `MovieClip.getDepth()`, page 566.

Permutation de profondeurs de clips

Pour permuter les profondeurs de deux clips sur un même scénario, utilisez `MovieClip.swapDepths()`. Pour plus d'informations, consultez `MovieClip.swapDepths()`, page 606.

Dessin de formes avec ActionScript

Vous pouvez utiliser des méthodes de la classe MovieClip pour dessiner des lignes et les remplissages sur la scène. Vous pouvez ainsi créer des outils de dessin pour les utilisateurs et tracer des formes dans l'animation en réponse à des événements. Les méthodes de dessin sont `beginFill()`, `beginGradientFill()`, `clear()`, `curveTo()`, `endFill()`, `lineTo()`, `lineStyle()` et `moveTo()`.

Vous pouvez utiliser les méthodes de dessin avec n'importe quel clip. Toutefois, si vous les utilisez dans un clip créé en mode auteur, les méthodes de dessin sont exécutées avant que le clip ne soit dessiné. En d'autres termes, le contenu créé en mode auteur est dessiné au-dessus du contenu dessiné à l'aide des méthodes de dessin.

Vous pouvez utiliser des clips comportant des méthodes de dessin en tant que masques. Toutefois, comme pour tous les masques de clips, les traits seront ignorés.

Pour dessiner une forme :

1 Utilisez `createEmptyMovieClip()` pour créer un clip vide sur la scène.

Le nouveau clip est l'enfant d'un clip existant ou du scénario principal, comme dans l'exemple suivant :

```
_root.createEmptyMovieClip ("triangle", 1);
```

2 Utilisez le clip vide pour appeler les méthodes de dessin.

L'exemple suivant trace un triangle comportant des lignes magenta d'une épaisseur de 5 points et aucun remplissage :

```
with (_root.triangle) {
     lineStyle (5, 0xff00ff, 100);
     moveTo (200, 200);
     lineTo (300, 300);
     lineTo (100, 300);
     lineTo (200, 200);
}
```

Pour plus d'informations sur ces méthodes, consultez les entrées correspondantes dans le Chapitre 12, *Dictionnaire ActionScript*, page 215.

Utilisation de clips comme masques

Vous pouvez utiliser un clip comme masque pour créer un trou qui laisse apparaître le contenu d'un autre clip. Le clip utilisé comme masque lit toutes les images de son scénario, comme un clip ordinaire. Vous pouvez rendre le clip déplaçable, l'animer le long d'un guide de mouvement, utiliser des formes distinctes dans un même masque, ou redimensionner un masque de façon dynamique. Vous pouvez également utiliser ActionScript pour activer et désactiver un masque.

Il est impossible d'utiliser un masque pour en masquer un autre. Il est impossible de définir la propriété `_alpha` d'un clip utilisé comme masque. Seuls les remplissages sont utilisés dans un clip utilisé comme masque, les traits étant ignorés.

Pour créer un masque :

1 Sur la scène, sélectionnez un clip à masquer.

2 Dans l'inspecteur des propriétés, entrez un nom d'occurrence pour le clip, tel que `image`.

3 Créez un clip devant être un masque. Donnez-lui un nom d'occurrence dans l'inspecteur des propriétés, tel que `masque`.

Le clip masqué sera révélé en dessous de chaque zone opaque (non transparente) du clip agissant comme masque.

4 Sélectionnez l'image 1 dans le scénario.

5 Ouvrez le panneau Actions (Fenêtre > Panneaux de développement > Actions) si ce dernier n'est pas déjà ouvert.

6 Dans le panneau Actions, entrez le code suivant :

```
image.setMask(masque);
```

Pour plus d'informations, consultez `MovieClip.setMask()`, page 603.

A propos du masquage des polices de périphérique

Vous pouvez utiliser un clip pour masquer le texte défini dans une police de périphérique. Pour qu'un masque de clip fonctionne sur une police de périphérique, l'utilisateur doit disposer de Flash Player 6 version 40 ou ultérieure.

Lorsque vous utilisez un clip pour masquer le texte défini dans une police de périphérique, le cadre de délimitation rectangulaire du masque est utilisé comme forme de masque. Ainsi, si vous créez un masque de clip non rectangulaire pour du texte de police de périphérique dans un environnement auteur Flash, le masque qui apparaît dans le fichier SWF prend la forme du cadre de délimitation rectangulaire et non la forme du masque lui-même.

Vous pouvez uniquement masquer des polices de périphérique en utilisant un clip comme masque. Vous ne pouvez pas masquer des polices de périphérique en utilisant un calque de masque sur la scène.

Gestion d'événements de clip

Les clips peuvent répondre à des événements utilisateur, tels que des clics de souris ou des pressions sur des touches, ainsi qu'à des événements de niveau système, tel que le chargement initial d'un clip sur la scène. ActionScript fournit deux façons de gérer les événements de clip : via les méthodes de gestionnaire d'événement et via les gestionnaires d'événement `onClipEvent()` et `on()`. Pour plus d'informations, consultez le Chapitre 4, *Gestion d'événements*, page 89.

Affectation d'une classe à un symbole de clip

ActionScript 2.0 vous permet de créer votre propre classe qui étend le comportement de la classe intégrée MovieClip puis affecte cette classe à un symbole de bibliothèque de clip à l'aide de la boîte de dialogue Propriétés de liaison. Lorsque vous créez une occurrence du clip auquel la classe est affectée, elle assume les propriétés et comportements définis par la classe qui lui est affectée. (Pour plus d'informations sur ActionScript 2,0, consultez le Chapitre 9, *Création de classes avec ActionScript 2.0*, page 163.)

Dans une sous-classe de la classe MovieClip, vous pouvez fournir les définitions des méthodes et gestionnaires d'événements intégrés MovieClip, tels que `onEnterFrame` et `onRelease`. Dans la procédure suivante, vous créez une classe intitulée MoveRight qui étend la classe MovieClip et définit un gestionnaire `onPress` qui déplace le clip de 20 pixels vers la droite chaque fois que l'utilisateur clique sur le clip. Dans la deuxième procédure, vous créez un symbole de clip dans un nouveau document Flash (FLA) et affectez la classe MoveRight à ce symbole.

Pour créer une sous-classe de clip :

1 Créez un nouveau répertoire nommé TestBalle.

2 Créez un nouveau fichier ActionScript en effectuant l'une des opérations suivantes :

- (Flash MX Professionnel 2004) Choisissez Fichier > Nouveau et sélectionnez Fichier ActionScript dans la liste des types de documents.
- (Flash MX 2004) Créez une nouveau fichier texte dans l'éditeur de texte de votre choix.

3 Entrez le code suivant dans votre script :

```
// Classe MoveRight -- déplace le clip de cinq pixels vers la droite à chaque
   image
class MoveRight extends MovieClip  {
  function onPress() {
    this._x += 20;
  }
}
```

4 Enregistrez le document sous MoveRight.as dans le répertoire TestBalle.

Pour affecter la classe à un symbole de clip :

1 Dans Flash, choisissez Fichier > Nouveau, sélectionnez Document Flash dans la liste des types de fichiers et cliquez sur OK.

2 A l'aide de l'outil Ovale, dessinez un cercle sur la scène.

3 Sélectionnez le cercle puis choisissez Modification > Convertir en symbole. Dans la boîte de dialogue Convertir en symbole, choisissez le comportement de symbole Clip et entrez `Balle` dans le champ Nom.

4 Ouvrez le panneau Bibliothèque (Fenêtre > Bibliothèque) et sélectionnez le symbole Balle.

5 Sélectionnez Liaison dans le menu d'options du panneau Bibliothèque pour ouvrir la boîte de dialogue Propriétés de liaison.

6 Dans la boîte de dialogue Propriétés de liaison, sélectionnez l'option Exporter pour ActionScript et tapez `MoveRight` dans le champ Classe AS 2.0. Cliquez sur OK.

7 Enregistrez le fichier sous Balle.fla dans le répertoire TestBalle (le répertoire contenant le fichier MoveRight.as).

8 Testez l'animation (Contrôle > Tester l'animation).

Chaque fois que vous cliquez sur le clip balle, il se déplace de 20 pixels sur la droite.

Initialisation de propriétés de classe

Dans l'exemple présenté précédemment, vous avez ajouté l'occurrence du symbole Balle manuellement, c'est-à-dire lors de la programmation. Comme expliqué précédemment (consultez *Ajout de paramètres aux clips créés dynamiquement*, page 134), vous pouvez affecter des paramètres à des clips que vous créez lors de l'exécution en utilisant le paramètre `objetInit` de `attachMovie()` et `duplicateMovie()`. Vous pouvez utiliser cette fonction pour initialiser des propriétés de la classe que vous affectez à un clip.

Par exemple, la classe suivante nommée MoveRightDistance est une variante de la classe MoveRight présentée plus tôt (consultez *Affectation d'une classe à un symbole de clip*, page 138). La différence est une nouvelle propriété nommée `distance`, dont la valeur détermine le nombre de pixels dont un clip se déplace chaque fois que vous cliquez dessus.

```
// Classe MoveRightDistance -- déplace le clip de cinq pixels vers la droite à
   chaque image
la classe MoveRightDistance étend MovieClip {
   // la propriété distance détermine le nombre de
   // pixels dont doit être déplacé le clip à chaque clic de souris
   var distance:Number;
   function onPress() {
      this._x += distance;
   }
}
```

En considérant que cette classe est affectée à un symbole avec un identifiant de liaison de Balle, le code suivant crée deux nouvelles occurrences du symbole sur le scénario racine du fichier SWF. La première occurrence, nommée `balle_50`, se déplace de 50 pixels chaque fois que vous cliquez dessus, la deuxième, nommée `balle_125`, se déplace de 125 pixels chaque fois que vous cliquez dessus.

```
_root.attachMovie("Balle", "balle_50", 10, {distance:50});
_root.attachMovie("Balle", "balle_125", 20, {distance:125});
```

CHAPITRE 8
Utilisation du texte

Un champ de texte dynamique ou de saisie est un objet TextField (une occurrence de la classe TextField). Lorsque vous créez un champ de texte, vous pouvez lui affecter un nom d'occurrence dans l'inspecteur des propriétés. Vous pouvez utiliser ce nom d'occurrence dans les instructions ActionScript pour définir, modifier et formater le champ de texte et son contenu à l'aide des classes TextField et TextFormat.

Les méthodes de la classe TextField vous permettent de définir, sélectionner et manipuler du texte dans un champ de texte dynamique ou de saisie que vous créez en cours de programmation ou à l'exécution. Pour plus d'informations, consultez *Utilisation de la classe TextField*, page 142. Pour plus d'informations sur les champs de texte de débogage lors de l'exécution, consultez *Affichage des propriétés de champ de texte pour le débogage*, page 84.

ActionScript propose également différentes manières de formater vos textes à l'exécution. La classe TextFormat vous permet de définir le formatage des caractères et des paragraphes pour les objets TextField (consultez *Utilisation de la classe TextFormat*, page 144). Flash Player prend également en charge un sous-ensemble de balises HTML à utiliser pour formater le texte (consultez *Utilisation de texte au format HTML*, page 154). Flash Player 7 et les versions ultérieures prennent en charge la balise HTML `<img>`, qui permet non seulement d'intégrer des images externes, mais également des fichiers SWF externes, ainsi que les clips qui résident dans la bibliothèque (consultez *Balise image (<img>)*, page 156).

Dans Flash Player 7 et versions ultérieures, vous pouvez appliquer des styles de feuilles de style en cascade (CSS) aux champs de texte à l'aide de la classe TextField.StyleSheet. Vous pouvez utiliser le style CSS pour l'appliquer aux balises HTML intégrées, définir de nouvelles balises de format ou appliquer des styles. Pour plus d'informations sur l'utilisation de CSS, consultez *Formatage de texte avec les feuilles de style en cascade*, page 145.

Vous pouvez également directement affecter du texte au format HTML, pouvant éventuellement utiliser des styles CSS, à un champ de texte. Dans Flash Player 7 et les versions ultérieures, le texte HTML que vous assignez à un champ de texte peut contenir des supports intégrés (clips vidéos, fichiers SWF et fichiers JPEG). Le texte enveloppe le média intégré, exactement comme un navigateur web enveloppe le texte autour d'un média intégré dans un document HTML. Pour plus d'informations, consultez *Balise image (<img>)*, page 156.

Utilisation de la classe TextField

La classe TextField représente tout champ de texte dynamique ou sélectionnable (modifiable) que vous créez à l'aide de l'outil Texte dans Flash. Utilisez les méthodes et les propriétés de cette classe pour contrôler les champs de texte à l'exécution. Les objets TextField supportent les mêmes propriétés que les objets MovieClip à l'exception des propriétés `_currentframe`, `_droptarget`, `_framesloaded` et `_totalframes`. Vous pouvez obtenir et définir des propriétés et invoquer des méthodes pour les champs de texte de façon dynamique.

Pour contrôler un champ de texte dynamique ou de saisie en utilisant ActionScript, vous devez lui affecter un nom d'occurrence dans l'inspecteur des propriétés. Vous pouvez ensuite faire référence au champ de texte avec le nom de l'occurrence et utiliser les méthodes et les propriétés de la classe TextField pour contrôler le contenu ou l'apparence générale du champ de texte. Vous pouvez également créer des objets TextField à l'exécution et leur affecter des noms d'occurrence à l'aide de la méthode `MovieClip.createTextField()`. Pour plus d'informations, consultez *Création de champs de texte à l'exécution*, page 143.

Affectation de texte à un champ de texte à l'exécution

Pour affecter du texte à un champ de texte, utilisez la propriété `TextField.text`.

Pour affecter du texte à un champ de texte à l'exécution :

1. En utilisant l'outil Texte, créez un champ de texte sur la scène.
2. Le champ de texte étant sélectionné, dans l'inspecteur des propriétés (Fenêtre > Propriétés), entrez `titre_txt` dans la zone de texte Nom de l'occurrence, située au-dessous du menu contextuel Type de texte sur le côté gauche de l'inspecteur.

   Les noms d'occurrence peuvent uniquement comporter des lettres, des traits de soulignement (_) et des dollars ($).
3. Dans le scénario, sélectionnez la première image dans le calque 1 et ouvrez le panneau Actions (Fenêtre > Panneaux de développement > Actions).
4. Tapez le code suivant dans le panneau Actions :

   ```
   titre_txt.text = "Le Brésil remporte la Coupe du monde";
   ```
5. Choisissez Contrôle > Tester l'animation pour tester l'animation.

A propos des noms d'occurrence et de variable de champ de texte

Dans l'inspecteur des propriétés, vous pouvez également affecter un nom de variable à un champ de texte dynamique ou de saisie, ainsi qu'un nom d'occurrence. Vous pouvez ensuite faire référence au nom de variable du champ de texte dans ActionScript, dont la valeur détermine le contenu du champ de texte. Cependant, il est important de ne pas confondre le nom d'occurrence et le nom de variable d'un champ de texte.

Utilisez le nom d'occurrence affecté à un champ de texte pour invoquer des méthodes, obtenir et définir les propriétés de l'objet champ de texte. Le nom de variable d'un champ de texte est tout simplement une référence de variable au texte contenu dans ce champ de texte, il ne s'agit pas d'une référence à un objet.

Par exemple, si vous affectez à un champ de texte le nom de variable `montexteVar`, vous pouvez ensuite définir le contenu du champ de texte en utilisant le code suivant :

```
var montexteVar = "Ceci apparaîtra dans le champ de texte";
```

Cependant, vous ne pouvez pas utiliser la variable `montexteVar` pour définir la même propriété de texte de champ de texte à un autre texte.

```
//Cela ne fonctionnera pas
monTexteVar.text = "Une variable de champ de texte n'est pas une référence à un
  objet";
```

En général, utilisez la propriété `TextField.text` pour contrôler le contenu d'un champ de texte, sauf si vous ciblez une version de Flash Player qui ne supporte pas la classe TextField. Ceci réduira l'éventualité d'un conflit de noms de variables, qui pourrait engendrer un comportement inattendu à l'exécution.

Création de champs de texte à l'exécution

Vous pouvez utiliser la méthode `createTextField()` de la classe MovieClip pour créer un champ de texte vide sur la scène à l'exécution. Ce nouveau champ est associé au scénario du clip qui appelle la méthode. La méthode `createTextField()` utilise la syntaxe suivante :

```
movieClip.createTextField(nomDoccurrence, profondeur, x, y, largeur, hauteur)
```

Par exemple, le code suivant crée un champ de texte de 300 x 100 pixels nommé `test_txt` au point (0,0) et à une profondeur (ordre *z*) de 10.

```
_root.createTextField("test_txt", 10, 0, 0, 300, 100);
```

Utilisez le nom d'occurrence spécifié dans l'appel `createTextField()` pour accéder aux méthodes et aux propriétés de la classe TextField. Par exemple, le code suivant crée un nouveau champ de texte nommé `test_txt`, puis modifie ses propriétés pour en faire un champ de texte multiligne avec retour automatique à la ligne, qui se développe pour s'ajuster à la taille du texte inséré. Enfin, il affecte du texte à la propriété `text` du champ de texte.

```
_root.createTextField("test_txt", 10, 0, 0, 100, 50);
test_txt.multiline = true;
test_txt.wordWrap = true;
test_txt.autoSize = true;
test_txt.text = "Créez de nouveaux champs de texte à l'aide de la méthode
  MovieClip.createTextField.";
```

Vous pouvez utiliser la méthode `TextField.removeTextField()` pour supprimer un champ de texte créé avec `createTextField()`. La méthode `removeTextField()` ne fonctionne pas pour les champs de texte placés par le scénario au cours de la programmation.

Pour plus d'informations, consultez `MovieClip.createTextField()`, page 556 et `TextField.removeTextField()`, page 797.

Utilisation de la classe TextFormat

Vous pouvez utiliser la classe TextFormat d'ActionScript pour définir les propriétés de formatage d'un champ de texte. Cette classe intègre des informations sur le formatage des caractères et des paragraphes. Les informations sur le formatage des caractères décrivent l'apparence des différents caractères : nom de police, taille, couleur et URL associée. Les informations sur le formatage des paragraphes décrivent l'apparence d'un paragraphe : marge de gauche, marge de droite, indentation de la première ligne, ainsi qu'alignement à gauche, droite ou au centre.

Pour utiliser la classe TextFormat, vous devez d'abord créer un objet TextFormat et définir ses styles de formatage de caractères et de paragraphes. Appliquez ensuite l'objet TextFormat à un champ de texte à l'aide des méthodes `TextField.setTextFormat()` ou `TextField.setNewTextFormat()`.

La méthode `setTextFormat()` modifie le format de texte appliqué à chaque caractère, à des groupes de caractères ou à l'ensemble du corps de texte d'un champ de texte. Cependant, le texte nouvellement inséré (tel que celui entré par l'utilisateur ou inséré avec ActionScript) n'adopte pas le formatage spécifié par un appel `setTextFormat()`. Pour spécifier le formatage par défaut pour d'un texte nouvellement inséré, utilisez `TextField.setNewTextFormat()`. Pour plus d'informations, consultez `TextField.setTextFormat`, page 802 et `TextField.setNewTextFormat`, page 801.

Pour formater un champ de texte avec la classe TextFormat :

1. Dans un nouveau document Flash, créez un champ de texte sur la scène à l'aide de l'outil Texte. Tapez du texte dans le champ de texte sur la scène; par exemple « Texte gras, italique, 24 points ».
2. Dans l'inspecteur des propriétés, tapez `monTexte_txt` dans la zone de texte Nom de l'occurrence, sélectionnez Dynamique dans le menu contextuel Type de texte et sélectionnez Multiligne dans le menu contextuel Type de ligne.
3. Dans le scénario, sélectionnez la première image dans le calque 1 et ouvrez le panneau Actions (Fenêtre > Panneaux de développement > Actions).
4. Entrez le code suivant dans le panneau Actions pour créer un objet TextFormat et définissez ses propriétés `bold` et `italic` sur la valeur `true` et sa propriété `size` sur 24.

```
// Créez un objet TextFormat
var txtfmt_fmt = new TextFormat();
// Spécifiez le formatage des paragraphes et des caractères
txtfmt_fmt.bold = "true";
txtfmt_fmt.italic = "true";
txtfmt_fmt.size = "24"
```

5. Appliquez l'objet TextFormat au champ de texte que vous avez créé à l'étape 1 en utilisant `TextField.setTextFormat()`.

```
monTexte_txt.setTextFormat(txtfmt_fmt);
```

   Cette version de `setTextFormat()` applique le formatage spécifié à l'intégralité du champ de texte. Deux autres versions de cette méthode vous permettent d'appliquer le formatage à des caractères individuels ou à des groupes de caractères. Par exemple, le code suivant applique le formatage gras, italique, 24 points aux quatre premiers caractères que vous avez entrés dans le champ de texte.

```
monTexte_txt.setTextFormat(txtfmt_fmt, 0, 3);
```

   Pour plus d'informations, consultez `TextField.setTextFormat`, page 802.

6. Choisissez Contrôle > Tester l'animation pour tester l'animation.

Propriétés par défaut des nouveaux champs de texte

Les champs de texte créés à l'exécution à l'aide de `createTextField()` reçoivent un objet TextFormat par défaut avec les propriétés suivantes :

```
font = "Times New Roman"
size = 12
textColor = 0x000000
bold = false
italic = false
underline = false
url = ""
target = ""
align = "left"
leftMargin = 0
rightMargin = 0
indent = 0
leading = 0
bullet = false
tabStops = [] (tableau vide)
```

Pour obtenir la liste complète des méthodes TextFormat et leur description, consultez l'entrée *Classe TextFormat* dans le Chapitre 12, *Dictionnaire ActionScript*, page 215.

Obtention des informations métriques du texte

Vous pouvez utiliser la méthode `TextFormat.getTextExtent()` pour obtenir les mesures de texte détaillées d'une chaîne de texte à laquelle est appliquée un formatage spécifique. Supposons, par exemple, que vous deviez créer, à l'exécution, un objet TextField contenant une quantité de texte aléatoire formaté en 24 points, gras, police Arial, avec un retrait de 5 pixels. Vous devez déterminer la largeur et la hauteur du nouvel objet TextField pour afficher tout le texte. La méthode `getTextExtent()` fournit des mesures telles que les mesures ascendantes, descendantes, de la largeur et de la hauteur.

Pour plus d'informations, consultez `TextFormat.getTextExtent()`, page 823.

Formatage de texte avec les feuilles de style en cascade

Les feuilles de style en cascade sont un mécanisme permettant de créer des styles de textes qui peuvent être appliqués aux documents HTML ou XML. Une feuille de style est un ensemble de règles de formatage qui spécifie comment formater des éléments HTML ou XML. Chaque règle associe un nom de style, ou *sélecteur*, à une ou plusieurs propriétés de style ainsi qu'à leurs valeurs. Par exemple, le style suivant définit un sélecteur nommé `bodyText`.

```
bodyText { text-align: left}
```

Vous pouvez créer des styles qui redéfinissent les balises HTML de formatage intégrées utilisées par Flash Player (telles que `<p>` et `<li>`), créer des « classes » de style pouvant être appliquées à des éléments HTML spécifiques à l'aide de l'attribut de `classe` des balises `<p>` ou `<span>` ou définir de nouvelles balises.

Utilisez la classe TextField.StyleSheet pour utiliser les feuilles de style de texte. Vous pouvez charger des styles à partir d'un fichier CSS externe ou les créer de façon native en utilisant ActionScript. Pour appliquer une feuille de style à un champ de texte contenant du texte au format HTML ou XML, utilisez la propriété `TextField.styleSheet`. Les styles définis dans la feuille de style sont automatiquement mappés aux balises définies dans le document HTML ou XML.

Pour utiliser des feuilles de style, vous devez suivre trois étapes basiques :

- Créez un objet feuille de style à partir de la classe TextField.StyleSheet. Pour plus d'informations, consultez *Création d'un objet feuille de style*, page 147.
- Ajoutez des styles à l'objet feuille de style, soit en les important d'un fichier CSS externe, soit en les définissant avec ActionScript. Consultez *Chargement de fichiers CSS externes*, page 147 et *Création de nouveaux styles avec ActionScript*, page 148.
- Affectez l'objet feuille de style à un champ de texte contenant du texte au format XML ou HTML. Consultez *Application de styles à un objet TextField*, page 149, *Exemple d'utilisation de styles avec HTML*, page 151 et *Exemple d'utilisation de styles avec XML*, page 153.

Propriétés CSS supportées

Flash Player supporte un sous-ensemble de propriétés dans la spécification CSS1 d'origine (www.w3.org/TR/REC-CSS1). Le tableau suivant présente les propriétés et les valeurs CSS supportées et les noms de propriétés ActionScript correspondants. (Chaque nom de propriété ActionScript est tiré du nom de propriété CSS correspondant. Le trait d'union est omis et le caractère suivant est une majuscule.)

Propriété CSS	Propriété ActionScript	Utilisation et valeurs supportées
text-align	textAlign	Les valeurs reconnues sont `left`, `center` et `right`.
font-size	fontSize	Seule la partie numérique de la valeur est utilisée ; les unités (px, pt) ne sont pas analysées ; les pixels et les points sont équivalents.
text-decoration	textDecoration	Les valeurs reconnues sont `none` et `underline`.
margin-left	marginLeft	Seule la partie numérique de la valeur est utilisée. Les unités (px, pt) ne sont pas analysées ; les pixels et les points sont équivalents.
margin-right	marginRight	Seule la partie numérique de la valeur est utilisée. Les unités (px, pt) ne sont pas analysées ; les pixels et les points sont équivalents.
font-weight	fontWeight	Les valeurs reconnues sont `normal` et `bold`.
font-style	fontStyle	Les valeurs reconnues sont `normal` et `italic`.
text-indent	textIndent	Seule la partie numérique de la valeur est utilisée. Les unités (px, pt) ne sont pas analysées ; les pixels et les points sont équivalents.
font-family	fontFamily	Liste des polices à utiliser, séparées par des virgules, classées par ordre de choix décroissant. Tous les noms de familles de polices peuvent être utilisés. Si vous spécifiez un nom de police générique, il sera converti dans la police de périphérique appropriée. Les conversions de police suivantes sont disponibles : `mono` est converti en `_typewriter`, `sans-serif` est converti en`_sans` et `serif` est converti en `_serif`.

Propriété CSS	Propriété ActionScript	Utilisation et valeurs supportées
color	color	Seules les valeurs hexadécimales de couleur sont supportées. Les couleurs nommées (comme `blue`) ne sont pas supportées.
display	display	Les valeurs supportées sont `inline`, `block` et `none`.

Création d'un objet feuille de style

Les feuilles de style CSS sont représentées dans ActionScript par la classe `TextField.StyleSheet`. Cette classe est uniquement disponible pour les fichiers SWF conçus pour Flash Player 7 et les versions ultérieures. Pour créer un objet feuille de style, appelez la fonction de constructeur de la classe TextField.StyleSheet.

```
var nouveauStyle = new TextField.StyleSheet();
```

Pour ajouter des styles à un objet feuille de style, chargez un fichier CSS externe dans l'objet ou définissez les styles dans ActionScript. Consultez *Chargement de fichiers CSS externes*, page 147 et *Création de nouveaux styles avec ActionScript*, page 148.

Chargement de fichiers CSS externes

Vous pouvez définir des styles dans un fichier CSS externe puis charger ce fichier dans un objet feuille de style. Les styles définis dans le fichier CSS sont ajoutés à l'objet feuille de style. Pour charger un fichier CSS externe, utilisez la méthode `load()` de la classe TextField.StyleSheet. Pour déterminer le moment où le chargement du fichier CSS est terminé, utilisez le gestionnaire d'événement `onLoad` de l'objet feuille de style.

Dans l'exemple suivant, créez et chargez un fichier CSS externe et utilisez la méthode `TextField.StyleSheet.getStyleNames()` pour récupérer les noms des styles chargés.

Pour charger une feuille de style externe :

1 Créez un nouveau fichier dans l'éditeur de texte ou l'éditeur XML de votre choix.

2 Ajoutez les définitions de style suivantes au fichier :

```
// Nom de fichier : styles.css
bodyText {
  font-family: Arial,Helvetica,sans-serif;
  font-size 12px;
}

headline {
  font-family: Arial,Helvetica,sans-serif;
  font-size 24px;
}
```

3 Enregistrez le fichier CSS sous styles.css.

4 Dans Flash, créez un document FLA.

5 Dans le scénario (Fenêtre > Scénario), sélectionnez le calque 1.

6 Ouvrez le panneau Actions (Fenêtre > Panneaux de développement > Actions).

7 Ajoutez le code suivant au panneau Actions

```
var css_styles = new TextField.StyleSheet();
css_styles.load("styles.css");
css_styles.onLoad = function(ok) {
  if(ok) {
    // afficher les noms de style
    trace(this.getStyleNames());
  } else {
    trace("Erreur lors du chargement du fichier CSS.");
  }
}
```

8 Enregistrez le fichier dans le même répertoire que celui contenant styles.css.

9 Testez l'animation (Contrôle > Tester l'animation).

Les noms des deux styles devraient s'afficher dans le panneau de sortie :

```
body
titre
```

Si « Erreur lors du chargement du fichier CSS » s'affiche dans le panneau de sortie, vérifiez que le fichier FLA et le fichier CSS se trouvent bien dans le même répertoire et que vous avez correctement saisi le nom du fichier CSS.

Comme pour les autres méthodes ActionScript qui chargent des données via le réseau, le fichier CSS doit résider dans le même domaine que le fichier SWF qui effectue le chargement du fichier. Pour plus d'informations, consultez *A propos de l'autorisation de chargement de données inter-domaines*, page 201.

Création de nouveaux styles avec ActionScript

Vous pouvez créer de nouveaux styles de texte avec ActionScript en utilisant la méthode `setStyle()` de la classe TextField.StyleSheet. Cette méthode prend deux paramètres : le nom du style et un objet qui définit les propriétés de ce style.

Par exemple, le code suivant crée un objet feuille de style nommé `styles` qui définit deux styles identiques à ceux que vous avez importés (consultez *Chargement de fichiers CSS externes*, page 147).

```
var styles = new TextField.StyleSheet();
styles.setStyle("bodyText",
  {fontFamily: 'Arial,Helvetica,sans-serif',
  fontSize: '12px'}
);
styles.setStyle("titre",
  {fontFamily: 'Arial,Helvetica,sans-serif',
  fontSize: '24px'}
);
```

Application de styles à un objet TextField

Pour appliquer un objet feuille de style à un champ de texte, affectez cet objet à la propriété `styleSheet` de l'objet champ de texte.

```
textObj_txt.styleSheet = styleSheetObj;
```

Remarque : Evitez de confondre la *propriété* `TextField.styleSheet` avec la *classe* TextField.StyleSheet. Les majuscules les distinguent.

Lorsque vous affectez un objet feuille style à un objet TextField, le comportement normal du champ de texte est modifié de la manière suivante :

- Les propriétés `text` et `htmlText` du champ de texte et toute variable associée au champ de texte, contiennent toujours la même valeur et se comportent toujours de la même façon.
- Le champ de texte est en lecture seule et ne peut plus être modifié par l'utilisateur.
- Les méthodes `setTextFormat()` et `replaceSel()` de la classe TextField ne fonctionnent plus avec le champ de texte. La seule façon de modifier le champ consiste à changer les propriétés `text` ou `htmlText` du champ de texte ou à modifier les variables associées au champ de texte.
- Tout texte affecté à la propriété `text`, à la propriété `htmlText` ou aux variables associées du champ de texte est stocké textuellement. Tout ce qui est écrit dans l'une de ces propriétés peut être récupéré dans la forme originale du texte.

Association de styles

Les styles CSS dans Flash Player sont additionnels, c'est-à-dire que lorsque les styles sont imbriqués, chaque niveau d'imbrication peut fournir des informations de style supplémentaires, qui sont ajoutées les unes aux autres pour donner le formatage final.

Par exemple, voici des données XML affectées à un champ de texte :

```
<sectionHeading>Ceci est une section</sectionHeading>
<mainBody>Ceci est le corps principal du texte, avec un
mot <emphasized>emphatique</emphasized>.</mainBody>
```

Pour le mot *emphatique* du texte ci-dessus, le style `emphasized` est imbriqué dans le style `mainBody`. Le style `mainBody` fournit les règles de couleur, de taille de police et de décoration. Le style `emphasized` ajoute une règle d'épaisseur de police à ces règles. Le mot *emphatique* sera formaté en associant les règles spécifiées par `mainBody` et `emphasized`.

Utilisation des classes de style

Vous pouvez créer des classes de style que vous pouvez appliquer à une balise `<p>` ou `<span>` en utilisant l'attribut `Class` de l'une des balises. Lorsqu'une balise `<p>` est appliquée, le style affecte tout le paragraphe. Vous pouvez également appliquer un style à une plage de texte qui utilise une classe de style à l'aide de la balise `<span>`.

Par exemple, la feuille de style suivante définit deux classes de style : `mainBody` et `emphasis`.

```
.mainBody {
  font-family: Arial,Helvetica,sans-serif;
  font-size: 24px;
}
.emphasis {
  color: #666666;
  font-style: italic;
}
```

A l'intérieur du texte HTML que vous affectez à un champ de texte, vous pouvez appliquer ces styles aux balises <p> et comme expliqué ci-dessous :

```
<p class="mainBody">C'est <span class="emphasis">très motivant !</span></p>
```

Définition du style de balises HTML intégrées

Flash Player supporte un sous-ensemble de balises HTML. Pour plus d'informations, consultez *Utilisation de texte au format HTML*, page 154. Vous pouvez affecter un style CSS à chaque occurrence d'une balise HTML intégrée qui apparaît dans un champ de texte. Par exemple, le code suivant définit un style pour la balise HTML intégrée <p>. Toutes les occurrences de cette balise verront leur style défini de la manière spécifiée par la règle de style :

```
p {
  font-family: Arial,Helvetica,sans-serif;
  font-size: 12px;
  display: inline;
}
```

Le tableau suivant indique les balises HTML intégrées dont le style peut être défini, et la façon dont chaque style est appliqué :

Nom de style	Comment le style est appliqué
p	Affecte toutes les balises <p>.
body	Affecte toutes les balises <body>. S'il est spécifié, le style p est prioritaire par rapport au style body.
li	Affecte toutes les balises à puce .
a	Affecte toutes les balises d'ancrage <a>.
a:link	Affecte toutes les balises d'ancrage <a>. Ce style est appliqué après tout style a.
a:hover	Appliqué à une balise d'ancrage <a> lorsque la souris se déplace sur le lien. Ce style est appliqué après tout style a et a:link. Une fois que la souris s'éloigne du lien, le style a:hover est supprimé du lien.
a:active	Appliqué à une balise d'ancrage <a> lorsque l'utilisateur clique avec la souris sur le lien. Ce style est appliqué après tout style a et a:link. Une fois que le bouton de la souris est relâché, le style a:active est supprimé du lien.

Exemple d'utilisation de styles avec HTML

Cette section présente un exemple de l'utilisation des styles avec des balises HTML. Vous créez une feuille de style qui définit le style de certaines balises intégrées ainsi que certaines classes de style. Vous appliquez ensuite cette feuille de style à un objet TextField qui contient du texte au format HTML.

Pour formater un HTML à l'aide d'une feuille de style, effectuez les opérations suivantes :

1 Créez un fichier dans l'éditeur de texte de votre choix.

2 Ajoutez la définition de feuille de style suivante dans le fichier :

```
p {
  color: #000000;
  font-family: Arial,Helvetica,sans-serif;
  font-size: 12px;
  display: inline;
}

a:link {
  color: #FF0000;
}

a:hover{
  text-decoration: underline;
}

.headline {
  color: #000000;
  font-family: Arial,Helvetica,sans-serif;
  font-size: 18px;
  font-weight: bold;
  display: block;
}

.byline {
  color: #666600;
  font-style: italic;
  font-weight: bold;
  display: inline;
}
```

Cette feuille de style définit des styles pour deux balises HTML intégrées (`<p>` et `<a>`) qui seront appliqués à toutes les occurrences de ces balises. Elle définit également deux classes de style (`.headline` et `.byline`) qui seront appliquées à des paragraphes et à des plages de texte spécifiques.

3 Enregistrez le fichier sous html_styles.css.

4 Dans Flash, créez un fichier FLA.

5 A l'aide de l'outil Texte, créez un champ de texte d'environ 400 pixels de large et 300 pixels de haut.

6 Ouvrez l'inspecteur des propriétés (Fenêtre > Propriétés) et sélectionnez le champ de texte.

7 Dans l'inspecteur des propriétés, sélectionnez Texte dynamique dans le menu Type de texte, sélectionnez Multiligne dans le menu Type de ligne, sélectionnez l'option Rendre le texte au format HTML et tapez `news_txt` dans la zone de texte Nom de l'occurrence.

8 Sélectionnez la première image dans le calque 1 du scénario (Fenêtre > Scénario).

9 Ouvrez le panneau Actions (Fenêtre > Panneaux de développement > Actions) et ajoutez-lui le code suivant :

```
// Créez un objet feuille de style
var feuille_style = new TextField.StyleSheet();
// Emplacement du fichier CSS qui définit les styles
var css_url = "html_styles.css";
// Créez du texte HTML à afficher
var storyText:String = "<p class='headline'>Flash Player supporte désormais
  les feuilles de style en cascade !</p><p><span class='byline'>San
  Francisco, Californie</span>--Macromedia Inc. annonce aujourd'hui une
  nouvelle version de Flash Player qui supporte les styles de texte de
  feuilles de style en cascade (CSS). Pour plus d'informations, visitez le
  <a href='http://www.macromedia.com'>site web Macromedia Flash.</a></p>";
// Chargez le fichier CSS et définissez le gestionnaire onLoad :
feuille_style.load(css_url);
feuille_style.onLoad = function(ok) {
  if (ok) {
    // Si la feuille de style a été chargée sans erreur,
    // affectez-la à l'objet texte,
    // et affectez le texte HTML au champ de texte.
    news_txt.styleSheet = style_sheet;
    news_txt.text = storyText;
  }
};
```

Remarque : Pour plus de simplicité, le texte HTML dont le style doit être défini est « codé en dur » dans le script. Dans une application réelle, vous chargeriez le texte à partir d'un fichier externe. Pour plus d'informations sur le chargement de données externes, consultez le Chapitre 10, *Utilisation de données externes*, page 187.

10 Enregistrez le fichier sous news_html.fla dans le même répertoire que celui contenant le fichier CSS que vous avez créé précédemment.

11 Exécutez l'animation (Contrôle > Tester l'animation) pour voir les styles appliqués au texte HTML automatiquement.

Utilisation de styles pour définir de nouvelles balises

Si vous définissez un nouveau style dans une feuille de style, ce style peut être utilisé comme balise, comme vous utiliseriez une balise HTML intégrée. Par exemple, si une feuille de style définit un style CSS nommé `sectionHeading`, vous pouvez utiliser `<sectionHeading>` comme élément dans tout champ de texte associé à la feuille de style. Cette fonction vous permet d'affecter directement du texte aléatoire au format XML dans un champ de texte, afin que le texte soit automatiquement formaté en utilisant les règles de la feuille de style.

Par exemple, la feuille de style suivante crée les nouveaux styles `sectionHeading`, `mainBody` et `emphasized`.

```
sectionHeading {
  font-family: Verdana, Arial, Helvetica, sans-serif;
  font-size: 18px; display: block
}
mainBody {
  color: #000099;
  text-decoration: underline;
  font-size: 12px; display: block
}
emphasized {
  font-weight: bold; display: inline
}
```

Vous pouvez ensuite remplir un champ de texte associé à cette feuille de style avec le texte au format XML suivant :

```
<sectionHeading>Ceci est une section</sectionHeading>
<mainBody>Ceci est le corps principal du texte,
avec un <emphasized>mot</emphasized> emphatique.
</mainBody>
```

Exemple d'utilisation de styles avec XML

Dans cette section, vous créez le même fichier FLA que précédemment (consultez *Exemple d'utilisation de styles avec HTML*, page 151), mais avec du texte au format XML. Dans cet exemple, cependant, vous créez la feuille de style en utilisant ActionScript au lieu d'importer des styles à partir d'un fichier CSS.

Pour formater du XML avec une feuille de style :

1. Dans Flash, créez un fichier FLA.
2. A l'aide de l'outil Texte, créez un champ de texte d'environ 400 pixels de large et 300 pixels de haut.
3. Ouvrez l'inspecteur des propriétés (Fenêtre > Propriétés) et sélectionnez le champ de texte.
4. Dans l'inspecteur des propriétés, sélectionnez Texte dynamique dans le menu Type de texte, sélectionnez Multiligne dans le menu Type de ligne, sélectionnez l'option Rendre le texte au format HTML et tapez `news_txt` dans la zone de texte Nom de l'occurrence.
5. Dans le calque 1 du scénario (Fenêtre > Scénario), sélectionnez la première image.
6. Pour créer l'objet feuille de style, ouvrez le panneau Actions (Fenêtre > Panneaux de développement > Actions) et ajoutez le code suivant au panneau Actions :

```
var xml_styles = new TextField.StyleSheet();
xml_styles.setStyle("mainBody", {
  color:'#000000',
  fontFamily:'Arial,Helvetica,sans-serif',
  fontSize:'12',
  display:'block'
});
xml_styles.setStyle("title", {
  color:'#000000',
  fontFamily:'Arial,Helvetica,sans-serif',
  fontSize:'18',
  display:'block',
  fontWeight:'bold'
});
xml_styles.setStyle("byline", {
  color:'#666666',
  fontWeight:'bold',
  fontStyle:'italic',
  display:'inline'
});
xml_styles.setStyle("a:link", {
  color:'#FF0000'
});
xml_styles.setStyle("a:hover", {
  textDecoration:'underline'
});
```

Ce code crée un nouvel objet feuille de style nommé `xml_styles` qui définit les styles à l'aide de la méthode `setStyle()`. Les styles correspondent exactement à ceux que vous avez créés dans un fichier CSS externe, précédemment dans ce même chapitre.

7 Pour créer du texte XML à affecter au champ de texte, ajoutez le code suivant au panneau Actions :

```
var storyText = "<title>Flash Player supporte désormais CSS</
  title><mainBody><byline>San Francisco, Californie</byline>--Macromedia
  Inc. annonce aujourd'hui une nouvelle version de Flash Player qui prend en
  charge les styles de texte de feuilles de style en cascade (CSS). Pour
  plus d'informations, visitez le <a href=\"http://www.macromedia.com\">site
  web de Macromedia Flash</a></mainBody>";
```

8 Enfin, ajoutez le code suivant pour appliquer l'objet feuille de style à la propriété `styleSheet` du champ de texte et affectez le texte XML au champ de texte.

```
news_txt.styleSheet = xml_styles;
news_txt.text = storyText;
```

9 Enregistrez le fichier sous news_xml.fla.

10 Exécutez l'animation (Contrôle > Tester l'animation) pour voir les styles automatiquement appliqués au texte dans le champ.

Utilisation de texte au format HTML

Flash Player supporte un sous-ensemble de balises HTML comme `<p>` et `<li>` que vous pouvez utiliser pour attribuer un style à un texte dans un champ de texte dynamique ou de saisie. Dans Flash Player 7 et les versions ultérieures, les champs de texte prennent également en charge la balise `<img>`, qui vous permet d'intégrer les fichiers JPEG, SWF et les clips vidéo, dans un champ de texte. Dans Flash Player, le texte enveloppe automatiquement les images intégrées dans les champs de texte, de la même manière qu'un navigateur web enveloppe le texte autour des images intégrées dans un document HTML. Pour plus d'informations, consultez *Intégration des images, fichiers SWF et des clips dans les champs de texte*, page 159.

Flash Player prend également en charge la balise `<textformat>`, qui vous permet d'appliquer les styles de formatage de paragraphe de la classe TextFormat aux champs de texte HTML. Pour plus d'informations, consultez *Utilisation de la classe TextFormat*, page 144.

Présentation de l'utilisation du texte au format HTML

Pour utiliser le format HTML dans un champ de texte, vous devez activer le formatage HTML du champ de texte en sélectionnant l'option Rendre le texte au format HTML dans l'inspecteur des propriétés ou en définissant la propriété `html` du champ de texte sur `true`. Pour insérer du texte HTML dans un champ de texte, utilisez la propriété `TextField.htmlText`.

Par exemple, le code suivant active le formatage HTML pour un champ de texte nommé `titre_txt`, puis lui affecte du texte HTML.

```
titre_txt.html = true;
titre_txt.htmlText = "<font face='Times New Roman' size='24'>Voici comment
  affecter du texte HTML à un champ de texte.</font>";
```

Les attributs des balises HTML doivent être encadrés de guillemets simples ou doubles. Les valeurs des attributs dépourvues de guillemets peuvent engendrer des résultats inattendus, comme le rendu incorrect du texte. Par exemple, le fragment de code HTML suivant ne sera pas rendu correctement par Flash Player, car la valeur affectée à l'attribut `align` (`left`) n'est pas encadrée de guillemets :

```
textField.htmlText = "<p align=left>Texte aligné sur la gauche</p>";
```

Si vous encadrez les valeurs d'attribut à l'aide de guillemets doubles, vous devez utiliser la fonction d'échappement pour les guillemets (`\"`). Par exemple, l'une des solutions suivantes est possible :

```
textField.htmlText = "<p align='left'>Guillemets simples utilisés</p>";
textField.htmlText = "<p align=\"left\">Guillemets doubles échappés</p>";
```

Il n'est pas nécessaire d'utiliser la fonction d'échappement pour les guillemets doubles si vous chargez du texte à partir d'un fichier externe. Cette action est uniquement nécessaire pour l'affectation des chaînes de texte dans ActionScript.

Balises HTML prises en charge

Cette section répertorie les balises HTML intégrées prises en charge par Flash Player. Vous pouvez également créer de nouveaux styles et balises en utilisant les feuilles de style en cascade (consultez *Formatage de texte avec les feuilles de style en cascade*, page 145).

Balise d'ancrage (<a>)

La balise `<a>` crée un hyperlien et prend en charge les attributs suivants :

- `href` Spécifie l'URL de la page à charger dans le navigateur. L'URL peut être absolue ou relative à l'emplacement du fichier SWF qui charge la page.
- `target` Spécifie le nom de la fenêtre cible dans laquelle charger la page.

Par exemple, le fragment de code HTML suivant crée le lien « Accueil », qui ouvre www.macromedia.com dans une nouvelle fenêtre du navigateur.

```
<a href="../home.htm" target="_blank">Accueil</a>
```

Vous pouvez également définir des styles `a:link`, `a:hover` et `a:active` pour les balises d'ancrage en utilisant les feuilles de style. Pour plus d'informations, consultez *Définition du style de balises HTML intégrées*, page 150.

Balise Bold (<b>)

La balise `<b>` rend le texte en caractères gras. Les caractères gras doivent être disponibles dans la police utilisée pour afficher le texte.

```
<b>Ceci est du texte en gras.</b>
```

Balise Break (
)

La balise `<br>` crée un saut de ligne dans le champ de texte, comme indiqué dans l'exemple suivant :

```
Une ligne de texte<br>Une autre ligne de texte<br>
```

Balise Font (<font>)

La balise `<font>` spécifie une police ou une liste de polices pour l'affichage du texte.

La balise font prend en charge les attributs suivants :

- `color` Seules les valeurs de couleur hexadécimales (`#FFFFFF`) sont prises en charge. Par exemple, le code HTML suivant crée du texte rouge.

  ```
  <font color="#FF0000">Texte rouge</font>
  ```

- `face` Spécifie le nom de la police à utiliser. Vous pouvez également spécifier une liste de noms de polices séparés par des virgules, auquel cas Flash Player choisit la première police disponible. Si la police spécifiée n'est pas installée sur le système de lecture ou si elle n'est pas intégrée dans le fichier SWF, Flash Player choisit une police de remplacement.

  Exemple :

  ```
  <font face="Times, Times New Roman">Il s'agit soit de la police Times, soit
    de la police Times New Roman..</font>
  ```

  Pour plus d'informations sur l'intégration des polices dans les applications Flash, consultez `TextField.embedFonts`, page 783 et « Définition des options de texte dynamique et de saisie », dans le guide Utilisation de Flash de l'aide.

- `size` Spécifie la taille de la police, en pixels. Vous pouvez également utiliser des tailles de points relatives (+2 ou -4).

  ```
  <font size="24" color="#0000FF">Texte vert à 24 points </font>
  ```

Balise image (<img>)

La balise `<img>` vous permet d'intégrer des fichiers JPEG, SWF et des clips externes à l'intérieur des champs de texte. Le texte se déroule automatiquement autour des images intégrées dans les champs de texte. Cette balise est prise en charge uniquement dans les champs de texte dynamique et de saisie multilignes avec retour à la ligne automatique.

Pour créer un champ de texte multiligne avec retour à la ligne automatique, effectuez l'une des opérations suivantes :

- Dans l'environnement auteur Flash, sélectionnez un champ de texte sur la scène, puis, dans l'inspecteur des propriétés, sélectionnez Multiligne dans le menu contextuel Type de texte.
- Pour un champ de texte créé à l'exécution avec `MovieClip.createTextField()`, définissez les propriétés `TextField.multiline` et `TextField.wordWrap` de la nouvelle occurrence du champ de texte sur `true`.

La balise `<img>` a un attribut requis, `src`, qui spécifie le chemin vers un fichier JPEG, un fichier SWF ou l'identifiant de liaison d'un symbole de clip. Tous les autres attributs sont facultatifs.

Les balises `<img>` prennent en charge les attributs suivants :

- `src` Spécifie l'URL vers un fichier JPEG ou SWF, ou l'identifiant de liaison pour un symbole de clip dans la bibliothèque. Cet attribut est requis ; tous les autres attributs sont facultatifs. Les fichiers externes (JPEG et SWF) ne s'affichent pas tant qu'ils ne sont pas entièrement téléchargés.

  Remarque : Flash ne supporte pas les fichiers JPEG tridimensionnels.

- `id` Spécifie le nom d'une occurrence de clip (créée par Flash Player) contenant le fichier JPEG, SWF ou le clip intégré. Cette fonction s'avère utile pour contrôler le contenu intégré avec ActionScript.
- `width` Largeur de l'image, du fichier SWF ou du clip, en pixels.
- `height` Hauteur de l'image, du fichier SWF ou du clip, en pixels.
- `align` Spécifie l'alignement horizontal de l'image intégrée dans le champ de texte. Les valeur valides sont `left` et `right`. La valeur par défaut est `left`.
- `hspace` Spécifie l'espace horizontal qui entoure l'image là où aucun texte n'apparaît. La valeur par défaut est 8.
- `vspace` Spécifie l'espace vertical qui entoure l'image là où aucun texte n'apparaît. La valeur par défaut est 8.

Pour plus d'informations et d'exemples sur l'utilisation de la balise `<img>`, consultez *Intégration des images, fichiers SWF et des clips dans les champs de texte*, page 159.

Balise Italic (<i>)

La balise `<i>` affiche le texte balisé en italique. Des caractères italiques doivent être disponibles dans la police utilisée.

```
C'est très <i>intéressant</i>.
```

Le code ci-dessus serait rendu de la manière suivante :

C'est très *intéressant*.

Balise List item (<li>)

La balise `<li>` place une puce devant le texte qu'elle encadre.

```
Liste de courses :
<li>Pommes</li>
<li>Oranges</li>
<li>Citrons</li>
```

Le code ci-dessus serait rendu de la manière suivante :

Liste de courses :

- Pommes
- Oranges
- Citrons

Balise Paragraph (<p>)

La balise `<p>` crée un paragraphe. Elle prend en charge les attributs suivants :

- `align` Spécifie l'alignement du texte dans le paragraphe. Les valeurs valides sont `left`, `right` et `center`.
- `class` Spécifie une classe de style CSS définie par un objet TextField.StyleSheet. Pour plus d'informations, consultez *Utilisation des classes de style*, page 149.

L'exemple suivant utilise l'attribut `align` pour aligner le texte sur le côté droit d'un champ de texte.

```
textField.htmlText = "<p align='right'>Ce texte est aligné sur la droite
  dans le champ de texte</p>";
```

L'exemple suivant utilise l'attribut `class` pour affecter une classe de style de texte à une balise `<p>`.

```
var maFeuilleDeStyle = new TextField.StyleSheet();
maFeuilleDeStyle.secreateTextField("test", 10, 0,0, 300,100);
createTextField("test", 10, 0,0, 300,100);
test.styleSheet = maFeuilleDeStyle;
test.htmlText = "<p class='body'>Ceci est du texte de style corps.</p>.";
```

Balise Span ()

La balise `<span>` peut uniquement être utilisée avec les styles de texte CSS. Pour plus d'informations, consultez *Formatage de texte avec les feuilles de style en cascade*, page 145. Elle prend en charge les attributs suivants :

- `class` Spécifie une classe de style CSS définie par un objet TextField.StyleSheet. Pour plus d'informations sur la création de classes de style, consultez *Utilisation des classes de style*, page 149.

Balise Text format (<textformat>)

La balise `<textformat>` permet d'utiliser un sous-ensemble de propriétés de formatage des paragraphes de la classe TextFormat dans les champs de texte HTML, y compris l'interlignage, le retrait, les marges et les taquets de tabulation. Vous pouvez associer des balises `<textformat>` aux balises HTML intégrées.

La balise `<textformat>` possède les attributs suivants :

- `blockindent` Spécifie l'indentation d'un bloc, en points. Correspond à `TextFormat.blockIndent`. Pour plus d'informations, consultez `TextFormat.blockIndent`, page 821.
- `indent` Spécifie l'indentation, de la marge gauche au premier caractère du paragraphe. Correspond à `TextFormat.indent`. Pour plus d'informations, consultez `TextFormat.indent`, page 825.
- `leading` Spécifie l'espace séparant les lignes (espace vertical). Correspond à `TextFormat.leading`. Pour plus d'informations, consultez `TextFormat.leading`, page 826.
- `leftmargin` Spécifie la marge gauche du paragraphe, en points. Correspond à `TextFormat.leftMargin`. Pour plus d'informations, consultez `TextFormat.leftMargin`, page 826.
- `rightmargin` Spécifie la marge droite du paragraphe, en points. Correspond à `TextFormat.rightMargin`. Pour plus d'informations, consultez `TextFormat.rightMargin`, page 826.
- `tabstops` Spécifie des taquets de tabulation personnalisés, sous forme d'un tableau d'entiers non négatifs. Correspond à `TextFormat.tabStops`. Pour plus d'informations, consultez `TextFormat.tabStops`, page 827.

L'exemple de code suivant utilise l'attribut `tabstops` de la balise `<textformat>` pour créer un tableau de données avec des en-têtes de lignes en gras, comme indiqué ci-dessous :

Nom	Age	Service
Thomas	32	IMD
Eluard	46	Ingénierie

Pour créer un tableau de données formaté en utilisant des taquets de tabulation :

1 Utilisez l'outil Texte pour créer un champ de texte dynamique d'environ 300 pixels de largeur et 100 pixels de hauteur.

2 Dans l'inspecteur des propriétés, saisissez `tableau_txt` dans le champ de texte Nom de l'occurrence, sélectionnez Multiligne dans le menu Type de ligne et choisissez l'option Rendre le texte au format HTML.

3 Dans le scénario, sélectionnez la première image sur le Calque 1.

4 Ouvrez le panneau Actions (Fenêtre > Panneaux de développement > Actions) et ajoutez-lui le code suivant :

```
var rowHeaders = "<b>Nom\t</b><b>Age\t</b><b>Service";
var row_1 = "Thomas\t31\tIMD";
var row_2 = "Eluard\t42\tQA";
tableau_txt.htmlText = "<textformat tabstops='[100, 200]'>";
tableau_txt.htmlText += rowHeaders;
tableau_txt.htmlText += row_1;
tableau_txt.htmlText += row_2 ;
tableau_txt.htmlText += "</textformat>";
```

Remarquez l'utilisation de la séquence d'échappement des caractères de tabulation (`\t`) pour ajouter des tabulations entre chaque « colonne » dans le tableau.

5 Choisissez Contrôle > Tester l'animation pour tester l'animation.

Balise Underline (<u>)

La balise `<u>` souligne le texte balisé.

```
Ce texte est <u>souligné</u>.
```

Le code ci-dessus serait rendu de la manière suivante :

Ce texte est souligné.

Intégration des images, fichiers SWF et des clips dans les champs de texte

Dans Flash Player 7 et les versions ultérieures, vous pouvez utiliser la balise `<img>` pour intégrer les fichiers JPEG, les fichiers SWF, ainsi que les clips, dans les champs de texte dynamique et de saisie. (pour obtenir la liste complète des attributs de la balise `<img>`, consultez *Balise image (<img>)*, page 156).

Par défaut, Flash affiche le média intégré dans un champ de texte, à sa taille normale. Pour spécifier les dimensions du média intégré, utilisez les attributs `height` et `width` de la balise `<img>`. Pour plus d'informations, consultez *Spécification des valeurs de hauteur et de largeur*, page 160.

En général, une image intégrée dans un champ de texte apparaît sur la ligne qui suit la balise `<img>`. Cependant, lorsque la balise `<img>` est le premier caractère dans le champ de texte, l'image apparaît sur la première ligne du champ de texte.

Intégration des fichiers SWF et JPEG

Pour intégrer un fichier JPEG ou SWF dans un champ de texte, spécifiez le chemin absolu ou relatif qui mène au fichier JPEG ou SWF dans l'attribut `src` de la balise `<img>`. Par exemple, le code suivant insère un fichier JPEG situé dans le même répertoire que le fichier SWF.

```
textField_txt.htmlText = "<p>Voici une photo de mes dernières vacances :<img
  src='plage.jpg'>";
```

Intégration de symboles de clip

Pour intégrer un symbole de clip dans un champ de texte, spécifiez l'identifiant de liaison du symbole pour l'attribut `src` de la balise `<img>` (pour plus d'informations sur la définition d'un identifiant de liaison, consultez *Association d'un symbole de clip à la scène*, page 133).

Par exemple, le code suivant insère un symbole de clip dont l'idenfiant de liaison est `symbol_ID`.

```
textField_txt.htmlText = "<p>Voici un symbole de clip :<img src='symbol_ID'>";
```

Pour qu'un clip intégré soit correctement et entièrement affiché, le point d'alignement de son symbole doit être (0,0).

Spécification des valeurs de hauteur et de largeur

Si vous spécifiez les attributs `width` et `height` d'une balise `<img>`, un espace est réservé dans le champ de texte pour le fichier JPEG, SWF ou le clip. Une fois le fichier JPEG ou SWF entièrement téléchargé, il est affiché dans l'espace réservé. Flash modifie la taille du média en fonction des valeurs `height` et `width`.

Si vous ne spécifiez pas de valeurs `height` et `width`, aucun espace n'est réservé au média intégré. Une fois le fichier JPEG ou SWF téléchargé, Flash l'insère dans le champ de texte à sa taille normale et sépare de nouveau le texte autour de lui.

Contrôle du média intégré avec ActionScript

Flash Player crée un nouveau clip pour chaque balise `<img>` et l'intègre dans l'objet TextField. L'attribut `id` de la balise `<img>` permet d'affecter un nom d'occurrence au clip créé et de contrôler ainsi le clip avec ActionScript.

Le clip créé par Flash Player est ajouté en tant que clip enfant du champ de texte contenant l'image.

Par exemple, le code suivant intègre un fichier SWF nommé animation.swf dans le champ de texte nommé `textField_txt` au niveau 0 et affecte le nom d'occurrence `animation_mc` au clip contenant le fichier SWF.

```
_level0.textField_txt.htmlText = "Voici une animation intéressante : <img
  src='animation.swf' id='animation_mc'>
```

Dans ce cas, le chemin entièrement qualifié vers le nouveau clip créé est `_level0.textField_txt.animation_mc`. Vous pouvez, par exemple, associer le code suivant à un bouton (dans le même scénario que `textField_txt`) afin d'arrêter la tête de lecture du fichier SWF intégré.

```
on(press) {
  textField_txt.animation_mc.stop();
}
```

Création d'hyperliens à partir d'un média intégré

Pour créer un hyperlien à partir d'un fichier JPEG, SWF ou d'un clip intégré, incluez la balise `<img>` dans une balise `<a>` :

```
textField.htmlText = "Cliquez sur l'image pour retourner à la page d'accueil<a
  href='accueil.htm'><img src='accueil.jpg'></a>";
```

Lorsque le pointeur de la souris survole une image, un fichier SWF ou un clip que vous avez placé entre des balises `<a>`, il prend la forme d'une « main », à l'instar des hyperliens standard. L'interactivité, telle que les clics de souris et la pression sur les touches du clavier, n'est pas enregistrée dans les fichiers SWF et les clips placés entre les balises `<a>`.

Création de texte défilant

Il existe plusieurs manières de créer du texte défilant dans Flash. Pour faire défiler des champs de texte dynamique et de saisie, vous pouvez sélectionner l'option Défilant dans le menu Texte ou le menu contextuel, ou double-cliquer sur la poignée du bloc de texte tout en maintenant la touche Maj enfoncée.

Vous pouvez utiliser les propriétés `scroll` et `maxscroll` de l'objet TextField pour contrôler le défilement vertical et les propriétés `hscroll` et `maxhscroll` pour contrôler le défilement horizontal d'un bloc de texte. Les propriétés `scroll` et `hscroll` spécifient respectivement les positions de défilement vertical et horizontal ; vous pouvez lire et rédiger ces propriétés. Les propriétés `maxscroll` et `maxhscroll` spécifient respectivement les positions verticales et horizontales maximales ; vous pouvez uniquement lire ces propriétés.

Le composant TextArea de Flash MX 2004 offre un moyen aisé de créer un champ de texte défilant avec un minimum de programmation. Pour plus d'informations, consultez l'entrée relative au composant TextArea, dans le guide Utilisation des composants de l'aide.

Pour créer un bloc de texte dynamique défilant, effectuez l'une des opérations suivantes :

- Double-cliquez sur la poignée du bloc de texte dynamique tout en maintenant la touche maj enfoncée.
- Sélectionnez le bloc de texte dynamique à l'aide de l'outil Flèche et choisissez Texte > Défilant.
- Sélectionnez le bloc de texte dynamique avec l'outil Flèche. Cliquez avec le bouton droit de la souris (Windows) ou en appuyant sur la touche Contrôle (Macintosh) sur le bloc de texte dynamique, puis sélectionnez Texte > Défilant.

Pour utiliser la propriété scroll afin de créer du texte défilant :

1. Effectuez l'une des opérations suivantes :
   - Sélectionnez l'outil Texte et tracez un champ de texte sur la scène. Affectez le nom d'occurrence `textField` au champ de texte dans l'inspecteur des propriétés.
   - Utilisez ActionScript pour créer un champ de texte dynamiquement à l'aide de la méthode `MovieClip.createTextField()`. Affectez le nom d'occurrence `textField` au champ de texte en tant que paramètre de la méthode.
2. Créez un bouton Vers le haut et un bouton Vers le bas ou choisissez Fenêtre > Autres panneaux > Bibliothèques communes > Boutons, puis faites glisser vos boutons sur la scène.

   Vous utiliserez ces boutons pour faire défiler le texte vers le haut et vers le bas.
3. Sélectionnez le bouton Vers le bas sur la scène.

4 Dans le panneau Actions (Fenêtre > Panneaux de développement > Actions), entrez le code suivant pour faire défiler le texte vers le bas dans le champ de texte :

```
on(press) {
  textField.scroll += 1;
}
```

5 Sélectionnez le bouton Vers le haut sur la scène.

6 Dans le panneau Actions, entrez le code suivant pour faire défiler le texte vers le haut :

```
on(press) {
  textField.scroll += 1;
}
```

CHAPITRE 9
Création de classes avec ActionScript 2.0

ActionScript 2.0 est une restructuration du langage ActionScript. Il offre de nouvelles fonctions de programmation, déjà disponibles dans d'autres langages, tels que Java. ActionScript 2.0 encourage les structures de programme réutilisables, évolutives, robustes et pouvant être gérées. Il permet également de réduire le temps imparti au développement, en offrant aux utilisateurs une aide à la programmation et des informations de débogage approfondies. ActionScript 2.0 respecte les normes existantes. Il est basé sur le projet ECMAScript 4 (www.mozilla.org/js/language/es4/). ActionScript 2.0 est disponible dans Macromedia Flash MX 2004 et Macromedia Flash MX Professionnel 2004.

Les fonctionnalités d'ActionScript 2.0 sont décrites ci-dessous.

Modèle familier de programmation orientée objet (OOP) La principale fonction d'ActionScript 2.0 est un modèle familier pour la création de programmes orientés objet. ActionScript 2.0 introduit plusieurs nouveaux concepts et mots-clés orientés objet, tels que *class*, *interface* et *packages* qui vous sembleront familiers si vous avez déjà programmé en Java.

Le modèle OOP d'ActionScript 2.0 est une « formalisation syntaxique » de la méthode de chaînage de prototype utilisée dans les précédentes versions de Macromedia Flash pour créer des objets et établir une relation d'héritage.

Typage strict des données ActionScript 2.0 vous permet également de spécifier explicitement des types de données pour les variables, paramètres de fonction et types de retour de fonction. Par exemple, le code suivant déclare une variable appelée `nomDutilisateur` de type chaîne (un type de données ActionScript intégré ou une classe).

```
var nomDutilisateur:String = "";
```

Avertissements et messages d'erreur du compilateur Les deux précédentes fonctions permettent à l'outil de programmation et au compilateur de générer des avertissements et messages d'erreur qui vous aident à localiser les bogues de vos applications dans Flash plus rapidement qu'auparavant.

Attention : Si vous souhaitez utiliser la syntaxe d'ActionScript 2.0, vérifiez que les paramètres de publication du fichier FLA indiquent ActionScript 2.0. Il s'agit de la configuration par défaut pour les fichiers créés dans Flash MX 2004. Toutefois, si vous ouvrez un ancien fichier FLA qui utilise ActionScript 1 et que vous commencez à le réécrire dans ActionScript 2.0, vous devez régler les paramètres de publication du fichier FLA sur ActionScript 2.0. Sinon, votre fichier FLA ne sera pas compilé correctement et aucune erreur ne sera générée.

Principes de la programmation orientée objet

Cette section est une rapide introduction aux principes du développement de programmes orientés objet. Ces principes sont présentés de façon plus approfondie dans le présent chapitre ; leur implémentation dans Macromedia Flash MX 2004 et Macromedia Flash MX Professionnel 2004 est notamment expliquée en détail.

Objets

Pensez à un « objet » du monde réel, un chat, par exemple. Supposons qu'un chat ait des propriétés (ou états) telles que nom, âge et couleur, et des comportements tels que dormir, manger et ronronner. Dans le monde de la programmation orientée objet, les objets ont également des propriétés et des comportements. En utilisant les techniques de programmation orientée objet, vous pouvez modeler un objet du monde réel (comme un chat) ou quelque chose de plus abstrait (un processus chimique, par exemple).

Classes et membres de classe

Continuons, si vous le voulez bien, avec l'analogie du monde réel. Supposons qu'il existe des chats de couleur, d'âge et de nom différents, qui mangent et ronronnent de façon différente. Mais tous les chats appartiennent à une certaine classe d'objet, un objet de type « chat ». Chaque chat (monde réel), considéré individuellement, est une occurrence du type de la classe chat.

De la même façon, dans la programmation orientée objet, une *classe* définit un modèle pour un type d'objet. Les caractéristiques et comportements qui appartiennent à une classe sont appelés *membres* de cette classe. Les caractéristiques (dans l'exemple du chat, nom, âge et couleur) sont appelées *propriétés* de la classe. Elles sont représentées sous forme de variables. Les comportements (manger, dormir) sont appelés *méthodes* de la classe ; ils sont représentés sous forme de fonctions.

Par exemple, vous pouvez créer une classe Personne, puis créer une personne individuelle correspondant à une occurrence de cette classe, appelée objet Personne. L'objet Personne contient l'ensemble des propriétés et méthodes de la classe Personne.

Dans ActionScript, vous définissez une classe à l'aide de l'instruction `class` (consultez *Création et utilisation de classes*, page 169). ActionScript inclut plusieurs classes intégrées, telles que les classes MovieClip, TextField et String. Pour plus d'informations, consultez le Chapitre 6, *Utilisation des classes intégrées*, page 119.

Héritage

L'un des principaux avantages de la programmation orientée objet est que vous pouvez créer des *sous-classes* de classe. La sous-classe *hérite* alors de l'ensemble des propriétés et méthodes de la *superclasse.* La sous-classe définit en général des méthodes et propriétés supplémentaires ou *permet d'étendre* la superclasse. Les sous-classes peuvent également *supplanter* (apporter leurs propres définitions) des méthodes héritées d'une superclasse.

Par exemple, vous créez une classe Mammifère qui définit certaines propriétés et certains comportements communs à tous les mammifères. Vous pouvez alors créer une classe Chat qui permet d'étendre la classe Mammifère. De cette manière, l'héritage peut promouvoir la réutilisation du code : au lieu de créer à nouveau le code commun aux deux classes, il vous suffit d'étendre la classe existante. Une autre sous-classe peut, à son tour, étendre la classe Chat, et ainsi de suite. Dans une application complexe, la définition de la structure hiérarchique des classes représente une grande partie du processus de création.

Dans ActionScript, utilisez le mot-clé `extends` pour créer une relation d'héritage entre une classe et sa superclasse. Pour plus d'informations, consultez *Création de sous-classes*, page 171.

Interfaces

Les interfaces, dans le cadre de la programmation orientée objet, peuvent être décrites comme des classes dont les méthodes ne sont pas implémentées (définies). Une autre classe peut implémenter les méthodes déclarées par l'interface.

Une interface peut également être vue comme un « contrat de programmation » pouvant être utilisé pour appliquer des relations entre des classes sans lien. Par exemple, supposons que vous travaillez avec une équipe de programmeurs et que chacun de vous travaille sur une partie (classe) différente de la même application. Lors de la réalisation de l'application, vous vous mettez d'accord sur un ensemble de méthodes que les différentes classes utiliseront pour communiquer. Ainsi, vous créez une interface qui déclare ces méthodes, leurs paramètres et leurs types de retour. Toute classe qui implémente cette interface doit fournir des définitions pour ces méthodes ; dans le cas contraire, une erreur du compilateur se produira.

Vous pouvez également utiliser les interfaces pour créer une forme limitée d'« héritage multiple », ce qui n'est pas autorisé dans ActionScript 2.0. Dans le cas d'un héritage multiple, une classe s'étend sur plusieurs classes. Par exemple, en C++, la classe Chat peut étendre la classe Mammifère, ainsi que la classe Espiègle, qui a les méthodes CourirAprèsQueue et FaireUnSomme. ActionScript 2.0, comme Java, n'autorise pas qu'une classe étende plusieurs classes directement. Toutefois, vous pouvez créer une interface Espiègle qui déclare les méthodes CourirAprèsQueue et FaireUnSomme. Une classe Chat ou toute autre classe peut alors implémenter cette interface et fournir des définitions pour ces méthodes.

Pour plus d'informations, consultez *Création d'une interface*, page 176.

Utilisation des classes : un exemple simple

Pour ceux qui débutent dans la programmation orientée objet, cette section est une présentation des tâches liées à la création et à l'utilisation des classes dans Flash. Le flux de travail implique les étapes suivantes (au minimum) :

1. Définition d'une classe dans un fichier de classe ActionScript externe.
2. Enregistrement du fichier de classe dans un répertoire de chemin de classe désigné (un emplacement où Flash va chercher les classes).
3. Création d'une occurrence de la classe dans un autre script, soit dans un document Flash (FLA), soit dans un fichier de script externe, ou création d'une sous-classe basée sur la classe d'origine.

Cette section présente également une nouvelle fonction d'ActionScript 2.0, appelée *typage strict des données*. Elle vous permet de spécifier le type de données d'une variable, d'un paramètre de fonction ou d'un type de renvoi de fonction.

Bien que cette section n'aborde que les classes, le déroulement général du travail est le même pour l'utilisation des interfaces. Pour plus d'informations, consultez *Création et utilisation d'interfaces*, page 175.

Création d'un fichier de classe

Pour créer une classe, vous devez tout d'abord créer un fichier ActionScript (AS) externe. Classes (et interfaces) peuvent uniquement être définies dans des fichiers de script externes. Par exemple, vous ne pouvez pas définir une classe dans un script associé à une image ou à un bouton dans un document Flash (FLA). Pour créer un fichier AS externe, utilisez l'éditeur ActionScript inclus dans Flash, ou l'éditeur de code ou de texte de votre choix.

Remarque : Le code ActionScript dans les fichiers externes est compilé dans un fichier SWF lors de la publication, de l'exportation, du test ou du débogage d'un fichier FLA. 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.

Dans les étapes suivantes, vous allez créer une classe appelée Personne, contenant deux propriétés (`âge` et `nom`) et une seule méthode (`showInfo()`) qui affiche les valeurs de ces propriétés dans le panneau de sortie.

Pour créer le fichier de classe :

1 Créez un nouveau répertoire sur votre disque dur et nommez-le FichiersPersonne. Ce répertoire comprendra tous les fichiers de ce projet.

2 Effectuez l'une des opérations suivantes :

- Créez un fichier dans l'éditeur de texte ou de code de votre choix.
- (Flash MX Professionnel uniquement) Pour ouvrir la boîte de dialogue Nouveau document, choisissez Fichier > Nouveau, choisissez Fichier ActionScript dans la liste des types de fichier, puis cliquez sur OK. La fenêtre de script s'ouvre sur un fichier vide.

3 Enregistrez le fichier sous Personne.as dans le répertoire FichiersPersonne.

4 Dans la fenêtre de script, entrez le code suivant :

```
class Personne {
}
```

Cela s'appelle la *déclaration* de classe. Dans sa forme la plus simple, une déclaration de classe comprend le mot-clé `class`, suivi du nom de la classe (Personne, dans ce cas), puis des accolades d'ouverture et de fermeture (`{}`). Tout ce qui est compris dans les accolades est appelé *corps* de la classe et c'est à cet endroit que les propriétés et méthodes de la classe sont définies.

Remarque : Le nom de la classe (Personne) correspond au nom du fichier AS qui la contient (Personne.as). Cela est très important. Si ces deux noms ne correspondent pas, la compilation de classe échoue.

5 Pour créer les propriétés de la classe Personne, utilisez le mot-clé `var` pour définir deux variables appelées `âge` et `nom`, comme indiqué ci-dessous.

```
class Personne {

  var âge:Number;
  var nom:String;

}
```

Conseil : Par convention, les propriétés de la classe sont définies au-dessus du corps de la classe, ce qui facilite la compréhension du code. Mais ce n'est pas obligatoire.

Notez bien la présence des deux-points dans la syntaxe (`var âge:Number` et `var nom:String`) utilisée pour la déclaration des variables. Ceci est un exemple de typage strict des données. Lorsque vous tapez une variable de cette façon, (`var nomDeVariable:TypeDeVariable`), le compilateur ActionScript 2.0 s'assure que la valeur affectée à cette variable correspond au type spécifié. Bien que cette syntaxe ne soit pas obligatoire, il est recommandé de la respecter. Elle peut d'ailleurs vous permettre de déboguer plus facilement vos scripts. Pour plus d'informations, consultez *Typage strict des données*, page 40.

6 Ensuite, créez la méthode `showInfo()`, qui renvoie une chaîne pré-formatée contenant les valeurs des propriétés `âge` et `nom`. Ajoutez la définition de fonction `showInfo()` au corps de la classe, comme indiqué ci-dessous.

```
class Personne {

  var âge:Number;
  var nom:String;

  // Méthodes pour renvoyer les valeurs des propriétés
  function showInfo():String {
    return("Bonjour, je m'appelle" + nom + " et j'ai " + âge + " ans.");
  }
}
```

Notez l'utilisation du typage des données (facultatif, mais recommandé) dans la définition de la fonction.

```
function showInfo():String {...}
```

Dans ce cas, c'est la valeur renvoyée (chaîne) de la fonction `showInfo()` qui est typée.

7 La dernière partie de code que vous ajoutez dans cette section concerne une fonction spéciale appelée *fonction constructeur*. En programmation orientée objet, la fonction constructeur initialise toutes les nouvelles occurrences d'une classe.

La fonction constructeur a toujours le même nom que la classe. Pour créer la fonction constructeur de la classe, ajoutez le code suivant :

```
class Personne {

  var âge:Number;
  var nom:String;

  // Méthodes pour renvoyer les valeurs des propriétés
  function showInfo():String {
    return("Bonjour, je m'appelle " + nom + " et j'ai " + âge + " ans.");
  }

  // Fonction constructeur
  function Personne (monNom:String, monAge:Number) {
    nom = monNom;
    âge = monAge;
  }
}
```

La fonction constructeur `Personne()` prend en compte deux paramètres, `monNom` et `monAge`, et affecte ces paramètres aux propriétés `nom` et `âge`. Les deux paramètres de la fonction sont strictement typés, respectivement comme chaîne et nombre. Pour plus d'informations sur les fonctions constructeur, consultez *Fonctions constructeur*, page 172.

Remarque : Si vous ne créez pas de fonction constructeur, une fonction constructeur vide est automatiquement créée pendant la compilation.

8 Enregistrez le fichier sous Personne.as dans le répertoire FichiersPersonne que vous avez créé à l'étape 1.

Si vous utilisez Flash MX 2004 (et non Flash Professionnel), passez à la section suivante.

9 (Flash Professionnel uniquement) Vérifiez la syntaxe du fichier de classe en choisissant Outils > Vérifier la syntaxe, ou en appuyant sur Ctrl+T (Windows) ou Commande+T (Macintosh).

Si des erreurs sont signalées dans le panneau de sortie, comparez le code de votre script au code final de l'étape 7, ci-dessus. Si vous ne pouvez pas corriger les erreurs de code, copiez le code terminée à l'étape 7, à partir de panneau d'aide.

Création d'une occurrence de la classe Personne

L'étape suivante permet de créer une occurrence de la classe Personne dans un autre script, tel qu'un script d'image dans un document Flash (FLA) ou dans un autre script AS, et de l'affecter à une variable. Pour créer une occurrence de classe personnalisée, utilisez l'opérateur `new`, comme pour créer une occurrence de classe ActionScript intégrée (telle que les classes XML ou TextField).

Par exemple, le code suivant crée une occurrence de la classe Personne et l'affecte à la variable `nouvellePersonne`.

```
var nouvellePersonne:Person = new Personne("Nadine", 32);
```

Cette classe invoque la fonction constructeur de la classe Personne, en transmettant comme paramètres les valeurs `"Nadine"` et 32.

La variable `nouvellePersonne` est typée en tant qu'objet Personne. Cette manière de typer les objets permet au compilateur de s'assurer que vous n'essayez pas d'accéder aux propriétés ou aux méthodes qui ne sont pas définies dans cette classe (sauf si vous déclarez que la classe est dynamique en utilisant le mot clé `dynamic`). Pour plus d'informations, consultez *Création de classes dynamiques*, page 182.

Pour créer une occurrence de la classe Personne dans un document Flash :

1 Dans Flash, choisissez Fichier > Nouveau, choisissez Document Flash dans la liste des types de documents, et cliquez sur OK.

2 Enregistrez le fichier sous créerPersonne.fla dans le répertoire FichiersPersonne que vous avez créé précédemment.

3 Dans le scénario, choisissez Calque 1, puis ouvrez le panneau Actions (Fenêtre > Panneaux de développement > Actions).

4 Dans le panneau Actions, entrez le code suivant :

```
var Personne_1:Person = new Personne("Nadine", 32);
var personne_2:Person = new Personne("Julie", 28);
trace(personne_1.showInfo());
trace(personne_2.showInfo());
```

Le code ci-dessus permet de créer deux occurrences de la classe Personne, `personne_1` et `personne_2`, et d'appeler la méthode `showInfo()` sur chaque occurrence.

5 Enregistrez votre travail, puis choisissez Contrôle > Tester l'animation. Les informations suivantes doivent s'afficher dans le panneau de sortie :

```
Bonjour, je m'appelle Nadine et j'ai 32 ans.
Bonjour, je m'appelle Julie et j'ai 28 ans.
```

Lorsque vous créez une occurrence de classe en appelant sa fonction constructeur, Flash recherche un fichier AS portant le même nom que le constructeur dans un ensemble prédéterminé d'emplacements de répertoires. Cet ensemble d'emplacements de répertoires est appelé *chemin de classe* (consultez *Compréhension du chemin de classe*, page 177).

A présent, vous devriez avoir une vision globale de la création et de l'utilisation des classes dans les documents Flash. La suite de ce chapitre aborde plus en détail les classes et les interfaces.

Création et utilisation de classes

Comme nous avons l'avons vu précédemment, une classe se compose de deux parties : la *déclaration* et le *corps*. La déclaration de la classe comporte au minimum l'instruction `class`, suivie de l'identifiant du nom de la classe, puis des accolades d'ouverture et de fermeture. Tout ce qui est compris dans les accolades constitue le corps de la classe.

```
class nomDeLaClasse {
  // corps de la classe
}
```

Vous pouvez définir des classes uniquement dans des fichiers ActionScript (AS). Par exemple, vous ne pouvez pas définir de classe sur un script d'image dans un fichier FLA. De même, le nom de la classe spécifiée doit correspondre au nom du fichier AS qui contient la classe. En ce sens, si vous créez une classe Shape, le fichier AS qui contient la définition de la classe doit s'appeler Shape.as.

```
// Dans le fichier Shape.as
class Shape {
  // Corps de la classe Shape
}
```

Tous les fichiers de classe AS que vous créez doivent être enregistrés dans l'un des répertoires désignés du chemin de classe (répertoires dans lesquels Flash recherche les définitions de classe lors de la compilation des scripts). Pour plus d'informations, consultez *Compréhension du chemin de classe*, page 177.

Les noms de classes doivent être des identifiants ; ainsi, le premier caractère doit être une lettre, un soulignement (_), ou le signe dollar ($), et chaque caractère suivant doit être une lettre, un nombre, ou le signe dollar. En outre, le nom de classe doit être entièrement qualifié dans le fichier dans lequel il est déclaré : il doit indiquer le nom du répertoire dans lequel il est enregistré. Par exemple, pour créer une classe nommée ClasseRequise enregistrée dans le répertoire myClasses/education/curriculum, vous devez déclarer cette classe dans le fichier ClasseRequise.as de la manière suivante :

```
classe myClasses.education.curriculum.ClasseRequise {
}
```

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.

Création de propriétés et de méthodes

Les membres d'une classe se composent des propriétés (déclarations de variables) et des méthodes (déclarations de fonctions). Vous devez déclarer toutes les propriétés et méthodes dans le corps de la classe (entre les accolades) ; sinon une erreur se produira lors de la compilation.

Toute variable déclarée dans une classe, mais en dehors d'une fonction, est une propriété de la classe. Par exemple, la classe Personne, dont il était question précédemment, a deux propriétés, `âge` et `nom`, respectivement de type chaîne et nombre.

```
class Personne {
  var âge:Number;
  var nom:String;
}
```

De la même façon, toute fonction déclarée dans une classe est considérée comme étant une méthode de cette classe. Dans l'exemple de la classe Personne, vous aviez créé une méthode unique appelée `showInfo()`.

```
class Personne {
  var âge:Number;
  var nom:String;
  function showInfo() {
    // définition de la méthode showInfo()
  }
}
```

Initialisation de propriétés en ligne

Vous pouvez initialiser des propriétés *en ligne*, c'est-à-dire lorsque vous les déclarez, avec des valeurs par défaut, comme l'illustre l'exemple suivant :

```
class Personne {
  var âge:Number = 50;
  var nom:String = "Jean Dubuste";
}
```

Lorsque vous initialisez des propriétés en ligne, l'expression du côté droit de l'affectation doit être une *constante de compilation*. C'est à dire que l'expression ne peut pas faire référence à un élément paramétré ou défini au moment de l'exécution. Les constantes de compilation comprennent les chaînes littérales, les nombres, les valeurs booléennes, `null` et `undefined`, ainsi que les fonctions constructeur pour les classes intégrées suivantes : Array, Boolean, Number, Object et String.

Par exemple, la définition de classe suivante initialise plusieurs propriétés en ligne :

```
class CompileTimeTest {
  var truc:String = "mon truc"; // OK
  var barre:Number = 5; // OK
  var bool:Boolean = true; // OK
  var nom:String = new String("Julie"); // OK
  var qui:String = truc; // OK, car 'truc' est une constante

  var whee:String = maFonc(); // erreur ! il ne s'agit pas d'une expression
  constante de compilation
  var lala:Number = whee; // erreur ! il ne s'agit pas d'une expression
  constante de compilation
  var star:Number = bar + 25; // OK, 'bar' et '25' sont des constantes

  function maFonc() {
    return "Bonjour monde";
  }
}
```

Cette règle s'applique uniquement aux variables d'occurrence (variables copiées dans chaque occurrence d'une classe). Elle ne s'applique pas aux variables de classe (variables qui appartiennent véritablement à la classe). Pour plus d'informations sur ces variables, consultez *Membres d'occurrence et de classe*, page 173.

Création de sous-classes

Dans la programmation orientée objet, une sous-classe peut hériter des propriétés et méthodes d'une autre classe, appelée superclasse. Pour créer ce type de relation entre deux classes, utilisez la clause `extends` de l'instruction `class`. Pour spécifier une superclasse, utilisez la syntaxe suivante :

```
class SubClass extends SuperClasse {}
```

La classe que vous avez spécifiée dans *`SubClass`* hérite de toutes les propriétés et méthodes définies par la superclasse. Par exemple, vous créez une classe Mammifère qui définit des propriétés et méthodes communes à tous les mammifères. Pour créer une variante de cette classe Mammifère, telle que la classe Marsupial, étendez la classe Mammifère, c'est-à-dire que vous allez créer une sous-classe de la classe Mammifère.

```
class Marsupial extends Mammifère {}
```

La sous-classe hérite de toutes les propriétés et méthodes de la superclasse, y compris des propriétés ou des méthodes que vous avez déclarées comme étant privées en utilisant le mot-clé `private`. (Pour plus d'informations sur les variables privées, consultez *Contrôle de l'accès des membres*, page 172.)

Vous pouvez étendre vos propres classes personnalisées, ainsi que toute classe ActionScript intégrée, telle que XML, Sound ou MovieClip. Lorsque vous étendez une classe ActionScript intégrée, votre classe personnalisée hérite de toutes les méthodes et propriétés de la classe intégrée.

Par exemple, le code suivant définit la classe JukeBox, qui étend la classe Son intégrée. Il définit un tableau appelé `listeChansons` et une méthode appelée `litChanson()` qui permet de lire une chanson et d'invoquer la méthode `chargeSon()`, dont il hérite de la classe Son.

```
class JukeBox extends Son {
  var listeChansons:Array = new Array("beethoven.mp3", "bach.mp3",
  "mozart.mp3");
  function litChanson(songID:Number) {
    this.chargeSon(listeChansons[songID]);
  }
}
```

Si vous ne placez pas un appel à `super()` dans la fonction constructeur d'une sous-classe, le compilateur génère automatiquement un appel au constructeur de sa superclasse immédiate, sans paramètre, en tant que première instruction de la fonction. Si la superclasse n'a pas de constructeur, le compilateur crée une fonction constructeur vide, puis génère un appel à cette fonction à partir de la sous-classe. Cependant, si une super-classe prend des paramètres dans sa définition, vous devez créer un constructeur dans la sous-classe et appeler la super-classe avec les paramètres requis.

L'héritage multiple ou l'héritage à partir de plus d'une classe n'est pas autorisé. Toutefois, les classes peuvent effectivement hériter de plusieurs classes, si vous utilisez des instructions `extends` individuelles :

```
// non autorisé
class C extends A, B {}
// autorisé
class B extends A {}
class C extends A, B {}
```

Vous pouvez également utiliser le mot-clé `extends` pour créer les sous-classes d'une interface :

```
interface iA extends interface iB {}
```

Fonctions constructeur

Un *constructeur* de classe est une fonction spéciale appelée automatiquement lorsque vous créez une occurrence de classe en utilisant l'opérateur `new`. La fonction constructeur porte le même nom que la classe qui la contient. Par exemple, la classe Personne créée plus haut contenait la fonction constructeur suivante :

```
// Fonction constructeur de la classe Personne
function Personne (monNom:String, monAge:Number) {
  nom = monNom;
  âge = monAge;
}
```

Si aucune fonction constructeur n'est explicitement déclarée, c'est-à-dire, si vous ne créez pas de fonction dont le nom correspond à celui de la classe, le compilateur crée automatiquement une fonction constructeur vide.

Une classe ne peut contenir qu'une fonction constructeur ; les fonctions constructeur surchargées ne sont pas autorisées dans ActionScript 2.0.

Contrôle de l'accès des membres

Par défaut, toute propriété ou méthode de classe est accessible à toute autre classe : tous les membres d'une classe sont considérés, par défaut, comme *publics*. Toutefois, dans certains cas, vous pouvez souhaiter que d'autres classes n'aient pas accès aux données ou aux méthodes d'une classe. Vous devez alors faire en sorte que ces membres deviennent *privés*, c'est-à-dire disponibles uniquement pour la classe qui les déclare ou qui les définit.

Pour spécifier des membres publics ou privés, utilisez l'attribut de membre `public` ou `private`. Par exemple, le code suivant déclare une variable privée (une propriété) et une méthode privée (une fonction).

Par exemple, la classe suivante (LoginClass) définit une propriété privée nommée `nomDutilisateur` et une méthode privée nommée `getNomDutilisateur()`.

```
classe LoginClass {
  private var nomDutilisateur:String;
  private function getNomDutilisateur() {
    return this.userName;
  }
//constructeur :
  function LoginClass(user:String) {
    this.userName = user;
  }
}
```

Les membres privés (propriétés et méthodes) sont uniquement accessibles à la classe qui définit ces membres et aux sous-classes de cette classe d'origine. Les occurrences de la classe d'origine ou celles des sous-classes de cette classe ne peuvent pas accéder aux propriétés et méthodes déclarées en privé ; c'est-à-dire que les membres privés ne sont accessibles qu'au sein des définitions de classe, et non au niveau des occurrences.

Par exemple, vous pouvez créer une sous-classe de la classe LoginClass nommée NewLoginClass. Cette sous-classe peut accéder à la propriété (`nomDutilisateur`) et à la méthode (`getnomDutilisateur()`) privées définies par LoginClass.

```
class NewLoginClass extends LoginClass {
  // peut accéder à nomDutilisateur et getnomDutilisateur()
}
```

Cependant, une occurrence de LoginClass ou de NewLoginClass ne peut pas accéder aux membres privés. Par exemple, le code suivant, ajouté à un script d'image dans un fichier FLA, engendrerait une erreur du compilateur indiquant que `getnomDutilisateur()` est privé et qu'il est impossible d'y accéder.

```
var loginObject:LoginClass = new LoginClass("Maxwell");
var utilisateur = loginObject.getnomDutilisateur();
```

Notez également que le contrôle de l'accès des membres est une fonction de compilation uniquement ; à l'exécution, Flash Player ne fait aucune distinction entre les membres publics et privés.

Membres d'occurrence et de classe

Dans la programmation orientée objet, les membres (propriétés ou méthodes) d'une classe peuvent être soit des *membres d'occurrence*, soit des *membres de classe*. Les membres d'occurrence sont créés pour chaque occurrence de la classe et sont copiés dans chacune d'entre elles. En revanche, les membres de classe sont créés une seule fois par classe (ils sont également appelés *membres statiques*).

Pour invoquer une méthode d'occurrence ou accéder à une propriété d'occurrence, faites référence à une occurrence de la classe. Par exemple, le code suivant permet d'invoquer la méthode `showInfo()` sur une occurrence de la classe MovieClip appelée `clip_mc`:

```
clip_mc.showInfo();
```

Toutefois, les membres de classe (statiques) sont affectés à la classe elle-même et non à n'importe quelle occurrence de la classe. Pour invoquer une méthode de classe ou accéder à une propriété de classe, faites référence au nom de la classe plutôt qu'à une occurrence spécifique de la classe :

```
nomDeLaClasse.membreDeLaClasse;
```

Par exemple, la classe Math ActionScript comprend uniquement des méthodes et des propriétés statiques. Pour appeler ses méthodes, au lieu de créer une occurrence de la classe Math, appelez tout simplement les méthodes de la classe Math. Le code suivant appelle la méthode `sqrt()` de la classe Math :

```
var racine_carrée:Number = Math.sqrt(4);
```

Les membres d'occurrence peuvent lire des membres statiques, mais ne peuvent pas les écrire. Les membres d'occurrence ne sont pas énumérables dans les boucles `for` ou `for..in`.

Création des membres de classe

Pour spécifier qu'une propriété de classe est statique, utilisez le modificateur `static`, comme indiqué ci-dessous.

```
static var nomDeLaVariable;
```

Vous pouvez également déclarer des méthodes de classe statiques.

```
static function nomDeLaFonction() {
  // corps de la fonction
}
```

Les méthodes de classe (statiques) peuvent accéder uniquement aux propriétés de classe (statiques). Elles n'ont pas accès aux propriétés d'occurrence. Par exemple, le code suivant génère une erreur de compilation, car la méthode de classe `getName()` fait référence au `nom` de variable de l'occurrence.

```
class StaticTest {
  var nom="Ted";

  static function getName() {
    var nom_local = name;
    // Erreur ! Il est impossible d'accéder aux variables d'occurrence dans
  des fonctions statiques.
  }
}
```

Pour résoudre ce problème, vous pouvez soit faire de la méthode une méthode d'occurrence, soit faire de la variable une variable de classe.

Utilisation de membres de classe : un exemple simple

Vous pouvez notamment utiliser des membres de classe (statiques) pour conserver les informations d'état sur une classe et ses occurrences. Supposons par exemple que vous souhaitez consigner le nombre d'occurrences créées à partir d'une classe donnée. Vous pouvez facilement y parvenir en utilisant une propriété de classe qui est incrémentée à chaque création d'une nouvelle occurrence.

Dans l'exemple suivant, vous créez une classe nommée Gadget qui définit un compteur d'occurrence statique unique nommé `compteurGadget`. A chaque création d'une nouvelle occurrence de la classe, la valeur de `compteurGadget` est incrémentée de 1 et la valeur courante de `compteurGadget` est affichée dans la panneau de sortie.

Pour créer un compteur d'occurrence en utilisant une variable de classe :

1 Créez un nouveau fichier ActionScript (AS).

2 Ajoutez le code suivant au fichier :

```
class Gadget {
  static var compteurGadget:Number = 0; // initialisez la variable de classe
  function Gadget() {
    trace("Création de gadget #" + compteurGadget);
    compteurGadget++;
  }
}
```

La variable `compteurGadget` est déclarée comme statique, et est initialisée à 0 une seule fois. Chaque fois que la fonction constructeur de la classe Gadget est appelée, elle ajoute 1 à `compteurGadget` puis affiche le numéro de l'occurrence en cours de création.

3 Enregistrez votre fichier sous Gadget.as.

4 Créez un nouveau document Flash (FLA) et enregistrez-le sous createGadget.fla dans le même répertoire que Gadget.as.

Dans ce fichier, vous créerez de nouvelles occurrences de la classe Gadget.

5 Dans createGadget.fla, sélectionnez le calque 1 dans le scénario et ouvrez le panneau Actions (Fenêtre > Panneaux de développement > Actions).

6 Ajoutez le code suivant au panneau Actions.

```
//Avant de créer une occurrence de la classe,
//compteurGadget est à zéro (0)
trace("Compteur Gadget au démarrage : " + Gadget.compteurGadget);
var gadget_1 = new Gadget();
var gadget_2 = new Gadget();
var gadget_3 = new Gadget();
```

7 Enregistrez le fichier, puis testez-le (Contrôle > Tester l'animation).

Les informations suivantes doivent s'afficher dans le panneau de sortie :

```
Compteur Gadget au démarrage : 0
Creating gadget # 0
Creating gadget # 1
Creating gadget # 2
```

Membres et sous-classes de classes

Les membres de classe se propagent dans les sous-classes de la superclasse qui définit ces membres. Dans l'exemple précédent, (consultez *Utilisation de membres de classe : un exemple simple*, page 174), vous avez utilisé une propriété de classe pour consigner le nombre d'occurrences de la classe que vous avez créée. Vous pouvez créer une sous-classe de la classe Gadget comme indiqué ci-dessous.

```
class SousGadget extends Gadget {
  function SousGadget() {
    trace("Création de sousgadget # "+Gadget.compteurGadget);
  }
}
```

Création et utilisation d'interfaces

En programmation orientée objet, une interface est similaire à une classe dont les méthodes ont été déclarées, mais n'effectuent aucune action. Une interface consiste en fait en des méthodes « vides ».

Vous pouvez notamment utiliser les interfaces pour appliquer un protocole entre des classes n'ayant aucun rapport entre elles, comme expliqué ultérieurement. Supposons par exemple que vous faites partie d'une équipe de programmeurs, dont chaque membre travaille sur une partie différente (c'est-à-dire une classe différente) d'une application volumineuse. La plupart de ces classes n'ont aucun rapport entre elles, mais il est tout de même nécessaire qu'elles puissent communiquer les unes avec les autres. Vous devez définir une interface, ou protocole de communication, à laquelle toutes les classes doivent adhérer.

Pour y parvenir, vous pouvez créer une classe Communication qui définit toutes ces méthodes, puis faire en sorte que chaque classe étende cette superclasse, ou en hérite. Mais étant donné que l'application se compose de classes n'ayant aucun rapport, il est inutile de toutes les placer dans une hiérarchie de classe commune. Il est préférable de créer une interface qui déclare les méthodes que ces classes utiliseront pour communiquer, puis que chaque classe implémente ces méthodes (fournisse ses propres définitions).

Vous pouvez généralement programmer de façon efficace sans utiliser les interfaces. Lorsqu'elles sont utilisées de façon appropriée, cependant, les interfaces peuvent rendre la conception de vos applications plus élégante, évolutive et stable.

Création d'une interface

Le processus de création d'une interface est identique au processus de création d'une classe. Comme pour les classes, vous pouvez uniquement définir des interfaces dans des fichiers ActionScript (AS) externes. Déclarez une interface en utilisant le mot-clé `interface` suivi du nom de l'interface et d'accolades gauche et droite qui définissent le corps de l'interface.

```
interface nomInterface {
  // déclarations de méthodes d'interface
}
```

Une interface ne peut contenir que des déclarations de méthodes (fonction), y compris des paramètres, des types de paramètres et des types de renvoi de fonction.

Par exemple, le code suivant déclare une interface nommée MonInterface qui contient deux méthodes, `method_1()` et `method_2()`. La première méthode ne prend aucun paramètre et n'a pas de type de renvoi (spécifiée comme `Void`). La deuxième déclaration de méthode ne prend qu'un seul paramètre de type String et spécifie un type de renvoi Boolean.

```
interface MonInterface {
  function method_1():Void;
  function method_2(param:String):Boolean;
}
```

Les interfaces ne peuvent contenir aucune déclaration ou affectation de variable. Les fonctions déclarées dans une interface de peuvent pas contenir d'accolades. Par exemple, l'interface suivante ne sera pas compilée.

```
interface MauvaiseInterface{
  // Erreur de compilation. Les déclarations de variables ne sont pas
  autorisées dans les interfaces.
  var illegalVar;

// Erreur de compilation. Les corps de fonctions ne sont pas autorisés dans les
  interfaces.
  function illegalMethod(){
  }
}
```

Les règles d'affectation de nom et de stockage des interfaces dans les paquets sont les mêmes que celles des classes ; consultez *Création et utilisation de classes*, page 169 et *Utilisation de paquets*, page 179.

Interfaces comme types de données

Tout comme une classe, une interface définit un nouveau type de données. Toute classe qui implémente une interface peut être considérée comme relevant du type défini par l'interface. Ceci est utile pour déterminer si un objet donné implémente une interface donnée. Considérons par exemple l'interface suivante.

```
interface Déplaçable {
  function monter();
  function descendre();
}
```

Considérons la classe Box qui implémente l'interface Déplaçable.

```
class Box implements Déplaçable {
  var x_pos, y_pos;

  function monter() {
    // définition de méthode
  }
  function descendre() {
    // définition de méthode
  }
}
```

Ensuite, dans un autre script où vous créez une occurrence de la classe Box, vous déclarez qu'une variable est de type Déplaçable.

```
var newBox:Movable = new Box();
```

A l'exécution, dans Flash Player 7 et ses versions ultérieures, vous pouvez attribuer une expression à un type d'interface. Si l'expression est un objet qui implémente l'interface, ou si elle possède une superclasse qui implémente l'interface, l'objet est renvoyé. Dans le cas contraire, `null` est renvoyé. Cela est particulièrement utile si vous voulez vous assurer qu'un objet particulier implémente une interface particulière.

Par exemple, le code suivant vérifie d'abord si le nom de l'objet `unObjet` implémente l'interface Déplaçable avant d'appeler la méthode `moveUp()` sur l'objet.

```
if(Movable(someObject) != null) {
  unObjet.moveUp();
}
```

Compréhension du chemin de classe

Pour utiliser une classe ou interface que vous avez définie, Flash doit être capable de localiser les fichiers AS externes qui contiennent la définition de classe ou d'interface. La liste des répertoires dans lesquels Flash recherche les définitions de classe et d'interface est appelée *chemin de classe*.

Lorsque vous créez un fichier de classe ActionScript, vous devez enregistrer le fichier dans l'un des répertoires spécifiés dans le chemin de classe, ou dans l'un de ses sous-répertoires. (Vous pouvez modifier le chemin de classe pour inclure le chemin souhaité ; consultez *Modification du chemin de classe*, page 178.) Sinon, Flash ne sera pas en mesure de *résoudre* ou de localiser la classe ou l'interface spécifiée dans le script. Les sous-répertoires que vous créez dans un répertoire de chemin de classe sont appelés *paquets*. Ils vous permettent d'organiser vos classes. Pour plus d'informations, consultez *Utilisation de paquets*, page 179.

Chemins de classe globaux et au niveau du document

Flash dispose de deux paramètres de chemin de classe : un chemin de classe global et un chemin de classe au niveau du document. Le chemin de classe global s'applique aux fichiers AS et FLA externes. Il est défini dans la boîte de dialogue Préférences (Edition > Préférences). Le chemin de classe au niveau du document s'applique uniquement aux fichiers FLA. Il est défini dans la boîte de dialogue Paramètres de publication (Fichier > Paramètres de publication) pour un fichier FLA donné.

Par défaut, le chemin de classe global contient deux chemins de répertoires : un chemin relatif qui pointe vers le répertoire contenant le document courant et un autre vers le répertoire Classes situé dans le répertoire de configuration utilisateur installé avec Flash. L'emplacement de ce répertoire est le suivant :

- Windows 2000 ou Windows XP : C:\Documents and Settings\<utilisateur>\Local Settings\Application Data\Macromedia\Flash MX2004\<langue>\Configuration\
- Windows 98 : C:\Windows\Application Data\Macromedia\Flash MX 2004\<langue>\Configuration\
- Mac OS X : Disque dur/Users/Library/Application Support/Macromedia/Flash MX 2004/<langue>/Configuration/

Le chemin de classe au niveau du document est vide par défaut.

Résolution des références de classe par le compilateur

Lorsque Flash tente de résoudre les références de classe dans un script FLA, il recherche tout d'abord le chemin de classe au niveau du document spécifié pour ce FLA. Si la classe n'est pas trouvée dans ce chemin de classe, ou si le chemin de classe est vide, Flash effectue une recherche dans le chemin de classe global. Si la classe n'est pas trouvée dans le chemin de classe global, une erreur de compilation se produit.

Lorsque Flash tente de résoudre les références de classe dans un script AS, il effectue uniquement une recherche dans les répertoires du chemin de classe global, étant donné que les fichiers AS ne possèdent pas de chemin de classe de document qui leur soit associé.

Modification du chemin de classe

Vous pouvez modifier le chemin de classe global à l'aide de la boîte de dialogue Préférences. Pour modifier le paramètre de chemin de classe au niveau du document, utilisez la boîte de dialogue Paramètres de publication pour le fichier FLA. Vous pouvez ajouter des chemins de répertoires absolus (par exemple C:/mes_classes) et des chemins de répertoires relatifs (par exemple, ..\mes_classes ou « . »).

Par défaut, le chemin de classe global contient un chemin absolu (le répertoire Classes dans le répertoire de configuration utilisateur) et un chemin de classe relatif, symbolisé par un point unique (.), qui pointe vers le répertoire du document en cours. Notez que les chemins de classe relatifs peuvent pointer vers des répertoires différents, en fonction de l'emplacement du document compilé ou publié. Pour plus d'informations, consultez *Chemins de classe globaux et au niveau du document*, page 178.

Pour modifier le chemin de classe global :

1 Choisissez Edition > Préférences pour ouvrir la boîte de dialogue Préférences.

2 Cliquez sur l'onglet ActionScript, puis cliquez sur le bouton Paramètres d'ActionScript 2.0.

3 Effectuez l'une des opérations suivantes :

- Pour ajouter un répertoire au chemin de classe, cliquez sur le bouton Rechercher le chemin, ouvrez le répertoire que vous souhaitez ajouter et cliquez sur OK.

  Vous pouvez également cliquer sur le bouton Ajouter un nouveau chemin (+) pour ajouter une ligne à la liste Chemin de classe. Double-cliquez sur la nouvelle ligne, tapez un chemin relatif ou absolu et cliquez sur OK.
- Pour modifier un répertoire de chemin de classe existant, sélectionnez le chemin dans la liste Chemin de classe, cliquez sur le bouton Rechercher le chemin, ouvrez le répertoire que vous souhaitez ajouter et cliquez sur OK.

  Vous pouvez également double-cliquer sur le chemin dans la liste Chemin de classe, taper le chemin désiré et cliquer sur OK.
- Pour supprimer un répertoire du chemin de classe, sélectionnez le chemin dans la liste Chemin de classe, et cliquez sur le bouton Supprimer du trajet.

Pour modifier le chemin de classe au niveau du document :

1 Choisissez Fichier > Paramètres de publication pour ouvrir la boîte de dialogue Paramètres de publication.

2 Cliquez sur l'onglet Flash.

3 Cliquez sur le bouton Paramètres en regard du menu déroulant Version d'ActionScript.

4 Effectuez l'une des opérations suivantes :

- Pour ajouter un répertoire au chemin de classe, cliquez sur le bouton Rechercher le chemin, ouvrez le répertoire que vous souhaitez ajouter et cliquez sur OK.

  Vous pouvez également cliquer sur le bouton Ajouter un nouveau chemin (+) pour ajouter une ligne à la liste Chemin de classe. Double-cliquez sur la nouvelle ligne, tapez un chemin relatif ou absolu et cliquez sur OK.
- Pour modifier un répertoire de chemin de classe existant, sélectionnez le chemin dans la liste Chemin de classe, cliquez sur le bouton Rechercher le chemin, ouvrez le répertoire que vous souhaitez ajouter et cliquez sur OK.

  Vous pouvez également double-cliquer sur le chemin dans la liste Chemin de classe, taper le chemin désiré et cliquer sur OK.
- Pour supprimer un répertoire du chemin de classe, sélectionnez le chemin dans la liste Chemin de classe, et cliquez sur le bouton Supprimer du trajet.

Utilisation de paquets

Vous pouvez organiser vos fichiers de classe ActionScript en *paquets*. Un paquet est un répertoire qui contient un ou plusieurs fichiers de classe et qui réside dans un répertoire de chemin de classe désigné. Pour plus d'informations, consultez *Compréhension du chemin de classe*, page 177. Un paquet peut également contenir d'autres paquets, appelés *sous-paquets*, chacun possédant ses propres fichiers de classe.

Les noms de paquets doivent être des identifiants ; ainsi, le premier caractère doit être une lettre, un soulignement (_), ou le signe dollar ($), et chaque caractère suivant doit être une lettre, un nombre, ou le signe dollar.

Les paquets sont généralement utilisés pour organiser des classes connexes. Vous pouvez par exemple avoir trois classes connexes, Carré, Cercle et Triangle, définies dans Carré.as, Cercle.as et Triangle.as. Supposons que vous ayez enregistré les fichiers AS dans un répertoire spécifié dans le chemin de classe.

```
// Dans Carré.as:
class Carré {}

// Dans Cercle.as:
class Cercle {}

// Dans Triangle.as:
class Triangle {}
```

Etant donné que ces trois classes sont connexes, vous pouvez choisir de les placer dans un paquet (répertoire) nommé Formes. Dans ce cas, le nom pleinement qualifié de la classe contient le chemin du paquet, ainsi que le nom de classe simple. Les chemins de paquet sont symbolisés par une syntaxe de point, chaque point représentant un sous-répertoire.

Par exemple, si vous placez tous les fichiers AS qui définissent une forme dans le répertoire Formes, vous devrez alors changer le nom de chaque fichier de classe pour répercuter le nouvel emplacement, comme suit :

```
// Dans Formes/Carré.as:
class Formes.Carré {}

// Dans Formes/Cercle.as:
class Formes.Cercle {}

// Dans Formes/Triangle.as:
class Formes.Triangle {}
```

Pour faire référence à une classe qui réside dans un répertoire de paquet, vous pouvez soit spécifier le nom pleinement qualifié de sa classe, soit importer le paquet en utilisant l'instruction `import` (voir ci-dessous).

Importation de classes

Pour faire référence à une classe dans un autre script, vous devez faire précéder le nom de la classe par le chemin de paquet de la classe. La combinaison du nom de la classe et de son chemin de paquet correspond au *nom de classe pleinement qualifié* de la classe. Si une classe réside dans un répertoire de chemin de classe de premier niveau (et non dans un sous-répertoire du répertoire de chemin de classe), son nom pleinement qualifié est tout simplement son nom de classe.

Pour spécifier des chemins de paquets, utilisez une notation à point pour séparer les noms de répertoires de paquets. Les chemins de paquets sont hiérarchiques, et chaque point représente un répertoire imbriqué. Supposons par exemple que vous créez une classe nommée Données qui réside dans un paquet com/réseau/ dans votre chemin de classe. Pour créer une occurrence de cette classe, vous pouvez spécifier le nom pleinement qualifié de la classe, de la manière suivante :

```
var occurrenceDonnées = new com.réseau.Data();
```

Vous pouvez utiliser le nom pleinement qualifié de la classe ou taper vos variables :

```
var occurrenceDonnées:com.réseau.Data = new Data();
```

Vous pouvez utiliser l'instruction `import` pour importer des paquets dans un script, ce qui vous permet d'utiliser le nom abrégé d'une classe à la place de son nom pleinement qualifié. Vous pouvez également utiliser le caractère générique (*) pour importer toutes les classes dans un paquet.

Supposons par exemple que vous créiez une classe nommée ClasseUtilisateur qui est incluse dans le répertoire de paquet macr.util.users :

```
//Dans le fichier macr/util/users/ClasseUtilisateur.as
class macr.util.users.ClasseUtilisateur { ... }
```

Supposons que dans un autre script, vous avez importé cette classe à l'aide de l'instruction `import` de la manière suivante :

```
import macr.util.users.ClasseUtilisateur;
```

Plus tard dans le même script vous pouvez faire référence à cette classe par son nom abrégé :

```
var monUtilisateur:ClasseUtilisateur = new ClasseUtilisateur();
```

Vous pouvez également utiliser le caractère générique (*) pour importer toutes les classes dans un paquet. Supposons par exemple que vous avez un paquet `macr.util` qui contient deux fichiers de classe ActionScript, machin.as et chose.as. Dans un autre script, vous pouvez importer les deux classes dans ce paquet en utilisant le caractère générique, comme montré ci-dessous.

```
import macr.util.*;
```

Dans le même script, vous pouvez ensuite faire directement référence à la classe machin ou chose.

```
var monMachin:machin = new machin();
var monChose:chose = new chose();
```

L'instruction `import` s'applique uniquement au script courant (image ou objet) dans lequel elle est appelée. Si une classe importée n'est pas utilisée dans un script, cette classe n'est pas incluse dans le pseudo-code binaire du fichier SWF résultant, et cette classe n'est pas disponible dans les fichiers SWF susceptibles d'être appelés par le fichier FLA contenant l'instruction `import`. Pour plus d'informations, consultez `import`, page 443.

Méthodes get/set implicites

La programmation orientée objet empêche l'accès direct aux propriétés à l'intérieur d'une classe. Les classes définissent généralement des méthodes « get » qui fournissent un accès en lecture et des méthodes « set » qui fournissent un accès en écriture à une propriété donnée. Imaginons par exemple une classe contenant une propriété nommée `nomDutilisateur` :

```
var nomDutilisateur:String;
```

Au lieu de permettre aux occurrences de la classe d'accéder directement à cette propriété (`obj.nomDutilisateur = "Julie"`, par exemple), la classe peut utiliser deux méthodes, `getNomDutilisateur` et `setNomDutilisateur`, qui seront implémentées de la façon suivante :

```
function getNomDutilisateur:String() {
  return nomDutilisateur;
}

function setNomDutilisateur(name:String): {
  nomDutilisateur = name;
}
```

Comme vous pouvez le constater, `getNomDutilisateur` renvoie la valeur courante de `nomDutilisateur` et `setNomDutilisateur` définit le paramètre de chaîne transmis à la méthode en tant que valeur de `nomDutilisateur`. Une occurrence de la classe utiliserait alors la syntaxe suivante pour obtenir ou définir la propriété `nomDutilisateur`.

```
// appel de la méthode "get"
var nom = obj.getNomDutilisateur();
// appel de la méthode "set"
obj.setNomDutilisateur("Julie");
```

Cependant, si vous souhaitez utiliser une syntaxe plus concise, utilisez les méthodes get/set *implicites*. Les méthodes get/set implicites vous permettent d'accéder directement aux propriétés de classe, tout en conservant de bonnes pratiques de programmation orientée objet.

Pour définir ces méthodes, utilisez les attributs de méthodes `get` et `set`. Créez des méthodes qui obtiennent ou définissent la valeur d'une propriété et ajoutez le mot-clé `get` ou `set` avant le nom de méthode.

```
function get user():String {
  return nomDutilisateur;
}

function set user(name:String):Void {
  nomDutilisateur = name;
}
```

Une méthode get ne doit prendre aucun paramètre. Une méthode set doit prendre exactement un paramètre requis. Une méthode set peut avoir le même nom qu'une méthode get dans le même domaine. Les méthodes get/set n'ont pas le même nom que les autres propriétés. Par exemple, dans le code ci-dessus qui définit des méthodes get/set nommées `utilisateur`, vous ne pourriez pas avoir de propriété nommée `utilisateur` dans la même classe.

Contrairement aux méthodes ordinaires, les méthodes get/set sont appelées sans parenthèses ou instructions. Par exemple, la syntaxe suivante peut désormais être utilisée pour accéder à la valeur de `nomDutilisateur` ou la modifier à l'aide des méthodes get/set définies plus haut.

```
var name = obj.user;
obj.user = "Jean";
```

Remarque : Les méthodes get/set implicites sont des abréviations syntaxiques de la méthode `Object.addProperty()` dans ActionScript 1.

Création de classes dynamiques

Par défaut, les propriétés et méthodes d'une classe sont fixes. C'est-à-dire que l'occurrence d'une classe ne peut créer ou accéder à des propriétés ou méthodes qui n'étaient pas déclarées ou définies à l'origine par la classe. Considérons par exemple une classe Personne qui définit deux propriétés, `nom` et `âge` :

```
class Personne {
  var nom:String;
  var âge:Number;
}
```

Si, dans un autre script, vous créez une occurrence de la classe Personne et essayez d'accéder à une propriété de la classe qui n'existe pas, le compilateur génère une erreur. Par exemple, le code suivant crée une nouvelle occurrence de la classe Personne (`une_personne`) et essaie d'affecter une valeur à une propriété nommée `couleurCheveux`, qui n'existe pas.

```
var une_personne:Personne = new Personne();
une_personne.couleurCheveux = "bleu"; // erreur de compilation
```

Ce code crée une erreur de compilation car la classe Personne ne déclare pas de propriété nommée `couleurCheveux`. Dans la plupart des cas, c'est exactement ce que vous souhaitez qu'il se passe.

Dans certains cas, cependant, il peut être utile d'ajouter et d'accéder à des propriétés ou méthodes d'une classe à l'exécution qui ne sont pas définies dans la définition de classe originale. C'est ce que vous permet de faire le modificateur de classe `dynamic`. Par exemple, le code suivant ajoute le modificateur `dynamic` à la classe Personne présentée précédemment :

```
dynamic class Personne {
  var nom:String;
  var âge:Number;
}
```

Les occurrences de la classe Personne peuvent désormais ajouter et accéder aux propriétés et méthodes qui ne sont pas définies dans la classe originale.

```
var une_personne:Personne = new Personne();
une_personne.couleurCheveux = "bleu"; // aucune erreur de compilation car
  cette classe est dynamique
```

Les sous-classes des classes dynamiques sont également des classes dynamiques.

Compilation et exportation des classes

Par défaut, les classes utilisées par un fichier SWF sont mises en paquets et exportées vers la première image du fichier SWF. Vous pouvez également spécifier l'image où vos classes sont mises en paquets et exportées. Cette option est très pratique, par exemple lorsqu'un fichier SWF utilise de nombreuses classes longues à télécharger. Si les classes sont exportées vers la première image, l'utilisateur doit attendre que tout le code de classe soit téléchargé avant de voir apparaître cette image. En spécifiant une image ultérieure dans le scénario, vous pouvez afficher une courte animation de chargement dans les premières images du scénario, pendant le téléchargement du code de classe dans l'image ultérieure.

Pour spécifier l'image à exporter pour les classes d'un document Flash :

1. Lorsqu'un fichier FLA est ouvert, choisissez Fichier > Paramètres de publication.
2. Dans la boîte de dialogue Paramètres de publication, cliquez sur l'onglet Flash.
3. Cliquez sur le bouton Paramètres qui se trouve en regard du menu contextuel de la version ActionScript pour ouvrir la boîte de dialogue Paramètres ActionScript.
4. Dans le champ de texte Exporter l'image pour les classes, saisissez le numéro de l'image où vous voulez exporter votre code de classe.

   Si l'image spécifiée n'existe pas dans le scénario, un message d'erreur apparaît lors de la publication du fichier SWF.
5. Cliquez sur OK pour fermer la boîte de dialogue Paramètres ActionScript, puis cliquez de nouveau sur OK pour fermer la boîte de dialogue Paramètres de publication.

PARTIE IV
Utilisation des données et des médias externes

Cette section présente l'intégration de données et de médias externes dans vos applications Macromedia Flash.

CHAPITRE 10
Utilisation de données externes

Dans Macromedia Flash MX 2004 et Macromedia Flash MX Professionnel 2004, vous pouvez utiliser ActionScript pour charger des données provenant de sources externes dans un fichier SWF. Vous pouvez également envoyer des données de fichier SWF afin qu'elles soient traitées par un serveur d'application (comme Macromedia ColdFusion MX ou Macromedia JRun) ou par un autre type de script côté serveur comme PHP ou Perl. Flash Player peut envoyer et charger des données sur HTTP, HTTPS ou à partir d'un fichier texte local. Vous pouvez également créer des connexions socket TCP/IP persistantes pour des applications qui requièrent un court délai (ex. : applications de dialogue en ligne ou services de devis).

Il est possible de formater les données que vous chargez ou que vous envoyez à partir d'un fichier SWF sous forme d'un fichier XML (Extensible Markup Language) ou de paires nom-valeur.

Flash Player peut également échanger des données avec son environnement hôte (un navigateur web, par exemple) ou avec une autre occurrence de Flash Player sur le même ordinateur.

Par défaut, un fichier SWF peut seulement accéder à des données se trouvant sur le même domaine que le domaine d'origine du clip Flash (par exemple, www.macromedia.com). Pour plus d'informations, consultez *Fonctions de sécurité de Flash Player*, page 199.

Echange de variables avec une source distante

Un fichier SWF est une fenêtre permettant de saisir et d'afficher des informations, un peu comme une page HTML. Cependant, les fichiers SWF peuvent rester chargés dans le navigateur et être mis à jour en permanence à l'aide de nouvelles informations sans qu'il soit nécessaire d'actualiser toute la page. Grâce aux fonctions et méthodes d'ActionScript, vous pouvez envoyer et recevoir des informations à partir de scripts côté serveur, de fichiers texte et de fichiers XML.

En outre, les scripts côté serveur peuvent demander des informations précises à une base de données et les transmettre à un fichier SWF. Les scripts côté serveur peuvent être rédigés en plusieurs langages, les plus communs étant CFML, Perl, ASP (Microsoft Active Server Pages) et PHP. Le stockage et l'extraction des informations par le biais d'une base de données vous permet de créer un contenu dynamique et personnalisé pour votre fichier SWF. Par exemple, vous pourriez créer un tableau de messages, des profils personnels pour les utilisateurs ou un panier suivant ce qu'un utilisateur a acheté afin de déterminer ses préférences.

Plusieurs fonctions et méthodes d'ActionScript vous permettent d'échanger des informations avec un fichier SWF. Chaque fonction ou chaque méthode utilise un protocole pour transférer les informations, dont le format doit être spécifique.

- Les fonctions et les méthodes MovieClip utilisant le protocole HTTP ou HTTPS pour envoyer des informations au format de code URL sont `getURL()`, `loadVariables()`, `loadVariablesNum()`, `loadMovie()` et `loadMovieNum()`.
- Les méthodes LoadVars utilisant le protocole HTTP ou HTTPS pour échanger des informations au format de code URL sont `load()`, `send()` et `sendAndLoad()`.
- Les méthodes utilisant le protocole HTTP ou HTTPS pour échanger des informations sous forme de fichiers XML sont `XML.send()`, `XML.load()` et `XML.sendAndLoad()`.
- Les méthodes qui créent et utilisent une connexion socket TCP/IP pour échanger des informations sous forme de fichiers XML sont `XMLSocket.connect()` et `XMLSocket.send()`.

Vérification des données chargées

Chaque fonction ou méthode qui charge des données vers un fichier SWF (sauf `XMLSocket.send()`) est *asynchrone* : les résultats de l'action sont renvoyés à un moment indéterminé.

Avant de pouvoir utiliser les données chargées dans un fichier SWF, vous devez d'abord vérifier si elles ont bien été chargées. Par exemple, vous ne pouvez pas charger des variables et en manipuler les valeurs dans le même script. Dans le script suivant, vous ne pouvez pas utiliser la variable `dernièreImageConsultée` tant que vous n'êtes pas certain que la variable a été chargée depuis le fichier `mesDonnées.txt` :

```
loadVariables("mesDonnées.txt", 0);
gotoAndPlay(dernièreImageConsultée);
```

Chaque fonction ou méthode possède une technique spécifique que vous pouvez utiliser pour vérifier les données qui ont été chargées. Si vous utilisez `loadVariables()` ou `loadMovie()`, vous pouvez charger des informations dans une cible de clip et utiliser l'événement `data` du gestionnaire `onClipEvent()` pour exécuter un script. Si vous utilisez `loadVariables()` pour charger les données, le gestionnaire `onClipEvent(data)` est exécuté une fois la dernière variable chargée. Si vous utilisez `loadMovie()` pour charger les données, le gestionnaire `onClipEvent(data)` est exécuté chaque fois qu'une partie du fichier SWF est transmise à Flash Player.

Par exemple, l'action de bouton suivante charge les variables depuis le fichier `mesDonnées.txt` dans le clip `cibleChargeMC` :

```
on(release) {
  loadVariables("mesDonnées.txt", _root.cibleChargeMC);
}
```

Un gestionnaire `onClipEvent()` affecté à l'occurrence `cibleChargeMC` utilise la variable `dernièreImageConsultée`, chargée depuis le fichier `mesDonnées.txt`. L'action suivante n'est exécutée qu'une fois que toutes les variables, y compris `dernièreImageConsultée`, sont chargées :

```
onClipEvent(data) {
  goToAndPlay(dernièreImageConsultée);
}
```

Si vous utilisez les méthodes `XML.load()`, `XML.sendAndLoad()` et `XMLSocket.connect()`, vous devez définir un gestionnaire qui traitera les données dès leur arrivée. Ce gestionnaire est une propriété d'un objet XML ou XMLSocket auquel vous affectez une fonction que vous avez définie. Les gestionnaires sont automatiquement appelés lorsque les informations sont reçues. Pour l'objet XML, utilisez `XML.onLoad()` ou `XML.onData()`. Pour l'objet XMLSocket, utilisez `XMLSocket.onConnect()`.

Pour plus d'informations, consultez *Utilisation de la classe XML*, page 192 et *Utilisation de la classe XMLSocket*, page 195.

Utilisation du protocole HTTP pour les connexions aux scripts côté serveur

Les fonctions `loadVariables()`, `loadVariablesNum()`, `getURL()`, `loadMovie()` et `loadMovieNum()`, ainsi que les méthodes `MovieClip.loadVariables()`, `MovieClip.loadMovie()` et `MovieClip.getURL()` peuvent toutes communiquer avec des scripts côté serveur sur des protocoles HTTP ou HTTPS. Ces fonctions envoient toutes les variables du scénario auquel la fonction est associée. Lorsqu'elles sont utilisées comme méthodes de l'objet MovieClip, les fonctions `loadVariables()`, `getURL()` et `loadMovie()` envoient toutes les variables du clip indiqué ; chaque fonction (ou méthode) traite sa réponse de la manière suivante :

- `getURL()` renvoie les informations dans une fenêtre de navigateur et non dans Flash Player.
- `loadVariables()` charge les variables dans un scénario ou niveau spécifié de Flash Player.
- `loadMovie()` charge un fichier SWF dans un clip ou niveau spécifié dans Flash Player.

Lorsque vous utilisez `loadVariables()`, `getURL()` ou `loadMovie()`, vous pouvez spécifier plusieurs paramètres :

- *`URL`* est le fichier dans lequel se trouvent les variables distantes.
- *`Emplacement`* est le niveau ou la cible dans le fichier SWF qui reçoit les variables. (La fonction `getURL` ne prend pas ce paramètre.)

  Pour plus d'informations sur les niveaux et les cibles, consultez A propos des scénarios et des niveaux dans le guide Utilisation de Flash de l'aide.
- *`Variables`* définit la méthode HTTP, `GET` ou `POST`, avec laquelle les variables seront envoyées. Lorsqu'elles sont omises, Flash Player utilise par défaut la méthode `GET`, mais aucune variable n'est envoyée.

Par exemple, pour suivre les meilleurs scores d'un jeu, vous pouvez stocker les scores sur un serveur et utiliser une fonction `loadVariables()` pour les charger dans le fichier SWF chaque fois que quelqu'un joue à ce jeu. L'appel de fonction pourrait avoir l'aspect suivant :

```
loadVariables("http://www.monSite.com/scripts/meilleur_score.php",
  _root.clipDeScore, GET);
```

Cet exemple charge les variables du script PHP `meilleur_score.php` dans l'occurrence de clip `clipDeScore` en utilisant la méthode HTTP `GET`.

Les variables chargées à l'aide de la fonction `loadVariables` doivent être au format MIME standard *application/x-www-form-urlencoded* (un format standard utilisé par les scripts CGI). Le fichier que vous spécifiez dans le paramètre *URL* de l'action `loadVariables()` doit écrire les paires de variables et valeurs dans ce format pour que Flash puisse les lire. Ce fichier peut spécifier n'importe quel nombre de variables, les paires variable et valeur devant être séparées par une esperluette (&) et les mots à l'intérieur d'une valeur devant être séparés par le signe plus (+). Par exemple, cette séquence définit plusieurs variables :

```
meilleurScore1=54000&nomDuJoueur1=rockin+good&meilleurScore2=53455&nomDuJoueur
  2=bonehelmet&meilleurScore3=42885&nomDuJoueur3=soda+pop
```

Pour plus d'informations, consultez `loadVariables()`, page 471, `getURL()`, page 434, `loadMovie()`, page 468 et l'entrée *Classe LoadVars* dans le Chapitre 12, *Dictionnaire ActionScript*, page 215.

Utilisation de la classe LoadVars

Vous pouvez utiliser la classe LoadVars au lieu de la classe `loadVariables()` pour transférer des variables entre un fichier SWF et un serveur. La classe LoadVars permet d'envoyer toutes les variables d'un objet à une adresse URL déterminée et de charger toutes les variables d'une adresse URL déterminée dans un objet. La réponse du serveur déclenche la méthode `LoadVars.onLoad()` et définit les variables de la cible. Vous pouvez utiliser l'objet LoadVars pour obtenir des informations sur les erreurs et des indicateurs d'avancement, ainsi que pour diffuser les données pendant leur téléchargement.

La classe LoadVars est similaire à la classe XML et utilise les méthodes `load()`, `send()` et `sendAndLoad()` pour établir la communication avec le serveur. La principale différence entre les classes LoadVars et XML réside dans le fait que les données LoadVars sont une propriété de l'objet LoadVars, et non une arborescence DOM (Document Object Model) XML stockée dans l'objet XML.

Vous devez créer un objet LoadVars pour appeler ses méthodes. Cet objet est un conteneur qui stocke les données chargées.

La procédure suivante montre comment utiliser un objet LoadVars pour charger des variables à partir d'un fichier texte et les afficher dans un champ de texte.

Pour charger des données avec l'objet LoadVars :

1. Dans un éditeur de texte comme Notepad ou SimpleText, créez un fichier texte et ajoutez-lui le texte suivant :

   ```
   jour=11&mois=juillet&année=2003
   ```

2. Enregistrez le fichier sous date.txt.
3. Dans Flash, créez un document.
4. Créez un champ de texte dynamique sur la scène et donnez-lui le nom d'occurrence `date_txt`.
5. Sélectionnez l'image 1 dans le scénario et ouvrez le panneau Actions (Fenêtre > Panneaux de développement > Actions) si ce dernier n'est pas déjà visible.

6 Entrez le code suivant dans le panneau Actions :

```
var dateVars = new LoadVars();
dateVars.onLoad = function(ok) {
  if (ok) {
    date_txt.text = dateVars.day+"/"+dateVars.month+"/"+dateVars.year;
  }
};
dateVars.load("date.txt");
```

Ce code charge les variables dans le fichier data.txt (`jour`, `mois`, `année`), puis il les formate et les affiche dans le champ de texte `date_txt`.

7 Enregistrez le document sous le nom dateReader.fla dans le répertoire contenant le fichier date.txt (le fichier texte que vous avez enregistré à l'étape 3).

8 Choisissez Contrôle > Tester l'animation pour tester le document.

Pour plus d'informations, consultez l'entrée *Classe LoadVars* dans le Chapitre 12, *Dictionnaire ActionScript*, page 215.

A propos du langage XML

Le langage *XML* (*Extensible Markup Language*) est en passe de devenir le standard d'échange de données structurées dans les applications Internet. Vous pouvez intégrer les données de Flash avec des serveurs qui utilisent la technologie XML pour construire des applications sophistiquées telles que des services de dialogue en ligne ou de courtage.

En XML, tout comme en HTML, vous utilisez des balises pour *marquer*, ou définir, une chaîne de texte. Dans le langage HTML, vous utilisez des balises prédéfinies pour indiquer la façon dont le texte doit apparaître dans un navigateur web (par exemple, la balise `<b>` indique que le texte doit être en gras). Dans le langage XML, vous définissez des balises qui identifient le type d'une partie de données (par exemple, `<motDePasse>monSecret</motDePasse>`). Le langage XML sépare la structure des informations de leur mode d'affichage, ce qui permet de réutiliser un même document XML dans des environnements différents.

Chaque balise XML est appelée *nœud* ou *élément*. Chaque nœud possède un type (1, qui indique un élément XML ou 3, qui indique un nœud texte) et les éléments peuvent également posséder des attributs. Un nœud imbriqué dans un autre est appelé *nœud enfant*. Cette structure hiérarchique de nœuds est appelée DOM (Document Object Model) XML, un peu comme le DOM JavaScript, qui correspond à la structure des éléments dans un navigateur web.

Dans l'exemple suivant, `<PORTEFEUILLE>` est le nœud parent ; il ne possède pas d'attributs et contient le nœud enfant `<EFFET>` qui possède les attributs `SYMBOLE`, `QTÉ`, `PRIX` et `VALEUR` :

```
<PORTEFEUILLE>
  <EFFET SYMBOLE ="RICHE"
    QTÉ="75"
    PRIX="245.50"
    VALEUR="18412.50" />
</PORTEFEUILLE>
```

Utilisation de la classe XML

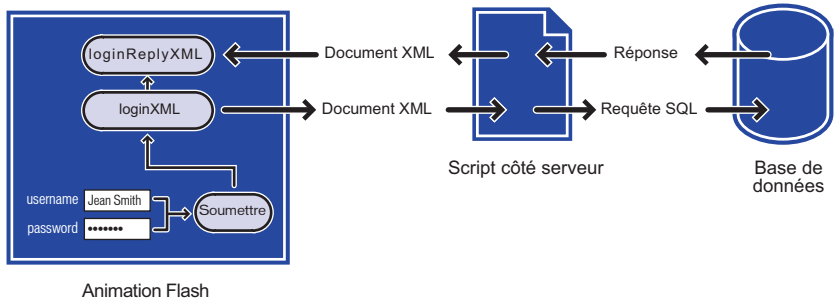
Les méthodes de la classe XML ActionScript (par exemple, `appendChild()`, `removeNode()` et `insertBefore()`) permettent de structurer les données XML dans Flash qui doivent être envoyées à un serveur et de manipuler et interpréter les données XML téléchargées.

Les méthodes de la classe XML suivantes permettent d'échanger des données XML avec un serveur à l'aide de la méthode HTTP `POST` :

- La méthode `load()` télécharge le code XML depuis une URL et le place dans un objet XML ActionScript.
- La méthode `send()` transmet un objet XML à une URL. Toutes les informations renvoyées sont affichées dans une fenêtre de navigateur.
- La méthode `sendAndLoad()` envoie un objet XML à une URL. Toutes les informations renvoyées sont placées dans un objet XML ActionScript.

Par exemple, vous pourriez créer un système de courtage qui stockerait toutes ses informations (noms d'utilisateur, mots de passe, identifiants de session, contenu des portefeuilles et informations de transaction) dans une base de données.

Le script côté serveur qui transmet les informations entre Flash et la base de données lit et écrit les données au format XML. Vous pouvez utiliser ActionScript pour convertir les informations récupérées dans le fichier SWF (par exemple, un nom d'utilisateur et un mot de passe) en un objet XML et envoyer ensuite les données au script côté serveur sous forme de document XML. Vous pouvez également utiliser ActionScript pour charger le document XML que le serveur renvoie dans un objet XML à utiliser dans le fichier SWF.



Flux et conversion des données entre une animation Flash, un script côté serveur et une base de données

La validation du mot de passe pour le système de courtage nécessite deux scripts : une fonction définie sur l'image 1 et un script qui crée et envoie les objets XML associés au bouton Envoyer du formulaire.

Lorsqu'un utilisateur entre des informations dans les champs de texte du fichier SWF avec les variables `nomDutilisateur` et `motDePasse`, les variables doivent être converties au format XML avant d'être transmises au serveur. La première section du script charge les variables dans un objet XML nouvellement créé et appelé `XMLDouvertureDeSession`. Lorsqu'un utilisateur clique sur le bouton Envoyer, l'objet `XMLDouvertureDeSession` est converti en une chaîne XML et envoyé au serveur.

Le script suivant est associé au bouton Envoyer. Pour comprendre le script, vous pourrez vous aider des commentaires (indiqués par les caractères `//`) :

```
on (release) {
  // A. Construire un document XML avec un élément d'ouverture de session
  XMLDouvertureDeSession = new XML();
  élémentDouvertureDeSession = XMLDouvertureDeSession.createElement("LOGIN");
  élémentDouvertureDeSession.attributes.nomDutilisateur = nomDutilisateur;
  élémentDouvertureDeSession.attributes.motDePasse = motDePasse;
  XMLDouvertureDeSession.appendChild(élémentDouvertureDeSession);

  // B. Construire un objet XML contenant la réponse du serveur
  XMLréponseDouv = new XML();
  XMLréponseDouv.onLoad = pourRéponseDouv;

  // C. Envoyer l'élément LOGIN au serveur,
  //    placer la réponse dans XMLréponseDouv
  XMLDouvertureDeSession.sendAndLoad("https://www.imexstocks.com/main.cgi",
              XMLréponseDouv);
}
```

La première section du script génère le code XML suivant lorsque l'utilisateur clique sur le bouton Envoyer:

```
<LOGIN NOMDUTILISATEUR="JeanSmith" MOTDEPASSE="monSecret" />
```

Le serveur reçoit le code XML, génère une réponse XML et la renvoie au fichier SWF. Si le mot de passe est accepté, le serveur envoie la réponse suivante :

```
<REPONSELOGIN ETAT="OK" SESSION="rnr6f7vkj2oe14m7jkkycilb" />
```

Ce code XML comprend un attribut `SESSION` qui contient une ID de session aléatoire unique et qui sera utilisée dans toutes les communications entre le client et le serveur pour le reste de la session. Si le mot de passe est rejeté, le serveur répond par le message suivant :

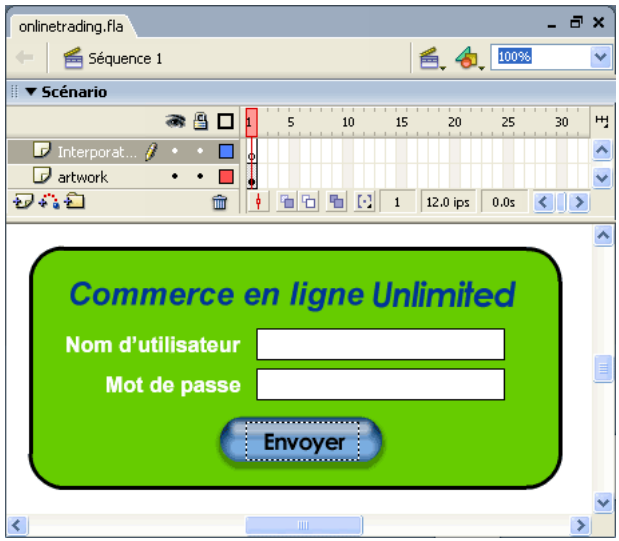
```
<REPONSELOGIN ETAT="ECHEC" />
```

Le nœud XML `REPONSELOGIN` doit être chargé dans un objet XML vide du fichier SWF. L'instruction suivante crée l'objet XML `XMLréponseDouv` pour recevoir le nœud XML :

```
// B. Construire un objet XML contenant la réponse du serveur
XMLréponseDouv = new XML();
XMLréponseDouv.onLoad = pourRéponseDouv;
```

La seconde instruction affecte la fonction `pourRéponseDouv()` au gestionnaire `XMLréponseDouv.onLoad`.

L'élément XML `REPONSELOGIN` arrive de manière asynchrone, un peu comme les données d'une fonction `loadVariables()` et est chargé dans l'objet `XMLréponseDouv`. Lorsque les données arrivent, le gestionnaire `onLoad` de l'objet `XMLréponseDouv` est appelé. Vous devez définir la fonction `pourRéponseDouv()` et l'affecter au gestionnaire `XMLréponseDouv.onLoad` pour qu'il puisse traiter l'élément `REPONSELOGIN`. Vous devez également affecter la fonction `pourRéponseDouv()` à l'image contenant le bouton Envoyer.



La fonction `pourRéponseDouv()` est définie dans la première image du fichier SWF. Pour comprendre le script, vous pourrez vous aider des commentaires.

```
function pourRéponseDouv() {
  // Récupérer le premier élément XML
  var e = this.premierEnfant;
  // Si le premier élément XML est un élément REPONSELOGIN avec
  // OK pour état, ouvrir l'écran de portefeuille. Sinon,
  // ouvrir l'écran d'échec et laisser l'utilisateur réessayer.
  if (e.nomDeNoeud == "REPONSELOGIN" && e.attributes.état == "OK") {
// Enregistrer l'identifiant de session pour les futures communications avec le
  serveur
  idSession = e.attributes.session;
// Ouvrir l'écran de portefeuille
    gotoAndStop("écranPortefeuille");
} else {
    // L'ouverture de session a échoué !  Ouvrir l'écran d'échec.
    gotoAndStop("échecOuvSession");
  }
}
```

La première ligne de cette fonction, `var e = this.premierEnfant`, utilise le mot-clé `this` pour faire référence à l'objet XML `XMLréponseDouv` qui vient d'être chargé avec XML depuis le serveur. Vous pouvez utiliser `this` car `pourRéponseDouv()` a été invoquée sous la forme `XMLréponseDouv.onLoad`, et donc, même si `pourRéponseDouv()` se révèle être une fonction normale, elle se comporte en fait comme une méthode de `XMLréponseDouv`.

Pour envoyer le nom d'utilisateur et le mot de passe au format XML au serveur et pour charger une réponse XML dans le fichier SWF, vous pouvez utiliser la méthode `sendAndLoad()`, comme dans l'exemple suivant :

```
// C. Envoyer l'élément LOGIN au serveur,
//    placer la réponse dans XMLréponseDouv
  XMLDouvertureDeSession.sendAndLoad("https://www.imexstocks.com/main.cgi",
  XMLréponseDouv);
```

Remarque : Cette démonstration n'est qu'un exemple et Macromedia ne garantit pas le niveau de sécurité fourni. Si vous implémentez un système sécurisé protégé par mot de passe, assurez-vous de bien comprendre la sécurité réseau.

Pour plus d'informations, consultez Intégration de XML et de Flash dans une application web sur le site www.macromedia.com/support/flash/interactivity/xml/ et l'entrée *Classe XML* dans le Chapitre 12, *Dictionnaire ActionScript*, page 215.

Utilisation de la classe XMLSocket

ActionScript fournit une classe XMLSocket intégrée qui vous permet d'établir une connexion continue avec un serveur. Une connexion socket permet au serveur de publier l'information au client dès qu'elle est disponible. Sans connexion continue, le serveur devra attendre une requête HTTP. Cette connexion ouverte supprime les périodes d'attente et est souvent utilisée dans des applications en temps réel telles que les dialogues en ligne. Les données sont envoyées sur la connexion socket sous forme d'une chaîne et doivent être au format XML. Vous pouvez utiliser la classe XML pour structurer les données.

Pour créer une connexion socket, vous devez créer une application côté serveur qui attendra la requête de connexion socket et enverra une réponse au fichier SWF. Ce type d'applications côté serveur peut être écrit dans un langage tel que Java.

Vous pouvez utiliser les méthodes `connect()` et `send()` de la classe XMLSocket pour transférer un objet XML vers et à partir d'un serveur sur une connexion socket. La méthode `connect()` établit une connexion socket avec le port d'un serveur web. La méthode `send()` transmet un objet XML au serveur spécifié dans la connexion socket.

Lorsque vous invoquez la méthode `connect()`, Flash Player ouvre une connexion TCP/IP avec le serveur et garde cette connexion ouverte jusqu'à ce qu'un des événements suivants se produise :

- La méthode `close()` de la classe XMLSocket est appelée.
- Il n'existe plus aucune référence à l'objet XMLSocket.
- Flash Player se ferme.
- La connexion est rompue (le modem est déconnecté, par exemple).

L'exemple suivant crée une connexion socket XML et envoie les données de l'objet XML `monCodeXML`. Pour comprendre le script, vous pourrez vous aider des commentaires (indiqués par les caractères `//`) :

```
// Créer un objet XMLSocket
sock = new XMLSocket();
// Appeler sa méthode connect() pour établir une connexion au port 1024
// du serveur à l'URL
sock.connect("http://www.monServeur.fr", 1024);
// Définir une fonction à affecter à l'objet socket gérant
// la réponse du serveur. Si la connexion réussit, envoyer
// l'objet monCodeXML. Sinon, afficher un message d'erreur dans un
// champ de texte.
```

```
function onSockConnect(succès){
  if (succès){
    sock.send(monCodeXML);
} else {
    msg="Une erreur de connexion s'est produite avec "+serverName;
  }
}
// Affecter la fonction onSockConnect() à la propriété onConnect
sock.onConnect = onSockConnect;
```

Pour plus d'informations, consultez l'entrée *Classe XMLSocket* dans le Chapitre 12, *Dictionnaire ActionScript*, page 215.

Echange de messages avec Flash Player

Pour envoyer des messages depuis un fichier SWF vers son environnement hôte (par exemple, un navigateur web, une animation Macromedia Director ou Flash Player autonome), vous pouvez utiliser la fonction `fscommand()`. Cette fonction vous permet d'étendre votre fichier SWF à l'aide des capacités de l'hôte. Par exemple, vous pouvez transmettre une fonction `fscommand()` à une fonction JavaScript dans une page HTML qui ouvre une nouvelle fenêtre de navigateur avec des propriétés spécifiques.

Pour contrôler un fichier SWF dans Flash Player avec des langages tels que JavaScript, VBScript et Microsoft JScript, vous pouvez utiliser les méthodes Flash Player (fonctions qui envoient des messages depuis un environnement hôte vers le fichier SWF). Par exemple, vous pouvez disposer d'un lien dans une page HTML qui envoie votre fichier SWF vers un cadre spécifique.

Utilisation de fscommand()

Utilisez la fonction `fscommand()` pour envoyer un message au programme hébergeant Flash Player. La fonction `fscommand()` dispose de deux paramètres : *`commande`* et *`arguments`*. Pour envoyer un message à la version autonome de Flash Player, vous devez utiliser des commandes et des arguments prédéfinis. Par exemple, l'action suivante définit le lecteur autonome pour qu'il affiche le fichier SWF en taille plein écran lorsque le bouton est relâché :

```
on(release) {
  fscommand("fullscreen", "true");
}
```

Le tableau suivant indique les valeurs que vous pouvez spécifier pour les paramètres *`commande`* et *`arguments`* de la fonction `fscommand()` pour contrôler un fichier SWF lu dans le lecteur autonome (y compris les projections) :

Commande	Arguments	Objectif
`quit`	Aucun	Ferme la projection.
`fullscreen`	`true` ou `false`	La spécification de `true` définit Flash Player en mode plein écran. La spécification de `false` renvoie le lecteur en affichage normal du menu.
`allowscale`	`true` ou `false`	La spécification de `false` définit le lecteur de sorte que le fichier SWF soit toujours affiché dans sa taille originale et que son échelle ne soit jamais modifiée. La spécification de `true` oblige le fichier SWF à adopter l'échelle 100 % du lecteur.

Commande	Arguments	Objectif
showmenu	true ou false	La spécification de true active le jeu complet des éléments de menu contextuel. La spécification de false masque tous les éléments de menu contextuel, à l'exception de Paramètres et A propos de Flash Player.
exec	Chemin de l'application	Exécute une application depuis la projection.

Pour utiliser `fscommand()` pour envoyer un message à un langage de programmation tel que JavaScript dans un navigateur web, vous pouvez transmettre deux arguments quelconques dans les paramètres *commande* et *arguments*. Ces paramètres peuvent être des chaînes ou des expressions et seront utilisés dans une fonction JavaScript qui traite la fonction `fscommand()`.

Une fonction `fscommand()` invoque la fonction JavaScript *nomDeLanimation*_DoFSCommand dans la page HTML qui contient le fichier SWF, où *nomDeLanimation* est le nom de Flash Player tel qu'il est affecté par l'attribut `NAME` de la balise `EMBED` ou par l'attribut `ID` de la balise `OBJECT`. Si le nom `monAnimation` a été affecté à Flash Player, la fonction JavaScript invoquée est `monAnimation_DoFSCommand`.

Pour utiliser `fscommand()` afin d'ouvrir une boîte de message à partir d'un fichier SWF dans la page HTML à l'aide de JavaScript :

1 Dans la page HTML qui contient le fichier SWF, ajoutez le code JavaScript suivant :

```
function Lanimation_DoFSCommand(commande, args) {
  if (commande == "fenêtreMessage") {
    alert(args);
  }
}
```

Si vous publiez votre fichier SWF en utilisant Flash avec le modèle FSCommand disponible dans les paramètres de publication HTML, ce code s'insère automatiquement. Les attributs `NAME` et `ID` du fichier SWF constitueront le nom du fichier. Par exemple, pour le fichier monAnimation.fla, les attributs seront définis avec `monAnimation`. (Pour plus d'informations sur la publication, consultez Publication dans le guide Utilisation de Flash de l'aide.)

Alternativement, pour les applications Microsoft Internet Explorer, vous pouvez associer un gestionnaire d'événement directement dans la balise `<SCRIPT>`, comme l'illustre cet exemple :

```
<Script Language = "JavaScript" event="FSCommand (commande, args)" for=
  "Lanimation">
...
</Script>
```

2 Dans le document Flash, ajoutez la fonction `fscommand()` à un bouton, comme dans l'exemple suivant :

```
on (press) {
  fscommand("fenêtreMessage", "Ceci est une fenêtre de message invoquée
  depuis Flash.");
}
```

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

```
fscommand("boîteDeMessage", "Bonjour, " + nom + ", bienvenue sur notre site
  web!")
```

3 Choisissez Fichier > Aperçu avant publication > HTML pour tester le document.

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

En Visual Basic, Visual C++, et dans d'autres programmes pouvant héberger les contrôles ActiveX, `fscommand` envoie un événement VB avec deux chaînes qui peut être traité dans l'environnement du langage de programmation. Pour plus d'informations, utilisez les mots-clés *méthode Flash* pour effectuer une recherche sur le centre de support de Flash à l'adresse www.macromedia.com/go/flash_support_fr.

A propos des méthodes Flash Player

Vous pouvez utiliser les méthodes Flash Player pour contrôler un fichier SWF dans Flash Player avec des langages tels que JavaScript et VBScript. Comme avec les autres méthodes, vous pouvez utiliser les méthodes Flash Player pour envoyer des appels à des fichiers SWF depuis un environnement de programmation autre qu'ActionScript. Chaque méthode possède un nom et la plupart prennent des paramètres. Un paramètre spécifie une valeur sur laquelle opère la méthode. Le calcul effectué par certaines méthodes renvoie une valeur qui peut être utilisée par l'environnement de programmation.

Deux technologies différentes permettent la communication entre le navigateur et Flash Player : LiveConnect (Netscape Navigator 3.0 et versions ultérieures sous Windows 95/98/2000/NT ou Power Macintosh) et ActiveX (Internet Explorer 3.0 et versions ultérieures sous Windows 95/98/2000/NT). Bien que les techniques de programmation soient équivalentes pour tous les navigateurs et les langages, des propriétés et événements supplémentaires sont disponibles pour l'utilisation des contrôles ActiveX.

Pour obtenir plus d'informations, ainsi que la liste complète des méthodes de programmation de Flash Player, utilisez les mots-clés *méthode Flash* pour rechercher le centre de support de Flash à l'adresse www.macromedia.com/go/flash_support_fr.

A propos de l'utilisation des méthodes Flash JavaScript avec Flash Player

Flash Player 6 version 40 et ultérieures prennent en charge les méthodes Flash Java Script et FSCommand dans Netscape 6.2 et versions ultérieures. Les versions antérieures ne prennent pas en charge les méthodes Flash JavaScript et FSCommand dans Netscape 6.2 ou versions ultérieures.

Pour Netscape 6.2 et versions ultérieures, vous n'avez pas besoin de définir l'objet `swLiveConnect` sur la valeur `true`. Cependant, la définition de `swLiveConnect` sur la valeur `true` n'a aucune incidence.

Fonctions de sécurité de Flash Player

Par défaut, Flash Player 7 et ses versions ultérieures empêchent un fichier SWF servi par un domaine d'accéder aux données, objets ou variables de fichiers SWF servis par des domaines différents. En outre, le contenu chargé à l'aide de protocoles non sécurisés (non-HTTPS) ne peut pas accéder au contenu chargé à l'aide d'un protocole sécurisé (HTTPS), même s'ils sont tous les deux situés exactement dans le même domaine. Par exemple, un fichier SWF situé à l'adresse http://www.macromedia.com/main.swf ne peut pas charger de données à partir de l'adresse https://www.macromedia.com/data.txt sans autorisation explicite. De même, un fichier SWF servi par un domaine ne peut pas charger les données (à l'aide de `loadVariables()`, par exemple) d'un autre domaine.

Les adresses IP numériques identiques sont compatibles. Cependant, un nom de domaine n'est pas compatible avec une adresse IP, même si le nom de domaine renvoie à la même adresse IP.

Le tableau suivant présente des exemples de domaines compatibles :

www.macromedia.com	www.macromedia.com
data.macromedia.com	data.macromedia.com
65.57.83.12	65.57.83.12

Le tableau suivant présente des exemples de domaines incompatibles :

www.macromedia.com	data.macromedia.com
macromedia.com	www.macromedia.com
www.macromedia.com	macromedia.com
65.57.83.12	www.macromedia.com (même si ce domaine correspond à l'adresse 65.57.83.12)
www.macromedia.com	65.57.83.12 (même si www.macromedia.com correspond à cette adresse IP)

Pour plus d'informations sur la façon de permettre à un fichier SWF servi par un domaine d'accéder aux données, objets ou variables de fichiers SWF qui sont servis par un autre domaine, consultez *A propos de l'autorisation d'accès aux données entre des fichiers SWF inter-domaines*, page 200. Pour plus d'informations sur la façon de permettre à un fichier SWF servi par un protocole sécurisé (HTTPS) d'accéder aux données, objets ou variables de fichiers SWF qui sont servis par des protocoles non sécurisés, consultez *A propos de l'autorisation d'accès du protocole HTTP à HTTPS entre fichiers SWF*, page 201. Pour plus d'informations sur la façon de permettre à un fichier SWF servi par un domaine de charger des données (à l'aide de `loadVariables()` par exemple) d'un autre domaine, consultez *A propos de l'autorisation de chargement de données inter-domaines*, page 201.

Pour plus d'informations sur la manière dont ces changements de sécurité ont des répercussions sur le contenu dans Flash MX et dans les versions précédentes, consultez *A propos de la compatibilité avec les précédents modèles de sécurité Flash Player*, page 202.

A propos de l'autorisation d'accès aux données entre des fichiers SWF inter-domaines

Un fichier SWF peut charger un autre fichier SWF depuis n'importe quel emplacement sur Internet. Cependant, pour que chacun des deux fichiers SWF puisse accéder aux données (variables et objets) de l'autre, ils doivent provenir du même domaine. Par défaut, dans Flash Player 7 et versions ultérieures, les deux domaines doivent correspondre exactement pour que les deux fichiers partagent des données. Un fichier SWF peut cependant accéder aux fichiers SWF servis par des domaines spécifiques en appelant `LocalConnection.allowDomain` ou `System.security.allowDomain()`.

Par exemple, supposons que le fichier main.swf soit servi à partir de www.macromedia.com. Ce fichier SWF charge alors un autre fichier SWF (data.swf) à partir de data.macromedia.com vers une occurrence de clip (`target_mc`).

```
// Dans macromedia.swf
target_mc.loadMovie("http://data.macromedia.com/data.swf");
```

De plus, supposons que le fichier data.swf définisse une méthode appelée `getData()` sur son scénario principal. Par défaut, le fichier main.swf ne peut pas appeler la méthode `getData()` définie dans le fichier data.swf, une fois que ce fichier a été chargé. Cela est dû au fait que les deux fichiers SWF ne résident pas sur le même domaine. En ce sens, l'appel de méthode suivant dans main.swf, une fois le chargement de data.swf effectué, échouera.

```
// Dans macromedia.swf, après le chargement de data.swf :
target_mc.getData(); // Cette méthode d'appel va échouer
```

Cependant, le fichier data.swf peut accéder à des fichiers SWF servis à partir de www.macromedia.com à l'aide du gestionnaire `LocalConnection.allowDomain` ou de la méthode `System.security.allowDomain()`, selon le type d'accès nécessaire. Le code suivant, ajouté au fichier data.swf, permet à un fichier SWF servi à partir de www.macromedia.com d'accéder à ses variables et méthodes :

```
// Dans data.swf
System.security.allowDomain("www.macromedia.com");
ma_lc.allowDomain = function(domaineDenvoi) {
  return(domaineDenvoi=="www.macromedia.com");
}
```

Notez que la commande `allowDomain` permet à tout fichier SWF du domaine autorisé de programmer tout autre fichier SWF dans le domaine qui autorise l'accès, à moins que le fichier SWF accédé soit hébergé sur un site utilisant un protocole sécurisé (HTTPS). Dans ce cas, vous devez utiliser la commande `allowInsecureDomain` plutôt que `allowDomain` ; consultez *A propos de l'autorisation d'accès du protocole HTTP à HTTPS entre fichiers SWF* ci-dessous.

Pour plus d'informations sur la mise en correspondance des noms de domaines, consultez *Fonctions de sécurité de Flash Player*, page 199.

A propos de l'autorisation d'accès du protocole HTTP à HTTPS entre fichiers SWF

Comme indiqué dans la section précédente, vous devez utiliser un gestionnaire ou une méthode `allowDomain` pour permettre à un fichier SWF d'un domaine d'accéder à un fichier SWF d'un autre domaine. Cependant, si le fichier SWF accédé est hébergé sur un site utilisant un protocole sécurisé (HTTPS), le gestionnaire ou la méthode `allowDomain` ne permet pas l'accès à partir du fichier SWF hébergé sur un site utilisant un protocole non sécurisé. Pour autoriser un tel accès, vous devez utiliser les instructions `LocalConnection.allowInsecure Domain()` ou `System.security.allowInsecureDomain()`.

Par exemple, si un fichier SWF à l'adresse http://www.unSite.com doit pouvoir accéder au fichier SWF à l'adresse https://www.unSite.com/data.swf, ajoutez le code suivant au fichier data.swf :

```
// Dans data.swf
System.security.allowInsecureDomain("www.unSite.com");
ma_lc.allowInsecureDomain = function(domaineDenvoi) {
  return(domaineDenvoi=="www.unSite.com");
}
```

A propos de l'autorisation de chargement de données inter-domaines

Un document Flash peut charger les données depuis une source externe à l'aide de l'un des appels de chargement de données suivants : `XML.load()`, `XML.sendAndLoad()`, `LoadVars.load()`, `LoadVars.sendAndLoad()`, `loadVariables()` et `loadVariablesNum()`. De plus, un fichier SWF peut importer des bibliothèques partagées à l'exécution ou des actifs définis dans un autre fichier SWF, au moment de l'exécution. Par défaut, les données ou le support SWF (dans le cas de bibliothèques partagées à l'exécution) doivent résider dans le même domaine que le fichier SWF qui charge ces données externes ou ce support.

Pour que les fichiers SWF situés dans différents domaines puissent accéder aux données et aux actifs contenus dans des bibliothèques partagées à l'exécution, utilisez un *fichier de régulation inter-domaines*. Il s'agit d'un fichier XML qui permet au serveur d'indiquer que ses données et ses documents sont disponibles pour les fichiers SWF servis par certains domaines ou par tous les domaines. Tout fichier SWF servi par un domaine spécifié par le fichier de régulation du serveur peut accéder aux données et aux actifs de ce serveur.

Lorsqu'un document Flash tente d'accéder aux données depuis un autre domaine, Flash Player essaie automatiquement de charger un fichier de régulation depuis ce domaine. Si le domaine du document Flash qui tente d'accéder aux données est inclus dans le fichier de régulation les données sont automatiquement accessibles.

Les fichiers de régulation doivent être nommés interdomaine.xml et se trouvent dans le répertoire racine du serveur qui sert les données. Les fichiers de régulation ne fonctionnent que sur des serveurs communiquant en HTTP, HTTPS, ou FTP. Le fichier de régulation est spécifique au port et au protocole du serveur dans lequel il réside.

Par exemple, un fichier de régulation situé dans https://www.macromedia.com:8080/interdomaine.xml ne s'appliquera qu'aux appels de chargement de données passés vers www.macromedia.com sur HTTPS au port 8080.

Cette règle a une exception : lorsque vous utilisez un objet XMLSocket pour vous connecter à un serveur socket dans un autre domaine. Dans ce cas, un serveur HTTP exécuté sur le port 80 du même domaine que le serveur socket doit fournir le fichier de régulation pour l'appel de la méthode.

Un fichier de régulation XML contient une seule balise `<régulation-inter-domaine>`, qui contient elle-même aucune ou plusieurs balises `<autoriser-accès-depuis>`. Chaque balise `<autoriser-accès-depuis>` contient un attribut, `domaine`, qui spécifie une adresse IP exacte, un domaine exact ou un domaine générique (domaine quelconque). Les domaines génériques sont indiqués par un astérisque (qui correspond à tous les domaines et à toutes les adresses IP) ou par un astérisque (*) suivi d'un suffixe, qui correspond uniquement aux domaines se terminant par le suffixe spécifié. Les suffixes doivent commencer par un point. Cependant, les domaines génériques suivis de suffixes peuvent correspondre à des domaines qui sont composés uniquement du suffixe sans le point de séparation. Par exemple, foo.com est considéré comme un élément de *.foo.com. Les caractères génériques ne sont pas autorisés dans les spécifications de domaine IP.

Si vous spécifiez une adresse IP, seuls les fichiers SWF chargés depuis cette adresse IP à l'aide de la syntaxe IP (par exemple, http://65.57.83.12/flashmovie.swf) sont accessibles ; les fichiers chargés à l'aide d'une syntaxe domaine-nom ne sont pas accessibles. Flash Player n'effectue pas de résolution DNS.

Voici un exemple de fichier de régulation permettant d'accéder à des documents Flash qui proviennent de foo.com, amiDeFoo.com, *.foo.com et 105.216.0.40, depuis un document Flash situé sur foo.com :

```
<<?xml version="1.0" ?>>
<!-- http://www.foo.com/interdomaine.xml -->
<régulation-inter-domaines>
  <autoriser-accès-depuis domaine="www.amiDeFoo.com" />
  <autoriser-accès-depuis domaine="*.foo.com" />
  <autoriser-accès-depuis domaine="105.216.0.40" />
</régulation-inter-domaines>
```

Un fichier de régulation ne contenant aucune balise `<autoriser-accès-depuis>` revient à ne pas avoir de régulation sur un serveur.

A propos de la compatibilité avec les précédents modèles de sécurité Flash Player

Suite aux changements des fonctions de sécurité dans Flash Player (consultez *Fonctions de sécurité de Flash Player*, page 199), un contenu exécuté correctement sous Flash Player 6 ou antérieur peut ne pas s'exécuter correctement sous Flash Player 7 ou ultérieur.

Par exemple, dans Flash Player 6, un fichier SWF se trouvant dans www.macromedia.com peut accéder à des données sur un serveur situé à l'adresse data.macromedia.com. Ainsi, Flash Player 6 a autorisé un fichier SWF provenant d'un domaine à charger des données à partir d'un domaine « similaire ».

Dans Flash Player 7 et ultérieur, si un fichier SWF version 6 (ou antérieure) tente de charger des données à partir d'un serveur se trouvant dans un autre domaine, et que ce serveur ne fournit aucun fichier de régulation autorisant l'accès à partir du domaine de ce fichier SWF, la boîte de dialogue Paramètres de Macromedia Flash Player apparaît. Elle demande à l'utilisateur d'autoriser ou de refuser l'accès aux données interdomaines.

Si l'utilisateur clique sur Autoriser, le fichier SWF peut accéder aux données requises ; s'il clique sur Refuser, le fichier SWF ne peut pas accéder aux données requises.

Pour empêcher l'apparition de cette boîte de dialogue, créez un fichier de régulation de sécurité sur le serveur fournissant les données. Pour plus d'informations, consultez *A propos de l'autorisation de chargement de données inter-domaines*, page 201.

CHAPITRE 11
Utilisation de médias externes

Si vous importez une image ou un son pendant la création d'un document dans Macromedia Flash MX 2004 ou Macromedia Flash MX Professionnel 2004, cette image ou ce son est placé dans le fichier SWF lorsque vous le publiez. En plus de l'importation de médias pendant la programmation, vous pouvez charger un média externe à l'exécution. Il existe plusieurs raisons pour lesquelles il est souhaitable de conserver les médias en dehors d'un document Flash.

Réduire la taille des fichiers En conservant les fichiers média volumineux en dehors de votre document Flash et en les chargeant à l'exécution, vous pouvez réduire le délai initial de téléchargement de vos applications et présentations, particulièrement dans le cas de connexions Internet lentes.

Modulariser les présentations volumineuses Vous pouvez diviser une présentation ou une application volumineuse en fichiers SWF séparés, puis les charger au fur et à mesure à l'exécution. Cela permet non seulement de réduire le délai initial de téléchargement, mais également de conserver et de mettre à jour les contenus de la présentation plus facilement.

Séparer le contenu de la présentation Cela est très commun dans le développement d'applications, particulièrement les applications orientées données. Par exemple, une application avec un panier pourrait afficher une image JPEG de chaque produit. En chargeant les fichiers JPEG de chaque image à l'exécution, vous pouvez facilement mettre à jour l'image d'un produit sans modifier le fichier FLA d'origine.

Bénéficier des fonctions d'exécution seule Certaines fonctions, telles que la lecture en continu des fichiers FLV et MP3, ne sont utilisables qu'à l'exécution dans ActionScript.

Aperçu du chargement de média externe

Vous pouvez charger quatre types de fichiers média dans une application Flash lors de l'exécution : les fichiers SWF, MP3, JPEG et FLV. Flash Player peut charger un média externe depuis n'importe quelle adresse HTTP ou FTP, un disque local utilisant un chemin relatif ou à l'aide du protocole `file://`.

Pour charger des fichiers SWF et JPEG externes, vous pouvez utiliser la fonction `loadMovie()` ou `loadMovieNum()`, ou bien la méthode `MovieClip.loadMovie()`. Lorsque vous chargez un fichier SWF ou JPEG, vous spécifiez un clip ou un niveau d'animation comme cible de ce média. Pour plus d'informations sur le chargement des fichiers SWF et JPEG, consultez *Chargement de fichiers SWF et JPEG externes*, page 206.

Pour lire un fichier MP3 (MPEG Couche 3) externe, utilisez la méthode `loadSound()` de la classe Sound. Cette méthode vous permet de spécifier si le fichier MP3 doit être diffusé en flux continu ou complètement téléchargé avant de démarrer la lecture. Vous pouvez également lire les informations ID3 intégrées dans les fichiers MP3, si elles sont disponibles. Pour plus d'informations, consultez *Lecture des balises ID3 dans les fichiers MP3*, page 208.

Flash Video (FLV) est le format vidéo natif utilisé par Flash Player. Vous pouvez lire les fichiers FLV sur HTTP ou sur le système de fichiers local. La lecture des fichiers FLV externes offre plusieurs avantages par rapport à l'intégration de vidéo dans un document Flash : performances et gestion de la mémoire améliorées, indépendance des cadences vidéo et Flash. Pour plus d'informations, consultez *Lecture dynamique des fichiers FLV externes*, page 209.

Vous pouvez également précharger ou suivre la progression du téléchargement d'un média externe. Flash Player 7 introduit la classe MovieClipLoader que vous pouvez utiliser pour suivre la progression du téléchargement de fichiers SWF ou JPEG. Pour précharger des fichiers MP3 FLV, vous pouvez utiliser la méthode `getBytesLoaded()` de la classe Sound et la propriété `bytesLoaded` de la classe NetStream. Pour plus d'informations, consultez *Préchargement de média externe*, page 210.

Chargement de fichiers SWF et JPEG externes

Pour charger un fichier SWF ou JPEG, utilisez la fonction globale `loadMovie()` ou `loadMovieNum()` ou la méthode `loadMovie()` de la classe MovieClip. Pour charger un fichier SWF ou JPEG dans un niveau de Flash Player, utilisez `loadMovieNum()`. Pour charger un fichier SWF or JPEG dans une cible de clip, utilisez la fonction ou la méthode `loadMovie()`. Dans les autres cas, le contenu chargé remplace le contenu du niveau ou du clip cible spécifié.

Lorsque vous chargez un fichier SWF ou JPEG dans un clip cible, le coin supérieur gauche du fichier SWF ou de l'image JPEG est placé sur le point d'alignement du clip. Ce point se trouvant souvent au centre du clip, il se peut que le contenu chargé ne soit pas centré. De même, lorsque vous chargez un fichier SWF ou une image JPEG dans un scénario racine, son coin supérieur gauche est placé sur le coin supérieur gauche de la scène. Le contenu chargé hérite de la rotation et de la mise à l'échelle du clip, mais le contenu d'origine du clip est supprimé.

Vous pouvez éventuellement envoyer les variables ActionScript en appelant la fonction `loadMovie()` ou `loadMovieNum()`. Cela est particulièrement utile si, par exemple, l'URL spécifiée dans l'appel d'une méthode est un script côté serveur qui renvoie une image JPEG ou un fichier SWF en fonction des données transmises depuis l'application Flash.

En ce qui concerne les fichiers d'image, Flash ne supporte que le type de fichier d'image JPEG standard, et non les fichiers JPEG progressifs.

Lorsque vous utilisez la fonction globale `loadMovie()` ou `loadMovieNum()`, spécifiez le niveau ou le clip cible en tant que paramètre. Par exemple, le code suivant permet de charger le fichier d'animation Flash contenu.swf dans l'occurrence de clip appelée `cible_mc` :

```
loadMovieNum("contenu.swf", cible_mc);
```

De même, vous pouvez utilisez `Clip.loadMovie()` pour obtenir le même résultat :

```
cible_mc.loadMovie("contenu.swf");
```

Le code suivant charge l'image JPEG fleurs.jpg dans l'occurrence de clip `image_clip` :

```
image_clip.loadMovie("fleurs.jpg");
```

Pour plus d'informations sur `loadMovie()`, `loadMovieNum()` et `MovieClip.loadMovie`, consultez les entrées correspondantes dans le Chapitre 12, *Dictionnaire ActionScript*, page 215.

A propos des fichiers SWF chargés et du scénario racine

La propriété ActionScript `_root` spécifie ou renvoie une référence au scénario racine d'un fichier SWF. Si vous chargez un fichier SWF dans le clip d'un autre fichier SWF, les références à `_root` dans le fichier SWF chargé se traduisent dans le scénario racine du fichier SWF hôte et non pas dans celui du fichier SWF chargé. Cela peut parfois engendrer un comportement inattendu à l'exécution, par exemple, si le fichier SWF hôte et le fichier SWF chargé utilisent tous les deux `_root` pour spécifier une variable.

Dans Flash Player 7 et ses versions ultérieures, vous pouvez utiliser la propriété `MovieClip._lockroot` pour obliger les références à `_root` réalisées dans un clip à se traduire dans son propre scénario, plutôt que dans celui de l'animation contenant le clip. Pour plus d'informations, consultez *Spécification d'un scénario racine pour les fichiers SWF chargés*, page 130.

A propos de l'accès aux données des fichiers SWF chargés

Un fichier SWF peut charger un autre fichier SWF depuis n'importe quel emplacement sur Internet. Cependant, pour qu'un fichier SWF accède aux données (variables, méthodes, etc.) définies dans l'autre fichier SWF, les deux fichiers doivent provenir du même domaine. Dans Flash Player 7 et ses versions ultérieures, toute écriture de scripts inter-domaines est interdite sauf si le fichier SWF chargé en décide autrement en appelant `System.security.allowDomain()`.

Pour plus d'informations, consultez *Fonctions de sécurité de Flash Player*, page 199 et `System.security.allowDomain()` dans le Chapitre 12, *Dictionnaire ActionScript*, page 215.

Chargement des fichiers externes MP3

Pour charger des fichiers MP3 à l'exécution, utilisez la méthode `loadSound()` de la classe Sound. Créez tout d'abord un objet Sound :

```
var chanson_1_sound = new Sound();
```

Utilisez ensuite le nouvel objet pour appeler la méthode `loadSound()` et charger un événement ou un son lu en flux continu. Alors que les sons d'événement sont entièrement chargés avant leur lecture, les sons lus en flux continu sont lus pendant leur téléchargement. Vous pouvez définir le paramètre *`isStreaming`* de la méthode `loadSound()` de manière à définir un son comme étant un son d'événement ou un son lu en flux continu. Après avoir chargé un son d'événement, vous devez appeler la méthode `start()` de la classe Sound pour lancer sa lecture. La lecture des sons lus en flux continu débute dès qu'une quantité suffisante de données a été chargée dans le fichier SWF et il n'est pas nécessaire d'utiliser la méthode `start()`.

Par exemple, le code suivant crée un objet Sound nommé `classique` et charge ensuite un fichier MP3 nommé beethoven.mp3 :

```
var classique.Sound = new Sound();
classique.loadSound("http://server.com/mp3s/beethoven.mp3", true);
```

Dans la plupart des cas, définissez le paramètre *`isStreaming`* sur la valeur `true`, particulièrement si vous chargez des fichiers sons volumineux qui doivent démarrer dès que possible, par exemple, lorsque vous créez une application « juke-box » MP3. Cependant, si vous téléchargez des clips son moins volumineux et devez les exécuter à un moment précis (par exemple, lorsqu'un utilisateur clique sur un bouton), définissez le paramètre *`isStreaming`* sur la valeur `false`.

Pour déterminer la fin du téléchargement d'un son, utilisez le gestionnaire d'événement `Son.onLoad`. Ce gestionnaire d'événement reçoit automatiquement une valeur booléenne (`true` ou `false`) qui indique si le fichier a bien été téléchargé correctement.

Par exemple, supposez que vous créez un jeu en ligne qui utilise différents sons en fonction du niveau atteint par l'utilisateur dans ce jeu. Le code suivant charge un fichier MP3 (blastoff.mp3) dans un objet Sound appelé `sonJeu`, puis lit le son à la fin du téléchargement :

```
var sonJeu = new Sound();
sonJeu.onLoad = function (chargementOK) {
  if(chargementOK) {
    sonJeu.start();
  }
}
sonJeu.loadSound("http://server.com/sounds/blastoff.mp3", false);
```

Pour les fichiers son, Flash Player ne supporte que le type de fichier MP3.

Pour plus d'informations, consultez `Sound.loadSound`, `Sound.start()` et `Sound.onLoad` dans le Chapitre 12, *Dictionnaire ActionScript*, page 215.

Lecture des balises ID3 dans les fichiers MP3

Les balises ID3 sont des champs de données ajoutés à un fichier MP3 contenant des informations sur ce fichier, comme le nom de la chanson, le nom de l'album et de l'artiste.

Pour lire les balises ID3 d'un fichier MP3, utilisez la propriété `Sound.ID3`, dont les propriétés correspondent aux noms des balises ID3 incluses dans le fichier MP3 en cours de téléchargement. Pour déterminer la disponibilité des balises ID3 pour télécharger un fichier MP3, utilisez le gestionnaire d'événement `Sound.onID3`. Flash Player7 prend en charge les balises des versions 1.0, 1.1, 2.3 et 2.4 ; les balises de la version 2.2 ne sont pas prises en charge.

Par exemple, le code suivant charge un fichier MP3 nommé chansonPreferee.mp3 dans l'objet Sound nommé `chanson`. Lorsque les balises ID3 sont disponibles pour le fichier, un champ de texte nommé `affichage_txt` affiche le nom de l'artiste et de la chanson.

```
var chanson = new Sound();
chanson.onID3 = function () {
  affichage_txt.text = "Artiste : " + song.id3.TCOM + newline;
  affichage_txt.text += "Chanson : " + song.id3.TIT2);
}
son.loadSound("mp3s/chansonPreferee.mp3, true");
```

Les balises ID3 2.0 se trouvant au début d'un fichier MP3 (avant les données audio), elles sont disponibles dès que le fichier démarre le téléchargement. Cependant, les balises ID3 1.0 se trouvent à la fin du fichier (après les données audio) ; elles ne sont pas disponibles tant que le téléchargement du fichier MP3 n'est pas terminé.

Le gestionnaire d'événement `onID3` est appelé à chaque fois que de nouvelles données ID3 sont disponibles. Cela signifie que si un fichier MP3 contient des balises ID3 2.0 et ID3 1.0, le gestionnaire d'événement `onID3` sera appelé deux fois, car les balises se trouvent dans des parties différentes du fichier.

Pour obtenir une liste des balises ID3 prises en charge, consultez `Sound.ID3`, page 715.

Lecture dynamique des fichiers FLV externes

Plutôt que d'importer des données vidéo dans l'environnement auteur de Flash, vous pouvez utiliser ActionScript pour lire dynamiquement les fichiers FLV externes dans Flash Player. Vous pouvez lire les fichiers FLV à partir d'une adresse HTTP ou du système de fichiers local. Pour lire les fichiers FLV, utilisez les classes NetConnection et NetStream, ainsi que la méthode `attachVideo()` de la classe Video. (Pour obtenir des informations complètes, consultez les entrées *Classe NetConnection*, *Classe NetStream* et `Video.attachVideo()` dans le Chapitre 12, *Dictionnaire ActionScript*, page 215.)

Vous pouvez créer des fichiers FLV en important la vidéo dans l'outil de programmation Flash et en l'exportant sous la forme d'un fichier FLV. (Consultez « Vidéo Macromedia Flash » dans le guide Utilisation de Flash de l'aide.) Si vous possédez Flash Professionnel, vous pouvez utiliser le module d'exportation de FLV pour exporter les fichiers FLV à partir d'applications d'édition vidéo prises en charge. (Consultez Exportation des fichiers FLV à partir des applications d'édition vidéo (Flash Professionnel uniquement) dans le guide Utilisation de Flash de l'aide.)

L'utilisation des fichiers FLV externes offre certaines fonctionnalités qui ne sont pas disponibles avec l'utilisation de la vidéo importée :

- Vous pouvez utiliser des clips vidéo plus longs dans vos documents Flash sans ralentir la lecture de l'animation. Les fichiers FLV externes sont lus à l'aide de la *mémoire cache*. Cela signifie que les fichiers volumineux sont stockés en petites parties et sont accessibles dynamiquement, et qu'ils ne nécessitent pas autant de mémoire que les fichiers vidéo intégrés.
- Un fichier FLV externe peut avoir une cadence différente de celle du document Flash dans lequel il est lu. Par exemple, vous pouvez définir la cadence du document Flash à 30 ips et celle de l'image vidéo à 21 ips. Cela vous octroie un meilleur contrôle et garantit la fluidité du flux vidéo.
- Avec les fichiers FLV externes, la lecture du document Flash n'a pas besoin d'être interrompue pendant le chargement du fichier vidéo. Les fichiers vidéo importés peuvent parfois interrompre la lecture du document pour exécuter certaines fonctions, par exemple, accéder à un lecteur de CD-ROM. Les fichiers FLV peuvent exécuter les fonctions indépendamment du document Flash et ainsi ne pas interrompre la lecture.
- Le sous-titrage du contenu vidéo est plus facile avec les fichiers FLV externes, parce que vous utilisez les gestionnaires d'événements pour accéder aux métadonnées de la vidéo.

La procédure suivante illustre comment lire un fichier nommé videoFile.flv stocké au même emplacement que votre fichier SWF.

Pour lire un fichier FLV externe dans un document Flash :

1. Si le document est ouvert dans l'outil de programmation Flash, dans le panneau Bibliothèque (Fenêtre > Bibliothèque), sélectionnez Nouvelle vidéo dans le menu d'options de la bibliothèque pour créer un objet vidéo.
2. Faites glisser un objet vidéo depuis le panneau Bibliothèque sur la scène. Une occurrence de l'objet vidéo est créée.
3. Lorsque l'objet vidéo est sélectionné sur la scène, dans l'inspecteur des propriétés (Fenêtre > Propriétés), entrez `ma_video` dans le champ de texte Nom de l'occurrence.
4. Ouvrez le panneau Composants (Fenêtre > Panneaux de développement > Composants) et faites glisser un composant TextArea sur la scène.
5. Lorsque l'objet TextArea est sélectionné sur la scène, entrez `status` dans le champ de texte Nom de l'occurrence, dans l'inspecteur des propriétés.

6 Sélectionnez l'image 1 dans le scénario et ouvrez le panneau Actions (Fenêtre > Panneaux de développement > Actions).

7 Ajoutez le code suivant au panneau Actions

```
// Créer un objet NetConnection :
var netConn:NetConnection = new NetConnection();
// Créer une connexion locale en flux continu :
netConn.connect(null);
// Créer un objet NetStream et définir une fonction onStatus() :
var netStream:NetStream = new NetStream(netConn);
netStream.onStatus = function(infoObject) {
  status.text += "Status (NetStream)" + newline;
  status.text += "Level: "+infoObject.level + newline;
  status.text += "Code: "+infoObject.code + newline;
};
// Associer la vidéo NetStream à l'objet Video :
ma_video.attachVideo(netStream);
// Définir la durée du tampon :
netStream.setBufferTime(5);
// Lire le fichier FLV :
netStream.play("videoFile.flv");
```

Préchargement de média externe

ActionScript offre plusieurs façons de précharger ou de suivre la progression du téléchargement du média externe. Pour précharger les fichiers SWF et JPEG, utilisez la classe MovieClipLoader, qui offre un mécanisme d'écouteur d'événements permettant de vérifier la progression du téléchargement. Cette classe est nouvelle dans Flash Player 7. Pour plus d'informations, consultez *Préchargement des fichiers SWF et JPEG*, page 210.

Pour suivre la progression du téléchargement des fichiers MP3, utilisez les méthodes `Sound.getBytesLoaded()` et `Sound.getBytesTotal()` ; pour suivre la progression du téléchargement des fichiers FLV, utilisez les propriétés `NetStream.bytesLoaded` et `NetStream.bytesTotal`. Pour plus d'informations, consultez *Préchargement des fichiers MP3 et FLV*, page 212.

Préchargement des fichiers SWF et JPEG

Pour précharger les fichiers SWF et JPEG dans les occurrences de clip, vous pouvez utiliser la *Classe MovieClipLoader*. Cette classe offre un mécanisme d'écouteur d'événements qui indique l'état des téléchargements de fichiers dans les clips. L'utilisation d'un objet MovieClipLoader pour précharger des fichiers SWF et JPEG implique les étapes suivantes :

Créer un objet MovieClipLoader Vous pouvez utiliser un seul objet MovieClipLoader pour suivre la progression du téléchargement de plusieurs fichiers ou créer un objet distinct pour suivre la progression des fichiers individuels.

```
var loader:MovieClipLoader = new MovieClipLoader();
```

Créer un objet d'écoute et créer des gestionnaires d'événements L'objet d'écoute peut être n'importe quel objet ActionScript, par exemple un objet Object générique, un clip ou un composant personnalisé.

Par exemple, le code suivant crée un objet d'écoute générique nommé `loadListener` et définit pour lui-même les fonctions `onLoadStart`, `onLoadProgress` et `onLoadComplete`.

```
// Créer un objet d'écoute :
var loadListener:Object = new Object();
loadListener.onLoadStart = function (loadTarget) {
  trace("Le chargement dans " + loadTarget + " a commencé.");
}
loadListener.onLoadProgress = function(loadTarget, octetsChargés, octetsTotal)
  {
  var percentLoaded = bytesLoaded/bytesTotal * 100;
  trace("%" + percentLoaded + " vers cible " + loadTarget);
}
loadListener.onLoadComplete = function(loadTarget) {
  trace("Chargement terminé dans : " + loadTarget);
}
```

Enregistrer l'objet d'écoute avec l'objet MovieClipLoader Vous devez enregistrer l'objet d'écoute avec l'objet MovieClipLoader pour qu'il reçoive les événements chargés.

```
loader.addListener(loadListener);
```

Commencer à charger le fichier (JPEG ou SWF) dans un clip cible Pour commencer le téléchargement du fichier JPEG ou SWF, utilisez la méthode `MovieClipLoader.loadClip()`.

```
loader.loadClip("scene_2.swf");
```

Remarque : Vous ne pouvez utiliser que les méthodes MovieClipLoader pour suivre la progression du téléchargement des fichiers chargés avec la méthode `MovieClipLoader.loadClip()`. Vous ne pouvez pas utiliser la fonction `loadMovie()` ou la méthode `MovieClip.loadMovie()`.

L'exemple suivant utilise la méthode `setProgress()` du composant ProgressBar pour afficher la progression du téléchargement d'un fichier SWF. (Consultez « composant ProgressBar » dans le guide Utilisation des composants de l'aide.)

Pour afficher la progression du téléchargement en utilisant le composant ProgressBar :

1. Dans un nouveau document Flash, créez un clip sur la scène et donnez-lui le nom `cible_mc`.
2. Ouvrez le panneau Composants (Fenêtre > Panneaux de développement > Composants).
3. Faites glisser un composant ProgressBar du panneau Composants jusqu'à la scène.
4. Dans l'inspecteur des propriétés, donnez le nom `pBar` au composant ProgressBar et, dans le volet Paramètres, sélectionnez Manuel dans le menu contextuel Mode.
5. Sélectionnez l'image 1 dans le scénario et ouvrez le panneau Actions (Fenêtre > Panneaux de développement > Actions).
6. Ajoutez le code suivant au panneau Actions

```
// créez un objet MovieClipLoader et un objet écouteur
monChargeur = new MovieClipLoader();
monEcouteur = new Object();
// ajoutez les rappels de MovieClipLoader à votre objet écouteur
monEcouteur.onLoadStart = function(clip) {
  // cet événement est déclenché une fois, lors du lancement du chargement
  pBar.label = "Chargement en cours : " + clip;
};
monEcouteur.onLoadProgress = function(clip, octetsChargés, octetsTotal) {
  var pourcentageChargé = int (100 x (octetsChargés/octetsTotal));
  pBar.setProgress(bytesLoaded, bytesTotal);
};monChargeur.addListener(monEcouteur);
monChargeur.loadClip("Fichiertrèsvolumineux.swf", cible_mc);
```

7 Testez le document en sélectionnant Contrôle > Tester l'animation.

Pour plus d'informations, consultez l'entrée *Classe MovieClipLoader* dans le Chapitre 12, *Dictionnaire ActionScript*, page 215.

Préchargement des fichiers MP3 et FLV

Pour précharger des fichiers MP3 et FLV, vous pouvez utiliser la fonction `setInterval()` pour créer un mécanisme « d'interrogation » qui vérifie les octets chargés par un objet Sound ou NetStream à des intervalles prédéterminés. Pour suivre la progression du téléchargement des fichiers MP3, utilisez les méthodes `Sound.getBytesLoaded()` et `Sound.getBytesTotal()` ; pour suivre la progression du téléchargement des fichiers FLV, utilisez les propriétés `NetStream.bytesLoaded` et `NetStream.bytesTotal`.

Le code suivant utilise `setInterval()` pour vérifier les octets chargés par un objet Sound ou NetStream à des intervalles prédéterminés.

```
// Créez un nouvel objet Sound pour lire le son.
var pisteSon = new Sound();
// Créez la fonction d'interrogation qui suit la progression du téléchargement.
// Il s'agit de la fonction qui est "interrogée". Elle vérifie
// la progression du téléchargement de l'objet Sound transmis comme une
  référence.
checkProgress = function (objSon) {
  var octetsChargés = objSon.getBytesLoaded();
  var octetsTotal = objSon.getBytesTotal();
  var pourcentageChargé = Math.floor(octetsChargés/octetsTotal x 100);
  trace("%" + pourcentageChargé + " chargé(s).");
}
// Lorsque le fichier a terminé le chargement, supprimez l'intervalle
  d'interrogation.
pisteSon.onLoad = function () {
  clearInterval(interrogation);
}
// Charger le fichier MP3 en flux continu et commencer à appeler
  checkProgress()
pisteSon.loadSound("beethoven.mp3", true);
var interrogation = setInterval(checkProgress, 1000, pisteSon);
```

Vous pouvez utiliser ce même type de technique d'interrogation pour précharger des fichiers FLV externes. Pour obtenir le nombre total d'octets et le nombre d'octets chargés pour un fichier FLV, utilisez les propriétés `NetStream.bytesLoaded` et `NetStream.bytesTotal`.

Une autre façon de précharger les fichiers FLV consiste à utiliser la méthode `NetStream.setBufferTime()`. Cette méthode prend un seul paramètre indiquant le nombre de secondes du flux FLV à télécharger avant le démarrage de la lecture.

Pour plus d'informations, consultez `MovieClip.getBytesLoaded()`, `MovieClip.getBytesTotal()`, `NetStream.bytesLoaded`, `NetStream.bytesTotal`, `NetStream.setBufferTime()`, `setInterval()`, `Sound.getBytesLoaded()` et `Sound.getBytesTotal()`dans le Chapitre 12, *Dictionnaire ActionScript*, page 215.

PARTIE V
Référence

Cette section contient le dictionnaire ActionScript, qui fournit des informations sur la syntaxe et l'utilisation des éléments du langage ActionScript. Elle contient également des annexes proposant des informations de référence que vous pouvez consulter lors de la rédaction de vos scripts.

CHAPITRE 12
Dictionnaire ActionScript

Ce dictionnaire décrit la syntaxe et l'utilisation des éléments ActionScript dans Macromedia Flash MX 2004 et Macromedia Flash MX Professionnel 2004. Pour utiliser les exemples de ce dictionnaire dans un script, copiez le code et collez-le dans la fenêtre de script ou dans un fichier de script externe. Ce dictionnaire répertorie l'ensemble des éléments ActionScript : opérateurs, mots-clés, instructions, actions, propriétés, fonctions, classes et méthodes. Pour une vue d'ensemble de toutes les entrées du dictionnaire, consultez *Contenu du dictionnaire*, page 217 ; les tableaux de cette section sont un bon point de départ pour rechercher des opérateurs symboles ou des méthodes dont vous ne connaissez pas la classe. Pour plus d'informations sur les composants, veuillez consulter le guide *Utilisation des composants.*

Ce dictionnaire contient deux types d'entrées différents :

- Des entrées individuelles pour les opérateurs, les mots-clés, les fonctions, les variables, les propriétés, les méthodes et les instructions.
- Des entrées de classe, qui fournissent des informations générales sur les classes intégrées.

Utilisez les informations contenues dans les entrées en exemple pour interpréter la structure et les conventions utilisées dans ces types d'entrées.

Exemple d'entrée pour la plupart des éléments ActionScript

L'exemple d'entrée suivant illustre les conventions utilisées pour tous les éléments ActionScript qui ne sont pas des classes.

Titre de l'entrée

Toutes les entrées sont classées par ordre alphabétique. L'ordre alphabétique ne tient pas compte des majuscules, des traits de soulignement de préfixe, et ainsi de suite.

Disponibilité

Sauf mention contraire, la section Disponibilité indique les versions de Flash Player qui supportent l'élément. Ceci est différent de la version de Flash utilisée pour développer le contenu. Par exemple, si vous utilisez Macromedia Flash MX 2004 ou Macromedia Flash MX Professionnel 2004 pour créer des contenus pour Flash Player 6, vous ne pourrez utiliser que les éléments d'ActionScript disponibles dans Flash Player 6.

Dans quelques cas rares, cette section indique également la version de l'outil de programmation supportant l'élément. Pour obtenir un exemple, consultez `System.setClipboard()`.

Enfin, si un élément n'est supporté que dans ActionScript 2.0, cette section le mentionne également.

Usage

Cette section présente la syntaxe correcte d'utilisation des éléments ActionScript dans votre code. La portion obligatoire de la syntaxe est indiquée en `police` de `code` ; le code que vous devez fournir est en *`italique`*. Les crochets (`[]`) indiquent des paramètres facultatifs.

Paramètres

Cette section décrit les paramètres répertoriés dans la syntaxe.

Renvoie

Cette section identifie les valeurs renvoyées par l'élément (le cas échéant).

Description

Cette section identifie le type d'élément (par exemple, opérateur, méthode, fonction ou autre) et décrit son utilisation.

Exemple

Cette section fournit un exemple de code illustrant l'utilisation de l'élément.

Consultez également

Cette section répertorie les entrées connexes du dictionnaire ActionScript.

Exemple d'entrée pour les classes

L'exemple d'entrée suivant illustre les conventions utilisées pour les classes ActionScript intégrées. Les classes sont répertoriées par ordre alphabétique avec tous les autres éléments du dictionnaire.

Titre de l'entrée

Le titre de l'entrée fournit le nom de la classe. Le nom de la classe est suivi d'une description générale.

Tableaux récapitulatifs des méthodes et propriétés

Chaque entrée de classe contient un tableau répertoriant toutes les méthodes associées. Si la classe possède des propriétés (souvent des constantes), des gestionnaires d'événement ou des écouteurs d'événement, ces éléments sont répertoriés dans un tableau complémentaire. Tous les éléments répertoriés dans ces tableaux disposent également de leur propre entrée dans le dictionnaire, à la suite de l'entrée de la classe correspondante.

Constructeur

Si une classe requiert l'utilisation d'un constructeur pour accéder à ses méthodes et propriétés, ledit constructeur est décrit dans chaque entrée de classe. La description du constructeur contient les mêmes éléments standard (syntaxe, description, etc.) que les autres entrées de dictionnaire.

Listes des méthodes et propriétés

Les méthodes et propriétés de la classe sont répertoriées dans l'ordre alphabétique après l'entrée de la classe.

Contenu du dictionnaire

Toutes les entrées du dictionnaire sont classées par ordre alphabétique. Cependant, certains opérateurs sont des symboles et sont classés dans l'ordre ASCII. En outre, les méthodes associées à une classe sont répertoriées avec le nom de la classe. Ainsi, la méthode `abs()` de la classe Math est répertoriée comme `Math.abs()`.

Les deux tableaux suivants vous aideront à trouver ces éléments. Le premier tableau répertorie les opérateurs symboles dans leur ordre d'apparition dans le dictionnaire. Le second tableau répertorie tous les autres éléments ActionScript.

Opérateurs symboles	Consultez l'entrée
--	-- (décrémentation)
++	++ (incrémentation)
!	! (NOT logique)
!=	!= (inégalité)
!==	!== (inégalité stricte)
%	% (modulo)
%=	%= (affectation de modulo)
&	& (Opérateur AND au niveau du bit)
&&	&& (AND logique)
&=	&= (affectation AND au niveau du bit)
()	() (parenthèses)
-	- (moins)
*	* (multiplication)
*=	*= (affectation de multiplication)
,	, (virgule)
.	.(point)
:	: (type)
?:	?: (conditionnel)
/	/ (division)
//	// (délimiteur de commentaires)
/*	/* (délimiteur de commentaires)
/=	/= (affectation de division)
[]	[] (accès tableau)
^	^ (XOR au niveau du bit)

Opérateurs symboles	Consultez l'entrée
^=	^= (affectation XOR au niveau du bit)
{}	{} (initialisateur d'objet)
|	| (OR au niveau du bit)
||	|| (OR logique)
|=	|= (affectation OR au niveau du bit)
~	~ (NOT au niveau du bit)
+	+ (addition)
+=	+= (affectation d'addition)
<	<(inférieur à)
<<	<< (décalage gauche au niveau du bit)
<<=	<<= (décalage gauche au niveau du bit et affectation)
<=	<= (inférieur ou égal à)
<>	<> (inégalité)
=	= (affectation)
-=	-= (affectation de soustraction)
==	== (égalité)
===	=== (égalité stricte)
>	> (supérieur à)
>=	>= (supérieur ou égal à)
>>	>> (décalage droit au niveau du bit)
>>=	>>= (décalage droit au niveau du bit et affectation)
>>>	>>> (décalage droit non signé au niveau du bit)
>>>=	>>>= (décalage droit non signé au niveau du bit et affectation)

Le tableau suivant répertorie tous les éléments ActionScript qui ne sont pas des opérateurs symboles.

Elément ActionScript	Consultez l'entrée
#endinitclip	#endinitclip
#include	#include
#initclip	#initclip
__proto__	Object.__proto__
_accProps	_accProps
_alpha	MovieClip._alpha, Button._alpha, TextField._alpha
_currentframe	MovieClip._currentframe
_droptarget	MovieClip._droptarget

Elément ActionScript	Consultez l'entrée
_focusrect	_focusrect, Button._focusrect, MovieClip._focusrect
_framesloaded	MovieClip._framesloaded
_global	_global object
_height	Button._height, MovieClip._height, TextField._height
_highquality	_highquality, Button._highquality, MovieClip._highquality, TextField._highquality
_lockroot	MovieClip._lockroot
_name	Button._name, MovieClip._name, TextField._name
_parent	_parent, Button._parent, MovieClip._parent, TextField._parent
_quality	_quality, Button._quality, TextField._quality
_root	_root
_rotation	Button._rotation, MovieClip._rotation, TextField._rotation
_soundbuftime	_soundbuftime, Button._soundbuftime, MovieClip._soundbuftime, TextField._soundbuftime
_target	Button._target, MovieClip._target, TextField._target
_totalframes	MovieClip._totalframes
_url	Button._url, MovieClip._url, TextField._url
_visible	Button._visible, MovieClip._visible, TextField._visible
_width	Button._width, MovieClip._width, TextField._width
_x	Button._x, MovieClip._x, TextField._x
_xmouse	Button._xmouse, MovieClip._xmouse, TextField._xmouse
_xscale	Button._xscale, MovieClip._xscale, TextField._xscale
_y	Button._y, MovieClip._y, TextField._y
_ymouse	Button._ymouse, MovieClip._ymouse, TextField._ymouse
_yscale	Button._yscale, MovieClip._yscale, TextField._yscale
abs	Math.abs()
Accessibility	Classe Accessibility
acos	Math.acos
activityLevel	Camera.activityLevel, Microphone.activityLevel
add	add
addListener	Key.addListener(), Mouse.addListener(), MovieClipLoader.addListener(), Selection.addListener, Stage.addListener, TextField.addListener()
addPage	PrintJob.addPage()
addProperty	Object.addProperty()

Elément ActionScript	Consultez l'entrée
addRequestHeader	LoadVars.addRequestHeader(), XML.addRequestHeader()
alignement	Stage.align, TextFormat.align
allowDomain	LocalConnection.allowDomain, System.security.allowDomain()
allowInsecureDomain	LocalConnection.allowInsecureDomain, System.security.allowInsecureDomain()
and	and
appendChild	XML.appendChild()
apply	Function.apply()
Arguments	Classe Arguments
Array	Classe Array, Array()
asfunction	asfunction
asin	Math.asin()
atan	Math.atan()
atan2	Math.atan2()
attachAudio	MovieClip.attachAudio()
attachMovie	MovieClip.attachMovie()
attachSound	Sound.attachSound()
attachVideo	Video.attachVideo()
attributs	XML.attributes
autosize	TextField.autoSize
avHardwareDisable	System.capabilities.avHardwareDisable
background	TextField.background
backgroundColor	TextField.backgroundColor
BACKSPACE	Key.BACKSPACE
bandwidth	Camera.bandwidth
beginFill	MovieClip.beginFill()
beginGradientFill	MovieClip.beginGradientFill()
blockIndent	TextFormat.blockIndent
bold	TextFormat.bold
Boolean	Boolean(), Classe Boolean
border	TextField.border
borderColor	TextField.borderColor
bottomScroll	TextField.bottomScroll
break	break

Elément ActionScript	Consultez l'entrée
bufferLength	NetStream.bufferLength
bufferTime	NetStream.bufferTime
builtInItems	ContextMenu.builtInItems
bullet	TextFormat.bullet
Button	Classe Button
bytesLoaded	NetStream.bytesLoaded
bytesTotal	NetStream.bytesTotal
call	call(), Function.call
callee	arguments.callee
caller	arguments.caller
Camera	Classe Camera
capabilities	Objet System.capabilities
CAPSLOCK	Key.CAPSLOCK
caption	ContextMenuItem.caption
case	case
catch	try..catch..finally
ceil	Math.ceil()
charAt	String.charAt
charCodeAt	String.charCodeAt
childNodes	XML.childNodes
chr	chr
class	class
clear	MovieClip.clear(), SharedObject.clear(), Video.clear()
clearInterval	clearInterval()
cloneNode	XML.cloneNode()
close	LocalConnection.close(), NetStream.close(), XMLSocket.close
Color	Classe Color, TextFormat.color
concat	Array.concat, String.concat()
connect	LocalConnection.connect(), NetConnection.connect(), XMLSocket.connect
condenseWhite	TextField.condenseWhite

Elément ActionScript	Consultez l'entrée
constructeur	Classe Array, Classe Boolean, Classe Camera, Classe Color, Classe ContextMenu, Classe ContextMenuItem, Classe Date, Classe Error, Classe LoadVars, Classe LocalConnection, Classe Microphone, Classe NetConnection, Classe NetStream, Classe Number, Classe Object, Classe PrintJob, Classe SharedObject, Classe Sound, Classe String, Classe TextField.StyleSheet, Classe TextFormat, Classe XML, Classe XMLSocket
contentType	LoadVars.contentType, XML.contentType
ContextMenu	Classe ContextMenu
ContextMenuItem	Classe ContextMenuItem
continue	continue
CONTROL	Key.CONTROL
copy	ContextMenu.copy(), ContextMenuItem.copy()
cos	Math.cos
createElement	XML.createElement
createEmptyMovieClip	MovieClip.createEmptyMovieClip()
createTextField	MovieClip.createTextField()
createTextNode	XML.createTextNode
currentFps	Camera.currentFps, NetStream.currentFps
curveTo	MovieClip.curveTo()
CustomActions	Classe CustomActions
customItems	ContextMenu.customItems
data	SharedObject.data
Date	Classe Date
deblocking	Video.deblocking
default	default
delete	delete
DELETEKEY	Key.DELETEKEY
do while	do while
docTypeDecl	XML.docTypeDecl
domain	LocalConnection.domain()
DOWN	Key.DOWN
duplicateMovieClip	duplicateMovieClip(), MovieClip.duplicateMovieClip
duration	Sound.duration
dynamic	dynamic
E	Math.E
else	else

Elément ActionScript	Consultez l'entrée
else if	else if
embedFonts	TextField.embedFonts
enabled	Button.enabled, ContextMenuItem.enabled, MovieClip.enabled
END	Key.END
endFill	MovieClip.endFill()
ENTER	Key.ENTER
eq	eq (égal à – spécifique à la chaîne)
Error	Classe Error
ESCAPE (constante)	Key.ESCAPE
escape (fonction)	escape
eval	eval()
exactSettings	System.exactSettings
exp	Math.exp
extends	extends
false	false
finally	try..catch..finally
findText	TextSnapshot.findText()
firstChild	XML.firstChild
floor	Math.floor
flush	SharedObject.flush()
focusEnabled	MovieClip.focusEnabled
font	TextFormat.font
for	for
for..in	for..in
fps	Camera.fps
fromCharCode	String.fromCharCode()
fscommand	fscommand()
function	function, Classe Function
gain	Microphone.gain
ge	ge (supérieur ou égal à - spécifique aux chaînes)
get	Camera.get(), CustomActions.get(), get, Microphone.get()
getAscii	Key.getAscii()
getBeginIndex	Selection.getBeginIndex()
getBounds	MovieClip.getBounds

Elément ActionScript	Consultez l'entrée
getBytesLoaded	LoadVars.getBytesLoaded(), MovieClip.getBytesLoaded(), Sound.getBytesLoaded(), XML.getBytesLoaded()
getBytesTotal	LoadVars.getBytesTotal(), MovieClip.getBytesTotal(), Sound.getBytesTotal(), XML.getBytesTotal()
getCaretIndex	Selection.getCaretIndex()
getCode	Key.getCode()
getCount	TextSnapshot.getCount()
getDate	Date.getDate
getDay	Date.getDay()
getDepth	Button.getDepth, MovieClip.getDepth(), TextField.getDepth()
getEndIndex	Selection.getEndIndex()
getFocus	Selection.getFocus()
getFontList	TextField.getFontList
getFullYear	Date.getFullYear
getHours	Date.getHours()
getInstanceAtDepth	MovieClip.getInstanceAtDepth()
getLocal	SharedObject.getLocal()
getMilliseconds	Date.getMilliseconds()
getMinutes	Date.getMinutes()
getMonth	Date.getMonth
getNewTextFormat	TextField.getNewTextFormat()
getNextHighestDepth	MovieClip.getNextHighestDepth()
getPan	Sound.getPan()
getProgress	MovieClipLoader.getProgress()
getProperty	getProperty
getRGB	Color.getRGB()
getSeconds	Date.getSeconds
getSelected	TextSnapshot.getSelected()
getSelectedText	TextSnapshot.getSelectedText()
getSize	SharedObject.getSize()
getStyle	TextField.StyleSheet.getStyle()
getStyleNames	TextField.StyleSheet.getStyleNames()
getSWFVersion	MovieClip.getSWFVersion()
getText	TextSnapshot.getText()
getTextExtent	TextFormat.getTextExtent()

Elément ActionScript	Consultez l'entrée
getTextFormat	TextField.getTextFormat()
getTextSnapshot	MovieClip.getTextSnapshot()
getTime	Date.getTime()
getTimer	getTimer
getTimezoneOffset	Date.getTimezoneOffset()
getTransform	Color.getTransform(), Sound.getTransform()
getURL	getURL(), MovieClip.getURL()
getUTCDate	Date.getUTCDate()
getUTCDay	Date.getUTCDay()
getUTCFullYear	Date.getUTCFullYear()
getUTCHours	Date.getUTCHours()
getUTCMilliseconds	Date.getUTCMilliseconds()
getUTCMinutes	Date.getUTCMinutes()
getUTCMonth	Date.getUTCMonth()
getUTCSeconds	Date.getUTCSeconds()
getVersion	getVersion
getVolume	Sound.getVolume()
getYear	Date.getYear()
globalToLocal	MovieClip.globalToLocal()
goto	gotoAndPlay, gotoAndStop()
gotoAndPlay	gotoAndPlay, MovieClip.gotoAndPlay()
gotoAndStop	gotoAndStop(), MovieClip.gotoAndStop
gt	gt (supérieur à – spécifique aux chaînes)
hasAccessibility	System.capabilities.hasAccessibility
hasAudio	System.capabilities.hasAudio
hasAudioEncoder	System.capabilities.hasAudioEncoder
hasChildNodes	XML.hasChildNodes()
hasEmbeddedVideo	System.capabilities.hasEmbeddedVideo
hasMP3	System.capabilities.hasMP3
hasPrinting	System.capabilities.hasPrinting
hasScreenBroadcast	System.capabilities.hasScreenBroadcast
hasScreenPlayback	System.capabilities.hasScreenPlayback
hasStreamingAudio	System.capabilities.hasStreamingAudio
hasStreamingVideo	System.capabilities.hasStreamingVideo
hasVideoEncoder	System.capabilities.hasVideoEncoder

Elément ActionScript	Consultez l'entrée
height	Camera.height, Stage.height, Video.height
hide	Mouse.hide()
hideBuiltInItems	ContextMenu.hideBuiltInItems()
hitArea	MovieClip.hitArea
hitTest	MovieClip.hitTest()
hitTestTextNearPos	TextSnapshot.hitTestTextNearPos()
HOME	Key.HOME
hscroll	TextField.hscroll
html	TextField.html
htmlText	TextField.htmlText
ID3	Sound.ID3
if	if
ifFrameLoaded	ifFrameLoaded
ignoreWhite	XML.ignoreWhite
implements	implements
import	import
indentation	TextFormat.indent
index	Camera.index, Microphone.index
indexOf	String.indexOf
Infinity	Infinity
-Infinity	-Infinity
INSERT	Key.INSERT
insertBefore	XML.insertBefore
install	CustomActions.install()
instanceof	instanceof
int	int
interface	interface
isActive	Accessibility.isActive()
isDebugger	System.capabilities.isDebugger
isDown	Key.isDown()
isFinite	isFinite
isNaN	isNaN()
isToggled	Key.isToggled()
italic	TextFormat.italic
join	Array.join()

Elément ActionScript	Consultez l'entrée
Key	Classe Key
Language	System.capabilities.language
lastChild	XML.lastChild
lastIndexOf	String.lastIndexOf
le	le (inférieur ou égal à – spécifique aux chaînes)
leading	TextFormat.leading
LEFT	Key.LEFT
leftMargin	TextFormat.leftMargin
length	length, arguments.length, Array.length, String.length, TextField.length
level	_level
lineStyle	MovieClip.lineStyle()
lineTo	MovieClip.lineTo
list	CustomActions.list()
LN10	Math.LN10
LN2	Math.LN2
load	LoadVars.load(), TextField.StyleSheet.load(), XML.load(),
loadClip	MovieClipLoader.loadClip()
loaded	LoadVars.loaded, XML.loaded
loadMovie	loadMovie(), MovieClip.loadMovie
loadMovieNum	loadMovieNum()
loadSound	Sound.loadSound
loadVariables	loadVariables(), MovieClip.loadVariables()
loadVariablesNum	loadvariablesNum()
LoadVars	Classe LoadVars
LocalConnection	Classe LocalConnection
localFileReadDisable	System.capabilities.localFileReadDisable
localToGlobal	MovieClip.localToGlobal()
log	Math.log()
LOG10E	Math.LOG10E
LOG2E	Math.LOG2E
lt	lt (inférieur à – spécifique aux chaînes)
manufacturer	System.capabilities.manufacturer
Math	Classe Math
max	Math.max()

Elément ActionScript	Consultez l'entrée
MAX_VALUE	Number.MAX_VALUE
maxChars	TextField.maxChars
maxhscroll	TextField.maxhscroll
maxscroll	maxscroll, TextField.maxscroll
mbchr	mbchr
mblength	mblength
mbord	mbord
mbsubstring	mbsubstring
menu	Button.menu, MovieClip.menu, TextField.menu
message	Error.message
Microphone	Classe Microphone
min	Math.min()
MIN_VALUE	Number.MIN_VALUE
MMExecute	MMExecute()
motionLevel	Camera.motionLevel
motionTimeOut	Camera.motionTimeOut
Mouse	Classe Mouse
mouseWheelEnabled	TextField.mouseWheelEnabled
moveTo	MovieClip.moveTo
MovieClip	Classe MovieClip
MovieClipLoader	Classe MovieClipLoader
multiline	TextField.multiline
muted	Camera.muted, Microphone.muted
name	Error.name, Microphone.name
names	Camera.names, Microphone.names
NaN	NaN, Number.NaN
ne	ne (pas égal à – spécifique aux chaînes)
NEGATIVE_INFINITY	Number.NEGATIVE_INFINITY
NetConnection	Classe NetConnection
NetStream	Classe NetStream
new (opérateur)	new
newline	newline
nextFrame	nextFrame(), MovieClip.nextFrame()
nextScene	nextScene()

Elément ActionScript	Consultez l'entrée
nextSibling	XML.nextSibling
nodeName	XML.nodeName
nodeType	XML.nodeType
nodeValue	XML.nodeValue
not	not
null	null
Number	Number, Classe Number
Object	Classe Object, Objet()
on	on()
onActivity	Camera.onActivity, Microphone.onActivity
onChanged	TextField.onChanged
onClipEvent	onClipEvent()
onClose	XMLSocket.onClose
onConnect	XMLSocket.onConnect
onData	LoadVars.onData, MovieClip.onData, XML.onData, XMLSocketXML.onData()
onDragOut	Button.onDragOut, MovieClip.onDragOut
onDragOver	Button.onDragOver, MovieClip.onDragOver
onEnterFrame	MovieClip.onEnterFrame
onID3	Sound.onID3
onKeyDown	Button.onKeyDown, Key.onKeyDown, MovieClip.onKeyDown
onKeyUp	Button.onKeyUp, Key.onKeyUp, MovieClip.onKeyUp
onKillFocus	Button.onKillFocus, MovieClip.onKillFocus, TextField.onKillFocus
onLoad	LoadVars.onLoad, MovieClip.onLoad, Sound.onLoad, TextField.StyleSheet.onLoad, XML.onLoad
onLoadComplete	MovieClipLoader.onLoadComplete()
onLoadError	MovieClipLoader.onLoadError()
onLoadInit	MovieClipLoader.onLoadInit()
onLoadProgress	MovieClipLoader.onLoadProgress()
onLoadStart	MovieClipLoader.onLoadStart()
onMouseDown	Mouse.onMouseDown, MovieClip.onMouseDown
onMouseMove	Mouse.onMouseMove, MovieClip.onMouseMove
onMouseUp	Mouse.onMouseUp, MovieClip.onMouseUp
onMouseWheel	Mouse.onMouseWheel
onPress	Button.onPress, MovieClip.onPress

Elément ActionScript	Consultez l'entrée
onRelease	Button.onRelease, MovieClip.onRelease
onReleaseOutisde	Button.onReleaseOutside, MovieClip.onReleaseOutside
onResize	Stage.onResize
onRollOut	Button.onRollOut, MovieClip.onRollOut
onRollOver	Button.onRollOver, MovieClip.onRollOver
onScroller	TextField.onScroller
onSelect	ContextMenu.onSelect, ContextMenuItem.onSelect
onSetFocus	Button.onSetFocus, MovieClip.onSetFocus, Selection.onSetFocus, TextField.onSetFocus
onSoundComplete	Sound.onSoundComplete
onStatus	Camera.onStatus, LocalConnection.onStatus, Microphone.onStatus, NetStream.onStatus, SharedObject.onStatus, System.onStatus
onUnload	MovieClip.onUnload
onUpdate	onUpdate
onXML	XMLSocket.onXML
or (OR logique)	or
ord	ord
os	System.capabilities.os
parentNode	XML.parentNode
parseCSS	TextField.StyleSheet.parseCSS()
parseFloat	parseFloat()
parseInt	parseInt
parseXML	XML.parseXML
password	TextField.password
pause	NetStream.pause()
PGDN	Key.PGDN
PGUP	Key.PGUP
PI	Math.PI
pixelAspectRatio	System.capabilities.pixelAspectRatio
play	play(), MovieClip.play(), NetStream.play()
playerType	System.capabilities.playerType
pop	Array.pop()
position	Sound.position
POSITIVE_INFINITY	Number.POSITIVE_INFINITY
pow	Math.pow()

Elément ActionScript	Consultez l'entrée
prevFrame	prevFrame(), MovieClip.prevFrame()
previousSibling	XML.previousSibling
prevScene	prevScene()
print	print()
printAsBitmap	printAsBitmap()
printAsBitmapNum	printAsBitmapNum
PrintJob	Classe PrintJob
printNum	printNum()
private	private
prototype	Function.prototype
public	public
push	Array.push()
quality	Camera.quality
random	random, Math.random()
rate	Microphone.rate
registerClass	Object.registerClass()
removeListener	Key.removeListener(), Mouse.removeListener(), MovieClipLoader.removeListener(), Selection.removeListener, Stage.removeListener, TextField.removeListener
removeMovieClip	removeMovieClip(), MovieClip.removeMovieClip()
removeNode	XML.removeNode
removeTextField	TextField.removeTextField()
replaceSel	TextField.replaceSel()
replaceText	TextField.replaceText()
resolutionX	System.capabilities.screenResolutionX
resolutionY	System.capabilities.screenResolutionY
restrict	TextField.restrict
return	return
reverse	Array.reverse()
RIGHT	Key.RIGHT
rightMargin	TextFormat.rightMargin
round	Math.round()
scaleMode	Stage.scaleMode
screenColor	System.capabilities.screenColor
screenDPI	System.capabilities.screenDPI

Elément ActionScript	Consultez l'entrée
screenResolutionX	System.capabilities.screenResolutionX
screenResolutionY	System.capabilities.screenResolutionY
scroll	scroll, TextField.scroll
seek	NetStream.seek()
selectable	TextField.selectable
Selection	Classe Selection
send	LoadVars.send(), LocalConnection.send(), PrintJob.send(), XML.send, XMLSocket.send
sendAndLoad	LoadVars.sendAndLoad(), XML.sendAndLoad
separatorBefore	ContextMenuItem.separatorBefore
serverString	System.capabilities.serverString
set	set
set variable	Variable set
setBufferTime	NetStream.setBufferTime()
setClipboard	System.setClipboard()
setDate	Date.setDate()
setFocus	Selection.setFocus
setFullYear	Date.setFullYear()
setGain	Microphone.setGain()
setHours	Date.setHours()
setInterval	setInterval()
setMask	MovieClip.setMask()
setMilliseconds	Date.setMilliseconds()
setMinutes	Date.setMinutes()
setMode	Camera.setMode()
setMonth	Date.setMonth()
setMotionLevel	Camera.setMotionLevel()
setNewTextFormat	TextField.setNewTextFormat
setPan	Sound.setPan
setProperty	setProperty()
setQuality	Camera.setQuality()
setRate	Microphone.setRate()
setRGB	Color.setRGB()
setSeconds	Date.setSeconds()
setSelectColor	TextSnapshot.setSelectColor()

Elément ActionScript	Consultez l'entrée
setSelected	TextSnapshot.setSelected()
setSelection	Selection.setSelection()
setSilenceLevel	Microphone.setSilenceLevel()
setStyle	TextField.StyleSheet.setStyle()
setTextFormat	TextField.setTextFormat
setTime	Date.setTime()
setTransform	Color.setTransform(), Sound.setTransform
setUseEchoSuppression	Microphone.setUseEchoSuppression()
setUTCDate	Date.setUTCDate()
setUTCFullYear	Date.setUTCFullYear()
setUTCHours	Date.setUTCHours()
setUTCMilliseconds	Date.setUTCMilliseconds()
setUTCMinutes	Date.setUTCMinutes()
setUTCMonth	Date.setUTCMonth()
setUTCSeconds	Date.setUTCSeconds()
setVolume	Sound.setVolume
setYear	Date.setYear()
SharedObject	Classe SharedObject
SHIFT (constante)	Key.SHIFT
shift (méthode)	Array.shift
show	Mouse.show()
showMenu	Stage.showMenu
showSettings	System.showSettings()
silenceLevel	Microphone.silenceLevel()
silenceTimeout	Microphone.silenceTimeout()
sin	Math.sin()
size	TextFormat.size
slice	Array.slice, String.slice
smoothing	Video.smoothing
sort	Array.sort
sortOn	Array.sortOn()
Sound	Classe Sound
SPACE	Key.SPACE
splice	Array.splice()

Elément ActionScript	Consultez l'entrée
split	String.split
sqrt	Math.sqrt()
SQRT1_2	Math.SQRT1_2
SQRT2	Math.SQRT2
Stage	Classe Stage
start	PrintJob.start(), Sound.start()
startDrag	startDrag(), MovieClip.startDrag()
static	static
status	XML.status
stop	stop(), MovieClip.stop(), Sound.stop
stopAllSounds	stopAllSounds()
stopDrag	stopDrag(), MovieClip.stopDrag()
String	Classe String, String
StyleSheet (classe)	Classe TextField.StyleSheet
styleSheet (propriété)	TextField.styleSheet
substr	String.substr
substring	substring, String.substring
super	super
swapDepths	MovieClip.swapDepths()
switch	switch
System	Classe System
Tab	Key.TAB
tabChildren	MovieClip.tabChildren
tabEnabled	Button.tabEnabled, mon_mc.tabEnabled, TextField.tabEnabled
tabIndex	Button.tabIndex, MovieClip.tabIndex, TextField.tabIndex
tabStops	TextFormat.tabStops
tan	Math.tan()
target	TextFormat.target
targetPath	targetPath
tellTarget	tellTarget
text	TextField.text
textColor	TextField.textColor
TextField	Classe TextField
TextFormat	Classe TextFormat

Elément ActionScript	Consultez l'entrée
textHeight	TextField.textHeight
TextSnapshot	Objet TextSnapshot
textWidth	TextField.textWidth
this	this
throw	throw
time	NetStream.time
toggleHighQuality	toggleHighQuality()
toLowerCase	String.toLowerCase
toString	Array.toString(), Boolean.toString, Date.toString(), Error.toString(), LoadVars.toString(), Number.toString(), Object.toString(), XML.toString
toUpperCase	String.toUpperCase
trace	trace()
trackAsMenu	Button.trackAsMenu, MovieClip.trackAsMenu
true	true
try	try..catch..finally
type	TextField.type
typeof	typeof
undefined	undefined
underline	TextFormat.underline
unescape	unescape
uninstall	CustomActions.uninstall
unloadClip	MovieClipLoader.unloadClip()
unloadMovie	unloadMovie(), MovieClip.unloadMovie()
unLoadMovieNum	unloadMovieNum()
unshift	Array.unshift()
unwatch	Object.unwatch()
UP	Key.UP
updateAfterEvent	updateAfterEvent()
updateProperties	Accessibility.updateProperties()
url	TextFormat.url
useCodePage	System.useCodepage
useEchoSuppression	Microphone.useEchoSuppression()
useHandCursor	Button.useHandCursor, MovieClip.useHandCursor
UTC	Date.UTC()

Elément ActionScript	Consultez l'entrée
valueOf	Boolean.valueOf(), Number.valueOf, Object.valueOf()
var	var
variable	TextField.variable
version	System.capabilities.version
Video	Classe Video
visible	ContextMenuItem.visible
void	void
watch	Object.watch()
while	while
width	Camera.width, Stage.width, Video.width
with	with
wordwrap	TextField.wordWrap
XML	Classe XML
xmlDecl	XML.xmlDecl
XMLNode	Classe XMLNode
XMLSocket	Classe XMLSocket

-- (décrémentation)

Disponibilité

Flash Player 4.

Usage

```
--expression
expression--
```

Paramètres

Aucun.

Renvoie

Nombre.

Description

Opérateur (arithmétique) : opérateur unaire de pré et post-décrémentation qui soustrait 1 de l'*expression*. La forme de pré-décrémentation de l'opérateur (`--expression`) soustrait 1 de *l'expression* et renvoie le résultat. La forme de post-décrémentation de l'opérateur (`expression--`) soustrait 1 de l'*expression* et renvoie la valeur initiale de l'*expression* (le résultat avant la soustraction).

Exemple

La forme de pré-décrémentation de l'opérateur décrémente `x` de 2 (`x - 1 = 2`) et renvoie le résultat comme `y` :

```
x = 3;
y = --x;
// y est égal à 2
```

La forme de post-décrémentation de l'opérateur décrémente `x` de 2 (`x - 1 = 2`) et renvoie la valeur originale de `x` comme résultat `y` :

```
x = 3;
y = x--
//y est égal à 3
```

++ (incrémentation)

Disponibilité

Flash Player 4.

Usage

++expression

expression++

Paramètres

Aucun.

Renvoie

Nombre.

Description

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

La forme de pré-incrémentation de l'opérateur incrémente `x` à 2 (`x + 1 = 2`) et renvoie le résultat comme `y` :

```
x = 1;
y = ++x
// y est égal à 2
```

La forme de post-incrémentation de l'opérateur incrémente `x` à 2 (`x + 1 = 2`) et renvoie la valeur originale de `x` comme résultat `y` :

```
x = 1;
y = x++;
//y est égal à 1
```

Exemple

L'exemple suivant utilise ++ comme opérateur de post-incrémentation pour entraîner cinq boucles de `while`.

```
i = 0;
while(i++  5){<
trace("ceci est l'exécution " + i);
}
```

Cet exemple utilise ++ comme opérateur de pré-incrémentation.

```
var a = [];
var i = 0;
while (i < 10) {
  a.push(++i);
}
trace(a.join());
```

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 ++ comme opérateur de post-incrémentation.

```
var a = [];
var i = 0;
while (i < 10) {
a.push(i++);
  }
trace(a.join());
```

Ce script affiche le résultat suivant dans le panneau de sortie :

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

! (NOT logique)

Disponibilité

Flash Player 4.

Usage

`!`*`expression`*

Paramètres

Aucun.

Renvoie

Une valeur booléenne.

Description

Opérateur (logique) : inverse la valeur booléenne d'une variable ou expression. Si *`expression`* est une variable avec la valeur absolue ou convertie `true`, la valeur de `!`*`expression`* est `false`. Si l'expression `x && y` est `false`, l'expression `!(x && y)` est `true`.

Les expressions suivantes illustrent le résultat de l'utilisation de l'opérateur ! :

`!`*`true`* renvoie `false`

`!`*`false`* renvoie `true`

Exemple

Dans l'exemple suivant, la variable `heureux` est définie sur `false`. La condition `if` évalue la condition `!heureux` et, si la condition est `true`, l'action `trace()` envoie une chaîne au panneau de sortie.

```
heureux = false;
if (!heureux) {
  trace("le bonheur, c'est être heureux");
}
```

!= (inégalité)

Disponibilité

Flash Player 5.

Usage

expression1 != *expression2*

Paramètres

Aucun.

Renvoie

Une valeur booléenne.

Description

Opérateur (inégalité) : teste l'opposé exact de l'opérateur ==. Si *expression1* est égale à *expression2*, le résultat est `false`. Tout comme pour l'opérateur ==, la définition du terme *égalité* dépend des types de données comparés.

- Les nombres, les chaînes et les valeurs booléennes sont comparés par valeur.
- Les variables, les objets, les tableaux et les fonctions sont comparés par référence.

Exemple

L'exemple suivant illustre le résultat de l'opérateur `!=` :

`5 != 8` renvoie `true`

`5 != 5` renvoie `false`

Cet exemple illustre l'utilisation de l'opérateur `!=` dans une instruction `if` :

```
a = "David";
b = "Idiot"
if (a != b){
  trace("David n'est pas idiot");
}
```

Consultez également

!== (inégalité stricte), == (égalité), === (égalité stricte)

!== (inégalité stricte)

Disponibilité

Flash Player 6.

Usage

expression1 !== *expression2*

Description

Opérateur : teste l'opposé exact de l'opérateur ===. L'opérateur d'inégalité stricte fonctionne de la même façon que l'opérateur d'inégalité, à ceci près que les types de données ne sont pas convertis. Si *expression1* est égale à *expression2*, et leur type de données est égal, le résultat est `false`. Tout comme pour l'opérateur ===, la définition du terme *égalité* dépend des types de données comparés.

- Les nombres, les chaînes et les valeurs booléennes sont comparés par valeur.
- Les variables, les objets, les tableaux et les fonctions sont comparés par référence.

Exemple

Le code suivant affiche la valeur renvoyée par les opérations utilisant les opérateurs d'égalité, d'égalité stricte et d'inégalité stricte.

```
s1 = new String("5");
s2 = new String("5");
s3 = new String("Bonjour");
n = new Number(5);
b = new Boolean(true);

s1 == s2; // true
s1 == s3; // false
s1 == n; // true
s1 == b; // false

s1 === s2; // true
s1 === s3; // false
s1 === n; // false
s1 === b; // false

s1 !== s2; // false
s1 !== s3; // true
s1 !== n; // true
s1 !== b; // true
```

Consultez également

!= (inégalité), == (égalité), === (égalité stricte)

% (modulo)

Disponibilité

Flash Player 4. Dans les fichiers Flash 4, l'opérateur `%` est développé dans le fichier SWF sous la forme `x - int(x/y) * y` et risque de ne pas être aussi rapide ou précis dans les versions suivantes de Flash Player.

Usage

```
expression1 % expression2
```

Paramètres

Aucun.

Renvoie

Rien.

Description

Opérateur (arithmétique) : calcule le reste de *expression1* divisé par *expression2*. Si l'un des paramètres de *expression* n'est pas un nombre, l'opérateur de modulo tente de les convertir en nombre. *expression* peut être un nombre ou une chaîne convertie en valeur numérique.

Exemple

L'exemple suivant illustre l'emploi de l'opérateur modulo (%).

```
trace (12 % 5);
// renvoie 2
trace (4.3 % 2.1);
// renvoie approximativement 0.1
```

%= (affectation de modulo)

Disponibilité

Flash Player 4.

Usage

expression1 %= expression2

Paramètres

Aucun.

Renvoie

Rien.

Description

Opérateur (affectation composée arithmétique) : affecte à *expression1* la valeur de *expression1 % expression2*. Par exemple, les deux expressions suivantes sont équivalentes :

```
x %= y
x = x % y
```

Exemple

L'exemple suivant affecte la valeur `4` à la variable `x`.

```
x = 14;
y = 5;
trace(x %= y);
// renvoie 4
```

Consultez également

% (modulo)

& (Opérateur AND au niveau du bit)

Disponibilité

Flash Player 5. Dans Flash 4, l'opérateur `&` était utilisé pour concaténer les chaînes. Dans Flash 5 et les versions suivantes, l'opérateur `&` est un AND au niveau du bit ; vous devez donc utiliser les opérateurs `add` et `+` pour concaténer les chaînes. Les fichiers Flash 4 utilisant l'opérateur `&` sont automatiquement mis à jour pour utiliser `add` lorsqu'ils sont importés dans l'environnement auteur de Flash 5 ou d'une version ultérieure.

Usage

```
expression1 & expression2
```

Paramètres

Aucun.

Renvoie

Rien.

Description

Opérateur (au niveau du bit) : convertit *`expression1`* et *`expression2`* en entiers 32 bits non signés et effectue une opération AND booléenne sur chaque bit des paramètres entiers. Le résultat est un nouvel entier 32 bits non signé.

&& (AND logique)

Disponibilité

Flash Player 4.

Usage

expression1 && *expression2*

Paramètres

Aucun.

Renvoie

Une valeur booléenne.

Description

Opérateur (logique) : effectue une opération booléenne sur les valeurs d'une ou des deux expressions. Evalue *expression1* (l'expression du côté gauche de l'opérateur) et renvoie `false` si l'expression est `false`. Si *expression1* est `true`, *expression2* (l'expression du côté droit de l'opérateur) est évaluée. Si *expression2* est `true`, le résultat final est `true` ; sinon, il est `false`.

Exemple

Cet exemple utilise l'opérateur `&&` pour effectuer un test permettant de déterminer si un joueur a gagné la partie. Les variables `tours` et `score` sont mises à jour lorsqu'un joueur prend son tour ou marque des points pendant la partie. Le script suivant affiche « Vous avez gagné ! » dans le panneau de sortie lorsque le score du joueur atteint 75 ou plus en 3 tours ou moins.

```
tours =2;
score=77;
gagnant = (tours <= 3) && (score >= 75);
if (gagnant) {
  trace("Vous avez gagné !");
} else {
  trace("Réessayez !");
}
```

&= (affectation AND au niveau du bit)

Disponibilité

Flash Player 5.

Usage

expression1 &= *expression2*

Paramètres

Aucun.

Renvoie

Rien.

Description

Opérateur : affecte à *expression1* la valeur de *expression1* & *expression2*. Par exemple, les deux expressions suivantes sont équivalentes.

```
x &= y;
x = x & y;
```

Exemple

L'exemple suivant affecte la valeur 9 à x.

```
x = 15;
y = 9;
trace(x &= y);
// renvoie 9
```

Consultez également

& (Opérateur AND au niveau du bit)

() (parenthèses)

Disponibilité

Flash Player 4.

Usage

(*expression1, expression2*)

fonction(*paramètre1,..., paramètreN*)

Paramètres

expression1, expression2 Nombres, chaînes, variables ou texte.

fonction La fonction devant être effectuée sur le contenu des parenthèses.

paramètre1...paramètreN Une série de paramètres à exécuter avant que les résultats ne soient transmis comme paramètres à la fonction extérieure aux parenthèses.

Renvoie

Rien.

Description

Opérateur : effectue une opération de regroupement sur un ou plusieurs paramètres ou entoure un ou plusieurs paramètres et les transmet comme paramètres à une fonction extérieure aux parenthèses.

Usage 1 : contrôle l'ordre dans lequel les opérateurs sont exécutés dans l'expression. Les parenthèses annulent l'ordre normal de priorité et obligent l'évaluation des expressions entre parenthèses en premier. Lorsque les parenthèses sont imbriquées, le contenu des parenthèses internes est évalué avant le contenu des parenthèses externes.

Usage 2 : encadre un ou plusieurs paramètres et les transmet sous forme de paramètres à la fonction extérieure aux parenthèses.

Exemple

Usage 1 : les instructions suivantes illustrent l'emploi des parenthèses pour contrôler l'ordre d'exécution des expressions. La valeur de chaque expression est affichée en dessous de chaque ligne, comme suit :

```
trace((2 + 3) * (4 + 5));
// affiche 45
trace(2 + (3 * (4 + 5)));
// affiche 29
trace(2 + (3 * 4) + 5);
// affiche 19
```

Usage 2 : l'exemple suivant illustre l'emploi des parenthèses avec des fonctions.

```
getDate();
facture(article, montant);
function traceParameter(param){
  trace(param);
}
traceParameter(2*2);
```

Consultez également

`with`

- (moins)

Disponibilité

Flash Player 4.

Usage

(Négation) `-expression`

(Soustraction) *`expression1`* `-` *`expression2`*

Paramètres

Aucun.

Renvoie

Rien.

Description

Opérateur (arithmétique) : utilisé pour la négation ou la soustraction.

Usage 1 : lorsque utilisé pour la négation, il inverse le signe de l'*expression* numérique.

Usage 2 : lorsque utilisé pour la soustraction, il effectue une soustraction arithmétique sur deux expressions numériques, soustrayant *expression2* de *expression1*. Lorsque les deux expressions sont des entiers, la différence est un entier. Lorsque l'une des expressions, ou les deux, est un nombre à virgule flottante, la différence est un nombre à virgule flottante.

Exemple

Usage 1 : l'instruction suivante inverse le signe de l'expression 2 + 3.

```
-(2 + 3)
```

Le résultat est `-5`.

Usage 2 : l'instruction suivante soustrait l'entier 2 de l'entier 5.

```
5 - 2
```

Le résultat est `3`, qui est un entier.

Usage 2 : l'instruction suivante soustrait le nombre à virgule flottante 1.5 du nombre à virgule flottante 3.25.

```
3.25 - 1.5
```

Le résultat est `1.75`, qui est un nombre à virgule flottante.

* (multiplication)

Disponibilité

Flash Player 4.

Usage

```
expression1 * expression2
```

Paramètres

Aucun.

Renvoie

Rien.

Description

Opérateur (arithmétique) : multiplie deux expressions numériques. Si les deux expressions sont des entiers, le produit est un entier. Si l'une des expressions, ou les deux, est un nombre à virgule flottante, le produit est un nombre à virgule flottante.

Exemple

Usage 1 : l'instruction suivante multiplie les entiers 2 et 3.

```
2 * 3
```

Le résultat est 6, qui est un entier.

Usage 2 : cette instruction multiplie les nombres à virgule flottante 2.0 et 3.1416.

```
2.0 * 3.1416
```

Le résultat est 6.2832, qui est un nombre à virgule flottante.

*= (affectation de multiplication)

Disponibilité

Flash Player 4.

Usage

*expression1 *= expression2*

Paramètres

Aucun.

Renvoie

Rien.

Description

Opérateur (affectation composée arithmétique) : affecte à *expression1* la valeur de *expression1 * expression2*. Par exemple, les deux expressions suivantes sont équivalentes :

```
x *= y
x = x * y
```

Exemple

Usage 1 : l'exemple suivant affecte la valeur 50 à la variable x.

```
x = 5;
y = 10;
trace(x *= y);
// renvoie 50
```

Usage 2 : les deuxième et troisième lignes de l'exemple suivant calculent les expressions se trouvant à droite du signe égal et affectent les résultats à x et y.

```
i = 5;
x = 4 - 6;
y = i + 2;
trace(x *= y);
// renvoie -14
```

Consultez également

* (multiplication)

, (virgule)

Disponibilité

Flash Player 4.

Usage

expression1, *expression2*

Paramètres

Aucun.

Renvoie

Rien.

Description

Opérateur : évalue *expression1*, puis *expression2*, et renvoie la valeur de *expression2*. Cet opérateur est principalement utilisé avec l'instruction de boucle `for`.

Exemple

Le code suivant utilise l'opérateur virgule :

```
var a=1, b=2, c=3;
```

Ceci est équivalent à la rédaction du code suivant :

```
var a=1;
var b=2;
var c=3;
```

.(point)

Disponibilité

Flash Player 4.

Usage

objet.propriété_ou_méthode

nomDoccurrence.variable

nomDoccurrence.occurrenceEnfant.variable

Paramètres

objet Une occurrence de la classe. L'objet peut être une occurrence de n'importe quelle classe intégrée ActionScript ou une classe personnalisée. Ce paramètre se situe toujours à gauche de l'opérateur point (.).

propriété_ou_méthode Le nom d'une propriété ou méthode associée à un objet. Toutes les méthodes et propriétés valides pour les classes intégrées sont répertoriées dans les récapitulatifs des propriétés et méthodes correspondants. Ce paramètre se situe toujours à droite de l'opérateur point (.).

nomDoccurrence Le nom d'occurrence d'un clip.

occurrenceEnfant Une occurrence de clip enfant, ou imbriquée dans un autre clip.

variable Une variable dans le scénario du clip dont le nom d'occurrence figure à gauche de l'opérateur point (.).

Renvoie

Rien.

Description

Opérateur : utilisé pour naviguer dans les hiérarchies de clip afin d'accéder aux clips (enfants) imbriqués, variables ou propriétés. L'opérateur point est également utilisé pour tester ou définir les propriétés d'un objet, exécuter une méthode d'un objet ou créer une structure de données.

Exemple

L'instruction suivante identifie la valeur courante de la variable `couleurCheveux` du clip `person_mc`.

```
person_mc.couleurCheveux
```

Ceci est équivalent à la syntaxe de Flash 4 suivante :

```
/person_mc:couleurCheveux
```

: (type)

Disponibilité

Flash Player 6.

Usage

```
[modificateurs] [var] nomDeVariable:[type]
function nomDeFonction,():[type] { ... }
function nomDeFonction(paramètre1[:type], ... , paramètreN[:type]) { ... }
```

Paramètres

nomDeVariable Un identifiant pour une variable.

type Un type de données natif, un nom de classe qui vous avez défini ou un nom d'interface.

nomDeFonction Un identifiant pour une fonction.

paramètre Un identifiant pour un paramètre de fonction.

Description

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 ; quand il est utilisé dans une déclaration ou une définition de fonction, il spécifie le type de renvoi de la fonction ; quand il est utilisé avec un paramètre de fonction dans une définition de fonction, il spécifie le type de variable attendu pour ce paramètre.

Les types ont une fonctionnalité de compilation seule. Tous les types sont vérifiés au moment de la compilation ; des erreurs sont générées en cas d'incompatibilité. Pour plus d'informations, consultez l'Annexe A, *Messages d'erreur*, page 893. Des incompatibilités peuvent se produire durant les opérations d'affectation, les appels de fonction et les déréférencements des membres de classe à l'aide de l'opérateur point (.) Afin d'éviter les erreurs d'incompatibilité de type, utilisez les types explicites (consultez *Typage strict des données*, page 40).

Les types que vous pouvez utiliser incluent tous les types d'objet natif, les classes et les interfaces que vous définissez, ainsi que Void et Function (existant en tant que type seulement et non en tant qu'objet). Les types natifs reconnus sont Array, Boolean, Button, Color, CustomActions, Date, Function, LoadVars, LocalConnection, Microphone, MovieClip, NetConnection, NetStream, Number, Object, SharedObject, Sound, String, TextField, TextFormat, Video, Void, XML, XMLNode et XMLSocket.

Exemple

Usage 1 : l'exemple suivant déclare une variable publique nommée `nomDutilisateur`, dont le type est `String` ; il lui affecte une chaîne vide.

```
public var nomDutilisateur:String = "";
```

Usage 2 : cet exemple montre comment spécifier un type de paramètre de fonction. Le code suivant définit une fonction nommée `déterminerDate()`, qui prend un paramètre nommé `dateDuJour` de type `Date`.

```
function déterminerDate(dateDuJour:Date) {
   this.date = dateDuJour;
}
```

Usage 3 : le code suivant définit une fonction nommée `racineCarrée()` prenant un paramètre nommé `val` du type `Number` et renvoie la racine carrée de `val`, ainsi qu'un type `Number`.

```
function racineCarrée(val:Number):Number {
  return Math.sqrt(val);
}
```

?: (conditionnel)

Disponibilité

Flash Player 4.

Usage

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

Paramètres

`expression1` Une expression évaluée comme valeur booléenne, généralement une expression de comparaison telle que `x < 5`.

`expression2`, *`expression3`* Valeurs de n'importe quel type.

Renvoie

Rien.

Description

Opérateur : indique à Flash d'évaluer *`expression1`* et, si la valeur de *`expression1`* est `true`, renvoie la valeur de *`expression2`* ; sinon, renvoie la valeur de *`expression3`*.

Exemple

L'instruction suivante affecte la valeur de la variable `x` à la variable `z` étant donné que `expression1` est `true`

```
x = 5;
y = 10;
z = (x < 6) ? x: y;
trace (z);
// renvoie 5
```

/ (division)

Disponibilité

Flash Player 4.

Usage

expression1 / expression2

Paramètres

expression Un nombre ou une variable évaluée comme un nombre.

Renvoie

Rien.

Description

Opérateur (arithmétique) : divise *expression1* par *expression2*. Le résultat de la division est un nombre à virgule flottante à double précision.

Exemple

L'instruction suivante divise le nombre à virgule flottante 22,0 par 7,0 et affiche ensuite le résultat dans le panneau de sortie.

```
trace(22.0 / 7.0);
```

Le résultat est `3,1429`, qui est un nombre à virgule flottante.

// (délimiteur de commentaires)

Disponibilité

Flash 1.

Usage

```
// commentaire
```

Paramètres

commentaire Tout caractère.

Renvoie

Rien.

Description

Commentaire : indique le début d'un commentaire de script. Tout caractère qui apparaît entre le délimiteur de commentaires `//` et le caractère de fin de ligne est interprété comme un commentaire et ignoré par l'interprète d'ActionScript.

Exemple

Ce script utilise des délimiteurs de commentaires pour identifier les première, troisième, cinquième et septième lignes comme commentaires.

```
// enregistrer la position X du clip de la balle
balleX = balle._x;
// enregistrer la position Y du clip de la balle
balleY = balle._y;
// enregistrer la position X du clip de la batte
batteX = batte._x;
// enregistrer la position Y du clip de la batte
batteY = batte._y;
```

Consultez également

`/* (délimiteur de commentaires)`

/* (délimiteur de commentaires)

Disponibilité

Flash Player 5.

Usage

```
/* commentaire */
```

```
/*
commentaire
commentaire
*/
```

Paramètres

commentaire Tout caractère.

Renvoie

Rien.

Description

Commentaire : indique une ou plusieurs lignes de commentaires de script. Tout caractère qui apparaît entre la balise d'ouverture de commentaires `/*` et la balise de fermeture de commentaires `*/` est interprété comme un commentaire et ignoré par l'interprète d'ActionScript. Utilisez le premier type de syntaxe pour identifier les commentaires à une seule ligne. Utilisez le second type de syntaxe pour identifier les commentaires contenant plusieurs lignes successives. L'oubli de la balise de fermeture `*/` lors de l'utilisation de cette forme de délimiteur de commentaires provoque un message d'erreur.

Exemple

Ce script utilise des délimiteurs de commentaires au début du script.

```
/* enregistre les positions X et Y des clips
de la balle et de la batte
*/

balleX = balle._x;
balleY = balle._y;
batteX = batte._x;
batteY = batte._y;
```

Consultez également

`// (délimiteur de commentaires)`

/= (affectation de division)

Disponibilité

Flash Player 4.

Usage

expression1 /= expression2

Paramètres

expression1,expression2 Nombre ou variable évaluée comme nombre.

Renvoie

Rien.

Description

Opérateur (affectation composée arithmétique) : affecte à *expression1* la valeur de *expression1 / expression2*. Par exemple, les deux instructions suivantes sont équivalentes :

```
x /= y
x = x / y
```

Exemple

Le code suivant illustre l'utilisation de l'opérateur /= avec des variables et des nombres.

```
x = 10;
y = 2;
x /= y;
// x contient maintenant la valeur 5
```

[] (accès tableau)

Disponibilité

Flash Player 4.

Usage

```
mon_array = ["a0", a1,...aN];
MultiDimensionnel_array = [["a0",...aN],...["a0",...aN]]
mon_array[E] = valeur
MultiDimensionnel_array[E][E] = valeur
object["valeur"];
```

Paramètres

mon_array Le nom d'un tableau.

a0, a1,...aN Eléments d'un tableau.

MultiDimensionnel_array Le nom d'un tableau multidimensionnel simulé.

E Le nombre (ou index) d'un élément d'un tableau.

object Le nom d'un objet.

valeur A Une chaîne ou une expression évaluée comme chaîne qui nomme une propriété de l'objet.

Renvoie

Rien.

Description

Opérateur : initialise un nouveau tableau ou un tableau multidimensionnel avec les éléments spécifiés (*a0*, etc.) ou accède aux éléments d'un tableau. L'opérateur d'accès tableau vous permet de définir et récupérer dynamiquement les noms d'occurrences, variables et objets. Il vous permet également d'accéder aux propriétés des objets.

Usage 1 : un tableau est un objet dont les propriétés sont appelées *éléments*, qui sont chacune identifiées par un nombre appelé *index*. Lorsque vous créez un tableau, vous encadrez ses éléments par l'opérateur d'accès tableau (ou *crochets*). Un tableau peut contenir des éléments de divers types. Par exemple, le tableau suivant, appelé `personnel`, contient trois éléments ; le premier est un nombre et les deux suivants sont des chaînes (entre guillemets).

```
personnel = [15, "Barbara", "Eric"];
```

Usage 2 : vous pouvez imbriquer des crochets pour simuler des tableaux multidimensionnels. Le code suivant crée un tableau, appelé `ticTacToe`, avec trois éléments ; chaque élément est également un tableau contenant trois éléments.

```
ticTacToe = [[1,2,3],[4,5,6],[7,8,9]];

// choisissez Déboguer > Lister les variables en mode de test d'animation
// pour afficher une liste des éléments du tableau
```

Usage 3 : encadrez l'index de chaque élément avec des crochets pour y accéder directement ; vous pouvez ajouter un nouvel élément à un tableau, changer ou récupérer la valeur d'un élément existant. Le premier élément d'un tableau est toujours 0 :

```
mon_array[0] = 15;
mon_array[1] = "Bonjour";
mon_array[2] = true;
```

Vous pouvez utiliser des crochets pour ajouter un quatrième élément, comme dans l'exemple suivant :

```
mon_array[3] = "George";
```

Usage 4 : vous pouvez utiliser des crochets pour accéder à un élément dans un tableau multidimensionnel. Le premier jeu de crochets identifie l'élément dans le tableau d'origine, le second jeu identifiant l'élément dans le tableau imbriqué. La ligne de code suivante envoie 6 au panneau de sortie.

```
ticTacToe = [[1,2,3],[4,5,6],[7,8,9]];
trace(ticTacToe[1][2]);
// renvoie 6
```

Usage 5 : vous pouvez utiliser l'opérateur d'accès tableau au lieu de la fonction `eval` pour définir et récupérer dynamiquement des valeurs pour des noms de clip ou n'importe quelle propriété d'un objet :

```
nom["mc" + i] = "coin_gauche";
```

Exemple

Usage 1 : les exemples de code suivants indiquent deux manières différentes de créer un objet Array vide (la première ligne utilisant des crochets).

```
mon_array =[];
mon_array = new Array();
```

Usages 1 et 2 : l'exemple suivant crée un tableau appelé `personnel_array` et utilise une action `trace()` pour envoyer les éléments au panneau de sortie. A la quatrième ligne, un élément du tableau est changé, et la cinquième ligne envoie le tableau nouvellement modifié au panneau de sortie :

```
personnel_array = ["Barbara", "George", "Mary"];
trace(personnel_array);
// Barbara, George, Mary
personnel_array[2]="Sam";
trace(personnel_array);
// Barbara, George, Sam
```

Usage 3 : dans l'exemple suivant, l'expression entre crochets (`"morceau" + i`) est évaluée et le résultat est utilisé comme nom de la variable qui doit être récupérée du clip `monClip_mc`. Dans cet exemple, la variable `i` doit exister dans le même scénario que le bouton. Si la variable `i` est égale à 5, par exemple, la valeur de la variable `morceau5` dans le clip `monClip_mc` s'affiche dans le panneau de sortie :

```
on(release) {
  x = monClip_mc["morceau"+i];
  trace(x);
}
```

Usage 3 : dans le code suivant, l'expression entre crochets est évaluée et le résultat est utilisé comme nom de la variable qui doit être récupérée à partir du clip `nom_mc` :

```
nom_mc["A" + i];
```

Si vous êtes habitué(e) à la syntaxe ActionScript à barre oblique de Flash 4, vous pouvez obtenir le même résultat en utilisant la fonction `eval` :

```
eval("nom.A" & i);
```

Usage 3 : vous pouvez également utiliser l'opérateur d'accès tableau sur le côté gauche d'une instruction d'affectation pour définir dynamiquement des noms d'occurrences, de variables et d'objets :

```
nom[index] = "Gary";
```

Consultez également

Classe Array, Classe Object, `eval()`

^ (XOR au niveau du bit)

Disponibilité

Flash Player 5.

Usage

`expression1 ^ expression2`

Paramètres

`expression1,expression2` Un nombre.

Renvoie

Rien.

Description

Opérateur (au niveau du bit) : convertit *`expression1`* et *`expression2`* en entiers 32 bits non signés et renvoie un 1 dans chaque position de bit où les bits correspondants dans *`expression1`* ou *`expression2`*, mais pas les deux, sont 1.

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 décimal = 1111 binaire
// 9 décimal = 1001 binaire
x = 15 ^ 9
trace (x)
// 1111 ^ 1001 = 0110
// renvoie 6 décimal (= 0110 binaire)
```

^= (affectation XOR au niveau du bit)

Disponibilité

Flash Player 5.

Usage

expression1 ^= *expression2*

Paramètres

expression1,*expression2* Entiers et variables.

Renvoie

Rien.

Description

Opérateur (affectation composée au niveau du bit) : affecte à *expression1* la valeur de *expression1* ^ *expression2*. Par exemple, les deux instructions suivantes sont équivalentes :

```
x ^= y
x = x ^ y
```

Exemple

L'exemple suivant illustre une opération ^=.

```
// 15 décimal = 1111 binaire
x = 15;
// 9 décimal = 1001 binaire
y = 9;
trace(x ^= y);
// renvoie 6 décimal ( = 0110 binaire)
```

Consultez également

^ (XOR au niveau du bit)

{} (initialisateur d'objet)

Disponibilité

Flash Player 5.

Usage

```
objet = {nom1: valeur1, nom2: valeur2,...nomN: valeurN}
```

Paramètres

objet L'objet à créer.

nom1,2,...N Les noms des propriétés.

valeur1,2,...N Les valeurs correspondantes pour chaque propriété de *nom*.

Renvoie

Aucun.

Description

Opérateur : crée un objet et l'initialise avec les paires de propriétés *nom* et *valeur* spécifiées. L'utilisation de cet opérateur est identique à l'utilisation de la syntaxe `new Object` et la distribution des paires de propriétés avec l'opérateur d'affectation. Le prototype de l'objet nouvellement créé est nommé de façon générique *Object*.

Exemple

La première ligne du code suivant crée un objet vide à l'aide de l'opérateur initialisateur d'objet, la seconde ligne créant un nouvel objet avec une fonction constructeur.

```
objet = {};
objet = new Object();
```

L'exemple suivant crée un objet `compte` et initialise les propriétés `nom`, `adresse`, `ville`, `pays`, `codePostal` et `solde` avec les valeurs associées.

```
compte = { nom : "Betty Skate",
  adresse : "123 rue Grande",
  ville : "Une ville",
  pays : "France",
  codePostal : "12345",
  solde : "1000" };
```

L'exemple suivant illustre l'imbrication d'initialisateurs d'objet et de tableau les uns dans les autres.

```
personne = { nom : "Gina Vechio",
  enfants : [ "Emilie", "Alice", "Charlotte"] };
```

L'exemple suivant utilise les informations de l'exemple précédent et produit le même résultat avec des fonctions constructeur.

```
personne = new Object();
personne.nom = 'Gina Vechio';
personne.enfants = new Array();
personne.enfants[0] = 'Emilie';
personne.enfants[1] = 'Alice';
personne.enfants[2] = 'Charlotte';
```

Consultez également

[] (accès tableau), new, Classe Object

| (OR au niveau du bit)

Disponibilité

Flash Player 5.

Usage

expression1 | *expression2*

Paramètres

expression1,expression2 Un nombre.

Renvoie

Rien.

Description

Opérateur (au niveau du bit) : convertit *expression1* et *expression2* en entiers 32 bits non signés et renvoie un 1 à chaque position de bit où les bits correspondants de *expression1* ou de *expression2* sont 1.

Exemple

L'exemple suivant illustre une opération OR au niveau du bit.

```
// 15 décimal = 1111 binaire
x = 15;
// 9 décimal = 1001 binaire
y = 9;
trace(x | y);
// 1111 | 0011 = 1111
//renvoie 15 décimal (= 1111 binaire)
```

|| (OR logique)

Disponibilité

Flash Player 4.

Usage

expression1 || *expression2*

Paramètres

expression1,expression2 Une valeur booléenne ou une expression convertie en valeur booléenne.

Renvoie

Une valeur booléenne.

Description

Opérateur (logique) : évalue *expression1* et *expression2*. Le résultat est `true` si une des deux ou les deux expressions sont `true`. Le résultat est `false` seulement si les deux expressions sont `false`. Vous pouvez utiliser l'opérateur logique OR avec n'importe quel nombre d'opérandes ; si un opérande est évalué comme `true`, le résultat est `true`.

Avec des expressions non booléennes, l'opérateur logique OR oblige Flash à évaluer l'expression de gauche ; si elle peut être convertie en `true`, le résultat est `true`. Sinon, il évalue l'expression de droite et le résultat est la valeur de cette expression.

Exemple

Usage 1 : l'exemple suivant utilise l'opérateur `||` dans une instruction `if`. La deuxième expression est `true`, le résultat final étant donc `true`

```
x = 10
y = 250
start = false
if(x > 25 || y > 200 || start){
  trace('le test de OR logique a réussi');
}
```

Usage 2 : cet exemple illustre la façon dont une expression non booléenne peut produire un résultat inattendu. Si l'expression de gauche est convertie en `true`, ce résultat est renvoyé sans convertir l'expression de droite.

```
function fx1(){
  trace ("fx1 appelé");
  return true;
}
function fx2(){
  trace ("fx2 appelé");
  return true;
}
if (fx1() || fx2()){
  trace ("instruction IF entrée");
}
// Le résultat suivant est affiché dans le panneau de sortie :
// fx1 appelé
// instruction IF entrée
```

|= (affectation OR au niveau du bit)

Disponibilité

Flash Player 5.

Usage

expression1 |= *expression2*

Paramètres

expression1,*expression2* Un nombre ou variable.

Renvoie

Rien.

Description

Opérateur (affectation composée au niveau du bit) : affecte à *expression1* la valeur de *expression1* | *expression2*. Par exemple, les deux instructions suivantes sont équivalentes :

```
x |= y;
x = x | y;
```

Exemple

L'exemple suivant utilise l'opérateur |= :

```
// 15 décimal = 1111 binaire
x = 15;
// 9 décimal = 1001 binaire
y = 9;
trace(x |= y);
// 1111 |= 1001
//renvoie 15 décimal (= 1111 binaire)
```

Consultez également

| (OR au niveau du bit)

~ (NOT au niveau du bit)

Disponibilité

Flash Player 5.

Usage

```
~ expression
```

Paramètres

expression Un nombre.

Renvoie

Rien.

Description

Opérateur (au niveau du bit) : convertit l'*expression* en entier 32 bits non signé, puis inverse les bits. Une opération NOT au niveau du bit change le signe d'un nombre et soustrait 1.

Exemple

L'exemple suivant illustre une opération NOT au niveau du bit effectuée sur une variable.

```
a = 0;
trace ("lorsque a = 0, ~a = "+~a);
// lorsque a = 0, ~a = -1
a = 1;
trace ("lorsque a = 0, ~a = "+~a);
// lorsque a = 0, ~a = -2
// donc, ~0=-1 et ~1=-2
```

+ (addition)

Disponibilité

Flash Player 4 ; Flash Player 5. Dans Flash 5 et les versions suivantes, + est un opérateur numérique ou un concaténateur de chaînes, selon le type de données du paramètre. Dans Flash 4, + est seulement un opérateur numérique. Les fichiers Flash 4 importés dans l'environnement auteur de Flash 5 ou d'une version ultérieure subissent un processus de conversion visant à préserver l'intégrité des types de données. L'exemple suivant illustre la conversion d'un fichier Flash 4 contenant une comparaison de qualité numérique :

Fichier Flash 4 :

```
x + y
```

Fichier converti Flash 5 ou version ultérieure :

```
Number(x) + Number(y)
```

Usage

```
expression1 + expression2
```

Paramètres

expression1,expression2 Un nombre ou une chaîne.

Renvoie

Rien.

Description

Opérateur : additionne des expressions numériques ou concatène (combine) des chaînes. Si une expression est une chaîne, toutes les autres expressions sont converties en chaînes et concaténées.

Si les deux expressions sont des entiers, la somme est un entier ; si une des deux ou les deux expressions sont des nombres à virgule flottante, la somme est un nombre à virgule flottante.

Exemple

Usage 1 : l'exemple suivant concatène deux chaînes et affiche le résultat dans le panneau de sortie.

```
nom = "Cola";
instrument = "percussions";
trace (nom + " joue des " + instrument);
```

Usage 2 : les variables associées à des champs de texte dynamique et de saisie ont le type de données String. Dans l'exemple suivant, la variable `dépôt` est un champ de texte de saisie sur la scène. Une fois qu'un utilisateur a entré un montant de dépôt, le script essaie d'ajouter `dépôt` à `ancienSolde`. Cependant, étant donné que `dépôt` est du type de données String, le script concatène (combine pour former une seule chaîne) les valeurs des variables plutôt que de les additionner.

```
ancienSolde = 1345.23;
soldeActuel = dépôt + ancienSolde;
trace (soldeActuel);
```

Par exemple, si l'utilisateur entre 475 dans le champ de texte de dépôt, l'action `trace()` envoie la valeur 4751345.23 au panneau de sortie.

Pour corriger cela, utilisez la fonction `Number` pour convertir la chaîne en un nombre, comme dans l'exemple suivant :

```
soldeActuel = Number(dépôt) + ancienSolde;
```

Usage 3 : cette instruction additionne les entiers 2 et 3 et affiche ensuite le résultat, l'entier 5, dans le panneau de sortie :

```
trace (2 + 3);
```

Cette instruction additionne les nombres à virgule flottante 2,5 et 3,25 et affiche le résultat, `5,75`, un nombre à virgule flottante, dans le panneau de sortie :

```
trace (2,5 + 3,25);
```

Consultez également

`_accProps`

+= (affectation d'addition)

Disponibilité

Flash Player 4.

Usage

`expression1 += expression2`

Paramètres

`expression1,expression2` Un nombre ou une chaîne.

Renvoie

Rien.

Description

Opérateur (affectation composée arithmétique) : affecte à *`expression1`* la valeur de *`expression1 + expression2`*. Par exemple, les deux instructions suivantes produisent le même résultat :

```
x += y;
x = x + y;
```

Cet opérateur effectue également la concaténation de chaînes. Toutes les règles de l'opérateur addition (+) s'appliquent à l'opérateur d'affectation d'addition (+=).

Exemple

L'exemple suivant illustre une utilisation numérique de l'opérateur +=.

```
x = 5;
y = 10;
x += y;
trace(x);
// x renvoie 15
```

Cet exemple utilise l'opérateur += avec une expression chaîne et envoie « Je m'appelle Gilbert » au panneau de sortie.

```
x = "Je m'appelle "
x += "Gilbert"
trace (x)
// renvoie "Je m'appelle Gilbert"
```

Consultez également

`+ (addition)`

<(inférieur à)

Disponibilité

Flash Player 4 ; Flash Player 5. Dans Flash 5 et les versions suivantes, l'opérateur < (inférieur à) est un opérateur de comparaison capable de manipuler divers types de données. Dans Flash 4, < est un opérateur numérique. Les fichiers Flash 4 importés dans l'environnement auteur de Flash 5 ou d'une version ultérieure subissent un processus de conversion visant à préserver l'intégrité des types de données. L'exemple suivant illustre la conversion d'un fichier Flash 4 contenant une comparaison de qualité numérique.

Fichier Flash 4 :

```
x < y
```

Fichier converti Flash 5 ou version ultérieure :

```
Number(x) < Number(y)
```

Usage

`expression1 < expression2`

Paramètres

`expression1,expression2` Un nombre ou une chaîne.

Description

Opérateur (comparaison) : compare deux expressions et détermine si *`expression1`* est inférieure à *`expression2`* le cas échéant, renvoie `true`. Si *`expression1`* est supérieure ou égale à *`expression2`*, l'opérateur renvoie `false`. Les expressions chaînes sont évaluées par ordre alphabétique ; les majuscules apparaissant avant les minuscules.

Exemple

Les exemples suivants illustrent les renvois `true` et `false` pour des comparaisons numériques et chaînes.

```
3 < 10;
// true

10 < 3;
// false

"Alain" < "Jacques";
// true

"Jacques" < "Alain";
// false

"11" < "3";
//true

"11" < 3;
// comparaison numérique
// false

"C" < "abc";
// false

"A" < "a";
// true
```

<< (décalage gauche au niveau du bit)

Disponibilité

Flash Player 5.

Usage

expression1 << *expression2*

Paramètres

expression1 Un nombre ou une expression devant être décalé(e) vers la gauche.

expression2 Un nombre ou une expression converti(e) en entier de 0 à 31.

Renvoie

Rien.

Description

Opérateur (au niveau du bit) : convertit *expression1* et *expression2* en entiers 32 bits et décale tous les bits de *expression1* vers la gauche du nombre de places spécifié par l'entier résultant de la conversion de *expression2*. Les emplacements des bits vidés par cette opération sont remplis par des 0. Le décalage d'une valeur vers la gauche de une position revient à la multiplier par deux.

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`. Cela est dû au fait que 1 décimal est égal à 1 binaire, 1 binaire décalé sur la gauche de 10 est 10000000000 binaire, et 10000000000 binaire est 1024 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`. Cela est dû au fait que 7 décimal est égal à 111 binaire, 111 binaire décalé sur la gauche de 8 bits est 11100000000 binaire, et 11100000000 binaire est 1792 décimal.

Consultez également

`>>= (décalage droit au niveau du bit et affectation)`, `>> (décalage droit au niveau du bit)`, `<<= (décalage gauche au niveau du bit et affectation)`

<<= (décalage gauche au niveau du bit et affectation)

Disponibilité

Flash Player 5.

Usage

`expression1 <<= expression2`

Paramètres

`expression1` Un nombre ou une expression devant être décalé(e) vers la gauche.

`expression2` Un nombre ou une expression converti(e) en entier de 0 à 31.

Renvoie

Rien.

Description

Opérateur (affectation composée au niveau du bit) : cet opérateur effectue une opération de décalage gauche au niveau du bit et stocke le contenu comme résultat dans *`expression1`*. Les deux expressions suivantes sont équivalentes.

```
A <<= B
A = (A << B)
```

Consultez également

`<< (décalage gauche au niveau du bit)`, `>>= (décalage droit au niveau du bit et affectation)`, `>> (décalage droit au niveau du bit)`

<= (inférieur ou égal à)

Disponibilité

Flash Player 4.

Fichier Flash 4 :

```
x <= y
```

Fichier converti Flash 5 ou version ultérieure :

```
Number(x) < Number(y)
```

Usage

```
expression1 <= expression2
```

Paramètres

expression1,expression2 Un nombre ou une chaîne.

Renvoie

Une valeur booléenne.

Description

Opérateur (comparaison) : compare deux expressions et détermine si *expression1* est inférieure ou égale à *expression2* le cas échéant, renvoie `true`. Si *expression1* est supérieure à *expression2*, l'opérateur renvoie `false`. Les expressions chaînes sont évaluées par ordre alphabétique ; les majuscules apparaissant avant les minuscules.

Dans Flash 5 ou ses versions ultérieures, l'opérateur inférieur ou égal à (<=) est un opérateur de comparaison capable de manipuler divers types de données. Dans Flash 4, <= est un opérateur numérique. Les fichiers Flash 4 importés dans l'environnement auteur de Flash 5 ou d'une version ultérieure subissent un processus de conversion visant à préserver l'intégrité des types de données. L'exemple suivant illustre la conversion d'un fichier Flash 4 contenant une comparaison de qualité numérique.

Exemple

Les exemples suivants illustrent les résultats `true` et `false` pour des comparaisons numériques et chaînes :

```
5 <= 10;
// true
2 <= 2;
// true
10 <= 3;
// false
"Alain" <= "Jacques";
// true
"Jacques" <= "Alain";
// false
"11" <= "3";
//true
"11" <= 3;
// comparaison numérique
// false
"C" <= "abc";
// false
"A" <= "a";
// true
```

<> (inégalité)

Disponibilité

Flash 2.

Usage

expression1 <> *expression2*

Paramètres

expression1,expression2 Un nombre, une chaîne, une valeur booléenne, une variable, un objet, un tableau ou une fonction.

Renvoie

Une valeur booléenne.

Description

Opérateur (inégalité) : teste l'opposé exact de l'opérateur ==. Si *expression1* est égale à *expression2*, le résultat est `false`. Tout comme pour l'opérateur ==, la définition du terme *égalité* dépend des types de données comparés :

- Les nombres, les chaînes et les valeurs booléennes sont comparés par valeur.
- Les variables, les objets, les tableaux et les fonctions sont comparés par référence.

Cet opérateur est déconseillé dans Flash 5 ; Macromedia recommande l'utilisation de l'opérateur !=.

Consultez également

`!= (inégalité)`

= (affectation)

Disponibilité

Flash Player 4.

Fichier Flash 4 :

```
x = y
```

Fichier converti Flash 5 ou version ultérieure :

```
Number(x) == Number(y)
```

Usage

```
expression1 = expression2
```

Paramètres

expression1 Une variable, un élément d'un tableau ou une propriété d'un objet.

expression2 Une valeur de tout type.

Renvoie

Rien.

Description

Opérateur : affecte le type de *expression2* (le paramètre de droite) à la variable, à l'élément de tableau ou à la propriété dans *expression1*.

Dans Flash 5 ou les versions ultérieures, = est un opérateur d'affectation et l'opérateur == est utilisé pour évaluer une égalité. Dans Flash 4, = est un opérateur d'égalité numérique. Les fichiers Flash 4 importés dans l'environnement auteur de Flash 5 ou d'une version ultérieure subissent un processus de conversion visant à préserver l'intégrité des types de données.

Exemple

L'exemple suivant utilise l'opérateur d'affectation pour affecter le type de données Number à la variable `x`.

```
x = 5
```

L'exemple suivant utilise l'opérateur d'affectation pour affecter le type de données String à la variable `x`.

```
x = "Bonjour"
```

Consultez également

== (égalité)

-= (affectation de soustraction)

Disponibilité

Flash Player 4.

Usage

expression1 -= expression2

Paramètres

expression1,expression2 Un nombre ou une expression évaluée comme nombre.

Renvoie

Rien.

Description

Opérateur (affectation composée arithmétique) : affecte à *expression1* la valeur de *expression1 - expression2*. Par exemple, les deux instructions suivantes sont équivalentes :

```
x -= y;
x = x - y;
```

Les expressions chaînes doivent être converties en nombres, sinon `NaN` est renvoyé.

Exemple

Usage 1 : l'exemple suivant utilise l'opérateur `-=` pour soustraire 10 de 5 et affecter le résultat à la variable `x`.

```
x = 5;
y = 10;
x -= y
trace(x);
//renvoie -5
```

Usage 2 : l'exemple suivant montre comment des chaînes sont converties en nombres.

```
x = "5";
y = "10";
x -= y;
trace(x);
// renvoie -5
```

== (égalité)

Disponibilité

Flash Player 5.

Usage

expression1 == expression2

Paramètres

expression1,expression2 Un nombre, une chaîne, une valeur booléenne, une variable, un objet, un tableau ou une fonction.

Renvoie

Une valeur booléenne.

Description

Opérateur (égalité) : teste l'égalité de deux expressions. Le résultat est `true` si les deux expressions sont égales.

La définition du terme *égalité* dépend du type de données du paramètre :

- Les nombres et les valeurs booléennes sont comparés par valeur et sont considérés égaux s'ils possèdent les mêmes valeurs.
- Les expressions chaînes sont égales si elles possèdent le même nombre de caractères (identiques).
- Les variables, les objets, les tableaux et les fonctions sont comparés par référence. Deux variables sont égales si elles font référence au même objet ou tableau ou à la même fonction. Deux tableaux distincts ne sont jamais considérés égaux, même s'ils comportent le même nombre d'éléments.

Exemple

Usage 1 : l'exemple suivant utilise l'opérateur == avec une instruction `if` :

```
a = "David" , b = "David";
if (a == b){
  trace("David est David");
}
```

Usage 2 : ces exemples montrent les résultats d'opérations qui comparent des types différents.

```
x = "5"; y = "5";
trace (x == y);
// true

x = "5"; y = "66";
trace (x == y);
// false

x = "chris"; y = "steve";
trace (x == y);
//false
```

Consultez également

`!= (inégalité)`, `=== (égalité stricte)`, `!== (inégalité stricte)`

=== (égalité stricte)

Disponibilité

Flash Player 6.

Usage

expression1 `===` *expression2*

Renvoie

Une valeur booléenne.

Description

Opérateur : teste l'égalité de deux expressions – l'opérateur d'égalité stricte fonctionne de la même façon que l'opérateur d'égalité, à ceci près que les types de données ne sont pas convertis. Le résultat est `true` si les deux expressions, y compris leurs types de données, sont égales.

La définition du terme *égalité* dépend du type de données du paramètre :

- Les nombres et les valeurs booléennes sont comparés par valeur et sont considérés égaux s'ils possèdent les mêmes valeurs.
- Les expressions chaînes sont égales si elles possèdent le même nombre de caractères (identiques).
- Les variables, les objets, les tableaux et les fonctions sont comparés par référence. Deux variables sont égales si elles font référence au même objet ou tableau ou à la même fonction. Deux tableaux distincts ne sont jamais considérés égaux, même s'ils comportent le même nombre d'éléments.

Exemple

Le code suivant affiche la valeur renvoyée par les opérations utilisant les opérateurs d'égalité, d'égalité stricte et d'inégalité stricte.

```
s1 = new String("5");
s2 = new String("5");
s3 = new String("Bonjour");
n = new Number(5);
b = new Boolean(true);

s1 == s2; // true
s1 == s3; // false
s1 == n; // true
s1 == b; // false

s1 === s2; // true
s1 === s3; // false
s1 === n; // false
s1 === b; // false

s1 !== s2; // false
s1 !== s3; // true
s1 !== n; // true
s1 !== b; // true
```

Consultez également

`==` (égalité), `!=` (inégalité), `===` (égalité stricte)

> (supérieur à)

Disponibilité

Flash Player 4.

Fichier Flash 4 :

```
x > y
```

Fichier converti Flash 5 ou version ultérieure :

```
Number(x) > Number(y)
```

Usage

```
expression1 > expression2
```

Paramètres

expression1,expression2 Un nombre ou une chaîne.

Renvoie

Une valeur booléenne.

Description

Opérateur (comparaison) : compare deux expressions et détermine si *expression1* est supérieure à *expression2* ; dans ce cas, l'opérateur renvoie `true`. Si *expression1* est inférieure ou égale à *expression2*, l'opérateur renvoie `false`. Les expressions chaînes sont évaluées par ordre alphabétique ; les majuscules apparaissant avant les minuscules.

Dans Flash 5 ou ses versions ultérieures, l'opérateur inférieur ou égal à (<=) est un opérateur de comparaison capable de manipuler divers types de données. Dans Flash 4, <= est un opérateur numérique. Les fichiers Flash 4 importés dans l'environnement auteur de Flash 5 ou d'une version ultérieure subissent un processus de conversion visant à préserver l'intégrité des types de données.

>= (supérieur ou égal à)

Disponibilité

Flash Player 4.

Fichier Flash 4 :

```
x > y
```

Fichier converti Flash 5 ou version ultérieure :

```
Number(x) > Number(y)
```

Usage

```
expression1 >= expression2
```

Paramètres

expression1, expression2 Une chaîne, un entier ou un nombre à virgule flottante.

Renvoie

Une valeur booléenne.

Description

Opérateur (comparaison) : compare deux expressions et détermine si *expression1* est supérieure ou égale à *expression2* (`true`) ou si *expression1* est inférieure à *expression2* (`false`).

Dans Flash 5 ou les versions ultérieures, supérieur ou égal à (>) est un opérateur de comparaison capable de manipuler divers types de données. Dans Flash 4, > est un opérateur numérique. Les fichiers Flash 4 importés dans l'environnement auteur de Flash 5 ou d'une version ultérieure subissent un processus de conversion visant à préserver l'intégrité des types de données.

>> (décalage droit au niveau du bit)

Disponibilité

Flash Player 5.

Usage

expression1 >> *expression2*

Paramètres

expression1 Un nombre ou une expression devant être décalé(e) vers la droite.

expression2 Un nombre ou une expression converti(e) en entier de 0 à 31.

Renvoie

Rien.

Description

Opérateur (au niveau du bit) : convertit *expression1* et *expression2* en entiers 32 bits et décale tous les bits de *expression1* vers la droite du nombre de places spécifié par l'entier résultant de la conversion de *expression2*. Les bits décalés vers la droite sont éliminés. Pour préserver le signe de l'*expression* d'origine, les bits de gauche sont remplis de 0 si le bit le plus significatif (le bit le plus à gauche) de *expression1* est 0 et remplis de 1 si le bit le plus significatif est 1. Le décalage d'une valeur à droite d'une position revient à la diviser par 2 et à éliminer le reste.

Exemple

L'exemple suivant convertit 65535 en entier de 32 bits et le décale de 8 bits vers la droite.

```
x = 65535 >> 8
```

Le résultat de l'opération précédente est :

```
x = 255
```

Ceci est dû au fait que 65535 décimal est égal à 1111111111111111 binaire (seize 1), 1111111111111111 binaire décalé à droite de 8 bits est 11111111 binaire, et 11111111 binaire est 255 décimal. Le bit le plus significatif est 0 car les entiers sont 32 bits, le bit de remplissage étant donc 0.

L'exemple suivant convertit -1 en un entier de 32 bits et le décale de 1 bit vers la droite.

```
x = 1 >>1
```

Le résultat de l'opération précédente est :

```
x = -1
```

Ceci est dû au fait que -1 décimal est égal à 11111111111111111111111111111111 binaire (trente-deux 1), le décalage vers la droite de 1 bit fait que le bit le moins significatif (le bit le plus à droite) est éliminé et que le bit le plus significatif est rempli de 1. Le résultat est 11111111111111111111111111111111 (trente-deux 1) binaire, ce qui représente l'entier de 32 bits -1.

Consultez également

>>= (décalage droit au niveau du bit et affectation)

>>= (décalage droit au niveau du bit et affectation)

Disponibilité

Flash Player 5.

Usage

expression1 =>>*expression2*

Paramètres

expression1 Un nombre ou une expression devant être décalé(e) vers la gauche.

expression2 Un nombre ou une expression converti(e) en entier de 0 à 31.

Renvoie

Rien.

Description

Opérateur (affectation composée au niveau du bit) : cet opérateur effectue une opération de décalage droit au niveau du bit et stocke le contenu comme résultat dans *expression1*.

Exemple

Les deux expressions suivantes sont équivalentes.

```
A >>= B
A = (A >> B)
```

Le code commenté suivant utilise l'opérateur (>>=) au niveau du bit. Cet exemple illustre également l'utilisation de tous les opérateurs au niveau du bit.

```
function convertToBinary(number){
  var result = "";
  for (var i=0; i<32; i++) {
    // extraire le bit le moins significatif avec AND au niveau du bit
    var lsb = number & 1;
    // ajouter ce bit à la chaîne de résultat
    result = (lsb ? "1" : "0") + result;
    // décaler le nombre de un bit vers la droite, pour voir le bit suivant
    number >>= 1;}
  return result;
}
trace(convertToBinary(479));
// renvoie la chaîne 00000000000000000000000111011111
// la chaîne ci-dessus est une représentation binaire
// du décimal 479
```

Consultez également

<< (décalage gauche au niveau du bit)

>>> (décalage droit non signé au niveau du bit)

Disponibilité

Flash Player 5.

Usage

expression1 >>> *expression2*

Paramètres

expression1 Un nombre ou une expression devant être décalé(e) vers la droite.

expression2 Un nombre ou une expression converti(e) en entier de 0 à 31.

Renvoie

Rien.

Description

Opérateur (au niveau du bit) : identique à l'opérateur de décalage droit au niveau du bit (>>) excepté qu'il ne conserve pas le signe de l'*expression* d'origine car les bits de gauche sont toujours remplis avec des 0.

Exemple

L'exemple suivant convertit -1 en un entier de 32 bits et le décale de 1 bit vers la droite.

```
x = 1 >>>1
```

Le résultat de l'opération précédente est :

```
x = 2147483647
```

Cela est dû au fait que -1 décimal est 11111111111111111111111111111111 binaire (trente-deux 1) et, quand vous le décalez vers la droite (non signé) de 1 bit, le bit le moins significatif (le plus à droite) est éliminé et le bit le plus significatif (le plus à gauche) est rempli avec un 0. Le résultat est 01111111111111111111111111111111 binaire, qui représente l'entier de 32 bits 2147483647.

Consultez également

>>= (décalage droit au niveau du bit et affectation)

>>>= (décalage droit non signé au niveau du bit et affectation)

Disponibilité

Flash Player 5.

Usage

expression1 >>>= *expression2*

Paramètres

expression1 Un nombre ou une expression devant être décalé(e) vers la gauche.

expression2 Un nombre ou une expression converti(e) en entier de 0 à 31.

Renvoie

Rien.

Description

Opérateur (affectation composée au niveau du bit) : effectue une opération de décalage vers la droite au niveau du bit non signé et stocke le contenu comme résultat dans *expression1*. Les deux expressions suivantes sont équivalentes :

```
A >>>= B
A = (A >>> B)
```

Consultez également

>>> (décalage droit non signé au niveau du bit), >>= (décalage droit au niveau du bit et affectation)

Classe Accessibility

Disponibilité

Flash Player 6 version 65.

Description

La classe Accessibility gère la communication avec les lecteurs d'écran. Les méthodes de la classe Accessibility sont statiques, c'est-à-dire qu'il n'est pas nécessaire de créer une occurrence de la classe afin d'en utiliser les méthodes.

Afin de récupérer et de définir les propriétés accessibles pour un objet spécifique, tel qu'un bouton, un clip, un champ de texte, utilisez la propriété _accProps. Pour déterminer si le lecteur est exécuté dans un environnement supportant les aides d'accessibilité, utilisez System.capabilities.hasAccessibility.

Méthode de la classe Accessibility

Méthode	Description
Accessibility.isActive()	Indique si un programme de lecture d'écran est actif.
Accessibility.updateProperties()	Met à jour la description des objets sur l'écran pour les logiciels de lecture d'écran.

Accessibility.isActive()

Disponibilité

Flash Player 6 version 65.

Usage

```
Accessibility.isActive()
```

Paramètres

Aucun.

Renvoie

Une valeur booléenne de `true` s'il existe des clients Microsoft Active Accessibility (MSAA) actifs et si le lecteur est exécuté dans un environnement supportant une communication entre Flash Player et des aides d'accessibilité, sinon `false`.

Description

Méthode : indique si un programme de lecture d'écran MSAA est actif et si le lecteur est exécuté dans un environnement supportant une communication entre Flash Player et des aides d'accessibilité. Utilisez cette méthode lorsque vous voulez que votre animation se comporte différemment en présence d'un lecteur d'écran.

Pour déterminer si le lecteur est exécuté dans un environnement supportant les aides d'accessibilité, utilisez `System.capabilities.hasAccessibility`.

Remarque : Si vous appelez cette méthode dans les une ou deux secondes suivant la première apparition de la fenêtre Flash dans laquelle votre document est lu, vous pourrez obtenir une valeur de renvoi `false` même s'il existe un client MSAA actif. Cela est dû à un mécanisme de communication asynchrone entre les clients Flash et MSAA. Pour éviter le problème, attendez une à deux secondes après avoir chargé votre document pour appeler cette méthode.

Consultez également

`Accessibility.updateProperties()`, `_accProps`, `System.capabilities.hasAccessibility`

Accessibility.updateProperties()

Disponibilité

Flash Player 6 version 65.

Usage

```
Accessibility.updateProperties()
```

Paramètres

Aucun.

Renvoie

Rien.

Description

Méthode : Flash Player réexamine toutes les propriétés d'accessibilité, met à jour sa description des objets pour les lecteurs d'écran et, si nécessaire, envoie des événements aux lecteurs d'écran afin de signifier que des changements ont eu lieu. Pour plus d'informations sur la définition de propriétés d'accessibilité, consultez `_accProps`.

Pour déterminer si le lecteur est exécuté dans un environnement supportant les aides d'accessibilité, utilisez `System.capabilities.hasAccessibility`.

Si vous modifiez les propriétés d'accessibilité de plusieurs objets, un seul appel de `Accessibility.updateProperties()` est nécessaire ; plusieurs appels peuvent réduire les performances et donner des résultats de lecture d'écran incompréhensibles.

Exemple

Le code ActionScript suivant tire profit des propriétés d'accessibilité dynamiques. Cet exemple est celui d'un bouton non texte pouvant changer l'icône affiché.

```
function definirIcone( numNouvelIcone, nouvelEquivalentTexte )
{
   this.imageIcone = this.imagesIcone[ numNouvelIcone ];
   if ( nouvelEquivalentTexte != undefined )
   {
      if ( this._accProps == undefined )
         this._accProps = new Object();
      this._accProps.name = newTextEquivalent;
      Accessibility.updateProperties()
   }
}
```

Consultez également

`Accessibility.isActive()`, `_accProps`, `System.capabilities.hasAccessibility`

_accProps

Disponibilité

Flash Player 6 version 65.

Usage

```
_accProps.nomDeProp
nomDoccurrence._accProps.nomDeProp
```

Paramètres

nomDeProp Un nom de propriété d'accessibilité (consultez la description suivante pour connaître les noms valides).

nomDoccurrence Le nom d'occurrence attribué à une occurrence d'un clip, d'un bouton, d'un champ de texte dynamique ou d'un champ de texte de saisie.

Description

Propriété : vous permet de contrôler les options d'accessibilité de lecture d'écran des fichiers SWF, des clips, des boutons, des champs de texte dynamiques et des champs de texte de saisie au moment de l'exécution. Ces propriétés remplacent les paramètres correspondants disponibles dans le panneau Accessibilité au cours de la programmation. Pour que les changements de ces propriétés prennent effet, vous devez appeler `Accessibility.updateProperties()`. Pour plus d'informations sur le panneau Accessibilité, consultez « Panneau Accessibilité de Flash », dans le guide Utilisation de Flash de l'aide.

Pour déterminer si le lecteur est exécuté dans un environnement supportant les aides d'accessibilité, utilisez `System.capabilities.hasAccessibility`.

Le tableau suivant répertorie le nom et le type de données de chaque propriété `_accProps`, son paramètre correspondant dans le panneau Accessibilité, ainsi que les types d'objets auxquels la propriété peut être appliquée. Le terme *logique inverse* signifie que le paramètre de propriété est l'inverse du paramètre correspondant dans le panneau Accessibilité. Par exemple, définir la propriété `silent` sur `true` revient à désactiver l'option Rendre l'animation accessible ou Rendre l'objet accessible.

Propriété	Type de données	Equivalent dans le panneau Accessibilité	S'applique à
`silent`	Boolean	Rendre l'animation accessible/ Rendre l'objet accessible *(logique inverse)*	Animations entières Clips Boutons Texte dynamique Texte de saisie
`forceSimple`	Boolean	Rendre les objets enfants accessibles *(logique inverse)*	Animations entières Clips
`name`	String	Nom	Animations entières Clips Boutons Texte de saisie

Propriété	Type de données	Equivalent dans le panneau Accessibilité	S'applique à
`description`	String	Description	Animations entières Clips Boutons Texte dynamique Texte de saisie
`raccourci`	String	Raccourci*	Clips Boutons Texte de saisie

* Pour plus d'informations sur l'affectation d'un raccourci clavier à un objet accessible, consultez `Key.addListener()`.

Pour définir des paramètres correspondant au paramètre Ordre des tabulations du panneau Accessibilité, utilisez la propriété `Button.tabIndex`, `MovieClip.tabIndex`, ou `TextField.tabIndex`.

Il n'est pas possible de définir un paramètre d'étiquetage automatique au moment de l'exécution.

Lorsqu'elles sont faites sans le paramètre *`nomDoccurrence`*, les modifications apportées aux propriétés `_accProps` s'appliquent à l'animation entière. Par exemple, le code suivant définit la chaîne `"Magasin animalier"` en tant que propriété d'accessibilité `nom` pour l'animation entière, puis appelle `Accessibility.updateProperties()` pour appliquer cette modification.

```
_accprops.name = "Magasin animalier";
Accessbility.updateProperties();
```

Au contraire, le code suivant définit la chaîne `"Prix"` en tant que propriété `nom` d'un clip avec le nom d'occurrence `prix_mc` :

```
prix_mc._accProps.name = "Prix";
Accessbility.updateProperties();
```

Si vous définissez plusieurs propriétés d'accessibilité, apportez autant de modifications que nécessaire avant d'appeler `Accessibility.updateProperties()`, plutôt que de l'appeler après chaque instruction de propriété :

```
_accprops.name = "Magasin animalier";
animal_mc._accProps.name = "Animal";
animal_mc._accProps.description = "Chat, chien, poisson, etc.";
prix_mc._accProps.name = "Prix";
price_mc._accProps.description = "Coût d'un article";
Accessbility.updateProperties();
```

Si vous ne définissez pas de propriété d'accessibilité pour une animation ou un objet, toutes les valeurs définies dans le panneau Accessibilité sont appliquées.

Lorsque vous avez défini une propriété d'accessibilité, vous ne pouvez plus reprendre la valeur définie dans le panneau Accessibilité. Vous pouvez cependant définir la propriété à sa valeur par défaut (`false` pour les valeurs booléennes, chaînes vides pour des valeurs de chaînes) en supprimant l'objet `_accProps` :

```
monClip_mc._accProps.silent = true; // définir une propriété
// autre code ici
delete monClip_mc._accProps.silent; // récupérer la valeur par défaut
```

Pour remplacer toutes les valeurs d'accessibilité d'un objet par les valeurs par défaut, vous pouvez supprimer l'objet *nomDoccurrence*._accProps :

```
delete mon_btn._accProps;
```

Pour remplacer toutes les valeurs d'accessibilité de tous les objets par les valeurs par défaut, vous pouvez supprimer l'objet global `accProps` :

```
delete _accProps;
```

Si vous définissez une propriété pour un type d'objet qui ne supporte pas cette propriété, l'affectation de la propriété est ignorée et aucune erreur n'est retournée. Par exemple, la propriété `forceSimple` n'est pas supportée pour les boutons. Une ligne similaire à la suivante est donc ignorée :

```
mon_btn._accProps.forceSimple = false; //ignorée
```

Exemple

Vous trouverez ci-après un exemple de code ActionScript tirant profit des propriétés d'accessibilité dynamiques. Vous pourriez affecter ce code à un bouton icône non texte pouvant changer l'icône affiché.

```
function definirIcone( numNouvelIcone, nouvelEquivalentTexte )
{
   this.imageIcone = this.imagesIcone[ numNouvelIcone ];
   if ( nouvelEquivalentTexte != undefined )
   {
      if ( this._accProps == undefined )
         this._accProps = new Object();
      this._accProps.name = newTextEquivalent;
      Accessibility.updateProperties()
   }
}
```

Consultez également

`Accessibility.isActive()`, `Accessibility.updateProperties()`, `System.capabilities.hasAccessibility`

add

Disponibilité

Flash Player 4.

Usage

chaîne1 add *chaîne2*

Paramètres

chaîne1, chaîne2 Une chaîne.

Renvoie

Rien.

Description

Opérateur : concatène (combine) deux ou plusieurs chaînes. L'opérateur `add` remplace l'opérateur d'addition (`&`) de Flash 4. Les fichiers Flash 4 utilisant l'opérateur `&` sont automatiquement convertis de façon à utiliser l'opérateur `add` pour la concaténation de chaînes lorsqu'ils sont importés dans l'environnement auteur de Flash 5 ou d'une version ultérieure. Toutefois, l'opérateur `add` n'est pas conseillé dans Flash Player5 ; Macromedia recommande l'utilisation de l'opérateur `+` pour la création de contenu pour Flash Player 5 ou une version ultérieure. Utilisez l'opérateur `add` pour concaténer des chaînes si vous créez des contenus pour Flash Player 4 ou une version antérieure de ce lecteur.

Consultez également

`+ (addition)`

and

Disponibilité

Flash Player 4.

Usage

condition1 and *condition2*

Paramètres

condition1,condition2 Conditions ou expressions qui équivalent à `true` ou `false`.

Renvoie

Rien.

Description

Opérateur : effectue une opération AND logique dans Flash Player 4. Si les deux expressions équivalent à `true`, l'expression entière est alors `true`. Cet opérateur n'est pas recommandé dans Flash 5 ; Macromedia recommande l'utilisation de l'opérateur `&&`.

Consultez également

`&& (AND logique)`

Classe Arguments

Disponibilité

Flash Player 5 ; propriété ajoutée à Flash Player 6.

Description

La classe Arguments est un tableau qui contient les valeurs qui ont été transmises comme paramètres à une fonction. Un objet Arguments est automatiquement créé à chaque fois qu'une fonction est appelée dans ActionScript. Une variable locale, `arguments`, est également créée et vous permet de faire référence à l'objet Arguments.

Propriétés de la classe Arguments

Propriété	Description
`arguments.callee`	Fait référence à la fonction appelée.
`arguments.caller`	Fait référence à la fonction appelante.
`arguments.length`	Le nombre de paramètres transmis à une fonction.

arguments.callee

Disponibilité

Flash Player 5.

Usage

```
arguments.callee
```

Description

Propriété : fait référence à la fonction qui est actuellement appelée.

Exemple

Vous pouvez utiliser la propriété `arguments.callee` pour créer une fonction anonyme récurrente, comme dans l'exemple suivant :

```
factorial = function (x) {
  if (x <= 1) {
    return 1;
  } else {
    return x * arguments.callee(x-1);
  }
};
```

L'exemple suivant est une fonction récurrente nommée :

```
function factorial (x) {
  if (x <= 1) {
    return 1;
  } else {
    return x * factorial(x-1);
  }
}
```

arguments.caller

Disponibilité

Flash Player 6.

Usage

```
arguments.caller
```

Description

Propriété : fait référence à la fonction d'appel.

arguments.length

Disponibilité

Flash Player 5.

Usage

```
arguments.length
```

Description

Propriété : le nombre de paramètres transmis à une fonction.

Classe Array

Disponibilité

Flash Player 5 (est devenu un objet natif dans Flash Player 6, améliorant ainsi les performances de manière significative).

Description

La classe Array vous permet d'accéder à des tableaux et de les manipuler. Un tableau est un objet dont les propriétés sont identifiées par un nombre représentant leur position dans le tableau. Ce nombre est appelé *index*. Tous les tableaux sont basés sur zéro, ce qui signifie que le premier élément du tableau est [0], le second élément est [1], et ainsi de suite. Dans l'exemple suivant, `mon_array` contient les mois de l'année.

```
mon_array[0] = "Janvier"
mon_array[1] = "Février"
mon_array[2] = "Mars"
mon_array[3] = "Avril"
```

Pour créer un objet Array, utilisez le constructeur `new Array` ou l'opérateur d'accès tableau (`[]`). Pour accéder aux éléments d'un tableau, utilisez l'opérateur d'accès tableau (`[]`).

Méthodes de la classe Array

Méthode	Description
Array.concat	Concatène les paramètres et les renvoie sous forme de nouveau tableau.
Array.join()	Joint tous les éléments d'un tableau dans une chaîne.
Array.pop()	Supprime le dernier élément d'un tableau et renvoie sa valeur.
Array.push()	Ajoute un ou plusieurs éléments à la fin d'un tableau et renvoie la nouvelle longueur du tableau.
Array.reverse()	Inverse la direction d'un tableau.
Array.shift	Supprime le premier élément d'un tableau et renvoie sa valeur.
Array.slice	Extrait une section d'un tableau et la renvoie sous forme de nouveau tableau.
Array.sort	Trie un tableau en place.
Array.sortOn()	Trie un tableau en fonction d'un champ du tableau.
Array.splice()	Ajoute et supprime des éléments d'un tableau.
Array.toString()	Renvoie une valeur de chaîne représentant les éléments dans l'objet Array.
Array.unshift()	Ajoute un ou plusieurs éléments au début d'un tableau et renvoie la nouvelle longueur du tableau.

Méthodes de la classe Array

Propriété	Description
Array.length	Un entier non basé sur zéro spécifiant le nombre d'éléments dans un tableau.

Constructeur de l'objet Array

Disponibilité

Flash Player 5.

Usage

```
new Array()
new Array(longueur)
new Array(élément0, élément1, élément2,...élémentN)
```

Paramètres

Longueur Un entier spécifiant le nombre d'éléments dans un tableau. Dans le cas d'éléments non contigus, le paramètre *longueur* spécifie le numéro d'index du dernier élément du tableau plus 1.

élément0...élémentN Une liste de deux ou plusieurs valeurs arbitraires. Les valeurs peuvent être des nombres, des chaînes, des objets ou d'autres tableaux. Le premier élément d'un tableau a toujours un index ou une position de 0.

Renvoie

Rien.

Description

Constructeur : permet de créer un tableau. Vous pouvez utiliser le constructeur pour créer différents types de tableaux : un tableau vide, un tableau avec une longueur spécifique, mais dont les éléments n'ont pas de valeur, ou un tableau dont les éléments ont des valeurs spécifiques.

Usage 1 : si vous ne spécifiez aucun paramètre, un tableau de longueur zéro est créé.

Usage 2 : si vous spécifiez seulement une longueur, un tableau est créé avec le nombre *longueur* d'éléments sans valeur.

Usage 3 : si vous utilisez les paramètres *élément* pour spécifier des valeurs, le tableau est créé avec des valeurs spécifiques.

Exemple

Usage 1 : l'exemple suivant crée un nouvel objet Array avec une longueur initiale de 0.

```
mon_array = new Array();
trace(mon_array.length); // renvoie 0
```

Usage 2 : l'exemple suivant crée un nouvel objet Array avec une longueur initiale de 4.

```
mon_array = new Array(4);
trace(mon_array.length); // renvoie 4
```

Usage 3 : l'exemple suivant crée le nouvel objet Array `go_gos_array`, avec une longueur initiale de 5.

```
go_gos_array = new Array("Belinda", "Gina", "Kathy", "Charlotte", "Jane");
trace(mon_array.length); // renvoie 5
trace(go_gos_array.join(", ")); // affiche les éléments
```

Les éléments initiaux du tableau `go_gos` sont identifiés comme suit :

```
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 change le deuxième élément :

```
go_gos_array[5] = "Donna";
go_gos_array[1] = "Nina"
trace(go_gos_array.join(" + "));
```

Consultez également

`Array.length`, `[] (accès tableau)`

Array.concat

Disponibilité

Flash Player 5.

Usage

mon_array.concat([*valeur0,valeur1,...valeurN*])

Paramètres

valeur0,...valeurN Nombres, éléments ou chaînes devant être concaténés dans un nouveau tableau. Si vous ne définissez aucune valeur, une copie de *mon_array* est créée.

Renvoie

Rien.

Description

Méthode : concatène les éléments spécifiés dans les paramètres avec les éléments de *mon_array* et crée un nouveau tableau. Si les paramètres *valeur* spécifient un tableau, ce sont les éléments de ce tableau qui sont concaténés et non le tableau même. Le tableau *mon_array* reste inchangé.

Exemple

Le code suivant concatène deux tableaux.

```
alpha_array = new Array("a","b","c");
numeric_array = new Array(1,2,3);
alphaNumeric_array=alpha_array.concat(numeric_array);
trace(alphaNumeric_array);
// crée le tableau ["a","b","c",1,2,3]
```

Le code suivant concatène trois tableaux.

```
num1_array = [1,3,5];
num2_array = [2,4,6];
num3_array = [7,8,9];
nums_array=num1_array.concat(num2_array,num3_array)
trace(nums_array);
// crée le tableau [1,3,5,2,4,6,7,8,9]
```

Les tableaux imbriqués ne sont pas aplatis de la même façon que les tableaux normaux. Les éléments d'un tableau imbriqué ne sont pas divisés en éléments séparés dans le `x_array`, comme dans l'exemple suivant.

```
a_array = new Array ("a","b","c");

// 2 et 3 sont des éléments d'un tableau imbriqué
n_array = new Array(1, [2, 3], 4);

x_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
```

Array.join()

Disponibilité

Flash Player 5.

Usage

```
mon_array.join([séparateur])
```

Paramètres

Séparateur Un caractère ou une chaîne qui sépare les éléments de tableau dans la chaîne renvoyée. Si vous omettez ce paramètre, une virgule est utilisée comme séparateur par défaut.

Renvoie

Chaînes.

Description

Méthode : convertit les éléments d'un tableau en chaînes, insère le séparateur spécifié entre les éléments, les concatène, et renvoie la chaîne résultante. Un tableau imbriqué est toujours séparé par une virgule et non par le séparateur transmis à la méthode `join()`.

Exemple

L'exemple suivant crée un tableau avec trois éléments : Terre, Lune et Soleil. Il joint ensuite le tableau trois fois —d'abord en utilisant le séparateur par défaut (une virgule et un espace), puis en utilisant un tiret, puis le signe plus (+)— et les affiche dans le panneau de sortie :

```
a_array = new Array("Terre","Lune","Soleil")
trace(a_array.join());
// renvoie Terre, Lune, Soleil
trace(a_array.join(" - "));
// renvoie Terre - Lune - Soleil
trace(a_array.join(" + "));
// renvoie Terre + Lune + Soleil
```

Array.length

Disponibilité

Flash Player 5.

Usage

mon_array.length

Description

Propriété : un entier non basé sur zéro spécifiant le nombre d'éléments dans un tableau. Cette propriété est automatiquement mise à jour lorsque de nouveaux éléments sont ajoutés au tableau. Lorsque vous affectez une valeur à un élément de tableau (par exemple, *mon_array*[*index*] = *valeur*), si *index* est un nombre et *index*+1 est supérieur à la propriété length, la propriété length est mise à jour à *index*+1.

Exemple

Le code suivant explique la façon dont la propriété length est mise à jour.

```
mon_array = new Array();
trace(mon_array.length); // la longueur initiale est 0
mon_array[0] = 'a';
trace(mon_array.length); // mon_array.length est mis à jour à 1
mon_array[1] = 'b';
trace(mon_array.length); // mon_array.length est mis à jour à 2
mon_array[9] = 'c';
trace(mon_array.length); // mon_array.length est mis à jour à 10
```

Array.pop()

Disponibilité

Flash Player 5.

Usage

mon_array.pop()

Paramètres

Aucun.

Renvoie

La valeur du dernier élément dans le tableau spécifié.

Description

Méthode : supprime le dernier élément d'un tableau et renvoie la valeur de cet élément.

Exemple

Le code suivant crée le tableau mesAnimaux contenant quatre éléments et supprime le dernier élément.

```
mesAnimaux = ["chat", "chien", "oiseau", "poisson"];
popped = mesAnimaux.pop();
trace(popped);
// renvoie poisson
```

Array.push()

Disponibilité

Flash Player 5.

Usage

mon_array.push(*valeur,...*)

Paramètres

Valeur Une ou plusieurs valeurs à ajouter au tableau.

Renvoie

La longueur du nouveau tableau.

Description

Méthode : ajoute un ou plusieurs éléments à la fin du tableau et renvoie la nouvelle longueur du tableau.

Exemple

L'exemple suivant crée le tableau `mesAnimaux` contenant deux éléments, `chat` et `chien`. La deuxième ligne ajoute deux éléments au tableau. Après l'appel de la méthode `push()`, la variable `pushed` contient quatre éléments. La méthode `push()` renvoyant la nouvelle longueur du tableau, l'action `trace()` de la dernière ligne envoie la nouvelle longueur de `mesAnimaux` (`4`) au panneau de sortie :

```
mesAnimaux = ["chat", "chien"];
pushed = mesAnimaux.push("oiseau", "poisson");
trace(pushed);
```

Array.reverse()

Disponibilité

Flash Player 5.

Usage

```
mon_array.reverse()
```

Paramètres

Aucun.

Renvoie

Rien.

Description

Méthode : inverse le tableau en place.

Exemple

L'exemple suivant illustre cette méthode.

```
var numbers_array = [1, 2, 3, 4, 5, 6];
trace(numbers_array.join()); //1,2,3,4,5,6
numbers_array.reverse();
trace(numbers_array.join()); // 6,5,4,3,2,1
```

Array.shift

Disponibilité

Flash Player 5.

Usage

```
mon_array.shift()
```

Paramètres

Aucun.

Renvoie

Le premier élément d'un tableau.

Description

Méthode : supprime le premier élément d'un tableau et renvoie sa valeur.

Exemple

Le code suivant crée le tableau `mesAnimaux` puis supprime le premier élément du tableau et l'affecte à la variable `shifted`.

```
var mesAnimaux_array = ["chat", "chien", "oiseau", "poisson"];
shifted = mesAnimaux_array.shift();
trace(shifted); // renvoie "chat"
```

Consultez également

`Array.pop()`

Array.slice

Disponibilité

Flash Player 5.

Usage

```
mon_array.slice( [ debut [ , fin ] ] )
```

Paramètres

debut Un nombre spécifiant l'index du point de début de la section. Si *debut* est un nombre négatif, le point de début commence à la fin du tableau, où -1 est le dernier élément.

fin Un nombre spécifiant l'index du point de fin de la section. Si vous omettez ce paramètre, la section comprend tous les éléments du début à la fin du tableau. Si *fin* est un nombre négatif, le point de fin est spécifié depuis la fin du tableau, où -1 est le dernier élément.

Renvoie

Un tableau.

Description

Méthode : extrait une section ou une sous-chaîne du tableau et la renvoie sous forme de nouveau tableau sans modifier le tableau original. Le tableau renvoyé comprend l'élément *debut* et tous les éléments jusqu'à l'élément *fin* (exclu).

Si vous ne définissez aucun paramètre, une copie de *mon_array* est créée.

Array.sort

Disponibilité

Flash Player 5 ; fonctionnalités supplémentaires ajoutées à Flash Player 7.

Usage

```
mon_array.sort()
mon_array.sort(fonctionDeComparaison)
mon_array.sort(option | option |... )
mon_array.sort(fonctionDeComparaison, option | option |... )
```

Paramètres

fonctionDeComparaison Une fonction de comparaison facultative utilisée pour déterminer l'ordre de tri des éléments d'un tableau. Etant donné les éléments A et B, le résultat de *fonctionDeComparaison* peut prendre l'une des trois valeurs suivantes :

- -1 si A doit apparaître avant B dans la séquence triée
- 0 si A = B
- -1 si A doit apparaître après B dans la séquence triée

option Un ou plusieurs nombres ou chaînes, séparés par l'opérateur `|` `(OR au niveau du bit)` qui modifie le comportement du tri par rapport au tri par défaut. Les valeurs suivantes sont possibles pour *option* :

- 1 ou `Array.CASEINSENSITIVE`
- 2 ou `Array.DESCENDING`
- 4 ou `Array.UNIQUE`
- 8 ou `Array.RETURNINDEXEDARRAY`
- 16 ou `Array.NUMERIC`

Pour plus d'informations sur ce paramètre, consultez `Array.sortOn()`.

Renvoie

La valeur renvoyée dépend de la définition des paramètres :

- Si vous définissez une valeur de 4 ou Array.UNIQUE pour *option* et que deux éléments ou plus triés comportent des champs de tri identiques, Flash renvoie une valeur de 0 et ne modifie pas le tableau.
- Si vous définissez une valeur de 8 ou Array.RETURNINDEXEDARRAY pour *option*, Flash renvoie un tableau illustrant les résultats du tri et ne modifie pas le tableau.
- Sinon, Flash ne renvoie aucune donnée et modifie le tableau d'après l'ordre du tri.

Description

Méthode : trie les éléments dans un tableau. Flash trie en fonction des valeurs ASCII (Unicode). Si l'un ou l'autre des éléments comparés ne comprend pas le champ défini dans le paramètre *nomDeChamp*, le champ est considéré `undefined` et les éléments sont placés les uns à la suite des autres dans le tableau trié, sans ordre particulier.

Par défaut, `Array.sort()` fonctionne comme suit :

- Le tri est sensible à la casse (*Z* précède *a*).
- Le tri est croissant (*a* précède *b*).
- Le tableau est modifié d'après l'ordre de tri ; si plusieurs éléments comportent des champs de tri identiques, ils sont placés les uns à la suite des autres dans le tableau trié, sans ordre particulier.
- Les champs numériques sont triés comme s'il s'agissait de chaînes, 100 précède donc 99 car « 1 » constitue une valeur de chaîne inférieure à « 9 ».
- Aucune donnée n'est renvoyée.

Si vous souhaitez trier différemment, créez une fonction de tri et transmettez son nom dans le paramètre *`fonctionDeComparaison`*. Par exemple, créez une fonction de tri si vous souhaitez trier par ordre alphabétique par le dernier nom, par ordre croissant, puis par code postal, par ordre décroissant.

Si vous souhaitez définir un ou plusieurs champs sur lesquels procéder au tri, en utilisant le tri par défaut ou le paramètre *`options`*, utilisez `Array.sortOn()`.

Exemple

Usage 1 : l'exemple suivant illustre l'utilisation de `Array.sort` avec et sans valeur définie pour *`option`* :

```
var fruits_array = ["oranges", "pommes", "fraises", "ananas", "cerises"];
trace(fruits_array.join());
fruits_array.sort();
trace(fruits_array.join());
fruits_array.sort(Array.DESCENDING);
trace(fruits_array.join());
```

Le panneau de sortie affiche les résultats suivants :

```
oranges,pommes,fraises,ananas,cerises// tableau d'origine
pommes,cerises,oranges,ananas,fraises// tri par défaut
fraises,ananas,oranges,cerises,pommes// tri décroissant
```

Usage 2 : l'exemple suivant utilise `Array.sort()` avec une fonction de comparaison.

```
var motsDePasse =
  ["maman:séduisante","anna:bague","jacques:magazine","anne:maison","régina:s
  tupide"];
function order (a,b){
  //Les entrées à trier se présentent sous la forme nom:mot de passe
  //Trier en n'utilisant que la partie du nom de l'entrée comme clé.
  var nom1 =a.split(":")[0 ];
  var nom2 =b.split(":")[0 ];
  if (nom1 <nom2) {
    return -1;
  }
  else if (nom1 >nom2) {
return 1;
  }
else
    return 0;
  }
}
trace ("Non trié :");
trace (motsDePasse.join());
```

```
motsDePasse.sort(order);
trace ("Trié :");
trace (motsDePasse.join());
```

Le panneau de sortie affiche les résultats suivants :

```
Non trié :
maman:séduisante,anna:bague,jacques:magazine,anne:maison,régine:stupide
Trié :
anna:bague,anne:maison,jacques:magazine,maman:séduisante,régine:stupide
```

Consultez également

`| (OR au niveau du bit)`, `Array.sortOn()`

Array.sortOn()

Disponibilité

Flash Player 6 ; fonctionnalités supplémentaires ajoutées à Flash Player 7.

Usage

mon_array`.sortOn("`*nomDeChamp*`")`

mon_array`.sortOn("`*nomDeChamp*`", `*option*` | `*option*` |... )`

mon_array`.sortOn( [ "`*nomDeChamp*`", "`*nomDeChamp*`" , ... ] )`

mon_array`.sortOn( [ "`*nomDeChamp*`", "`*nomDeChamp*`" , ... ] , `*option*` | `*option*` |... )`

Remarque : Lorsqu'il y a des crochets ([]), vous devez obligatoirement les inclure dans le code, c'est-à-dire que les crochets ne représentent pas des paramètres facultatifs.

Paramètres

nomDeChamp Une chaîne qui identifie un champ (dans un élément du tableau) à utiliser comme valeur de tri.

option Un ou plusieurs nombres ou chaînes, séparés par l'opérateur `| (OR au niveau du bit)` qui modifie le comportement du tri par rapport au tri par défaut. Les valeurs suivantes sont possibles pour *option* :

- 1 ou `Array.CASEINSENSITIVE`
- 2 ou `Array.DESCENDING`
- 4 ou `Array.UNIQUE`
- 8 ou `Array.RETURNINDEXEDARRAY`
- 16 ou `Array.NUMERIC`

Chacune de ces options est décrite plus en détail dans la section « Description » ci-dessous.

Renvoie

La valeur renvoyée dépend de la définition des paramètres :

- Si vous définissez une valeur de 4 ou Array.UNIQUE pour *option* et que deux éléments ou plus triés comportent des champs de tri identiques, Flash renvoie une valeur de 0 et ne modifie pas le tableau.
- Si vous définissez une valeur de 8 ou Array.RETURNINDEXEDARRAY pour *option*, Flash renvoie un tableau illustrant les résultats du tri et ne modifie pas le tableau.
- Sinon, Flash ne renvoie aucune donnée et modifie le tableau d'après l'ordre du tri.

Description

Méthode : trie les éléments d'un tableau en fonction d'un ou plusieurs champs du tableau. Si vous définissez plusieurs paramètres *nomDeChamp*, le premier champ représente le champ de tri principal, le deuxième représente le champ de tri suivant, etc. Flash trie en fonction des valeurs ASCII (Unicode). Si l'un ou l'autre des éléments comparés ne comprend pas le champ défini dans le paramètre *nomDeChamp*, le champ est considéré `undefined` et les éléments sont placés les uns à la suite des autres dans le tableau trié, sans ordre particulier.

Par défaut, `Array.sortOn()` fonctionne comme suit :

- Le tri est sensible à la casse (*Z* précède *a*).
- Le tri est croissant (*a* précède *b*).
- Le tableau est modifié d'après l'ordre de tri ; si plusieurs éléments comportent des champs de tri identiques, ils sont placés les uns à la suite des autres dans le tableau trié, sans ordre particulier.
- Les champs numériques sont triés comme s'il s'agissait de chaînes, 100 précède donc 99 car « 1 » constitue une valeur de chaîne inférieure à « 9 ».
- Aucune donnée n'est renvoyée.

Vous pouvez utiliser les indicateurs *option* pour remplacer ces valeurs par défaut. Les exemples suivants utilisent différentes formes de l'indicateur *option* à titre d'illustration. Si vous souhaitez trier un tableau simple (un tableau avec un seul champ par exemple) ou si vous souhaitez définir un ordre de tri non supporté par le paramètre *options*, utilisez `Array.sort`.

Pour définir plusieurs indicateurs au format numérique, séparez-les à l'aide de l'opérateur `| (OR au niveau du bit)` ou regroupez les indicateurs. Le code suivant illustre les diverses méthodes permettant de définir un tri décroissant numérique :

```
mon_array.sortOn(unNomDeChamp, 2 | 16);
mon_array.sortOn(unNomDeChamp, 18);
mon_array.sortOn(unNomDeChamp, Array.DESCENDING | Array.NUMERIC);
```

Des conseils de code (consultez *Utilisation des conseils de code*, page 68) sont activés si vous utilisez la forme de chaîne de l'indicateur (`DESCENDING` par exemple) plutôt que la forme numérique (2).

Considérons par exemple le tableau suivant :

```
var mon_array:Array = new Array();
mon_array.push({motDePasse: "Bob", age:29});
mon_array.push({motDePasse: "abcd", age:3});
mon_array.push({motDePasse: "barb", age:35});
mon_array.push({motDePasse: "catchy", age:4});
```

L'application d'un tri par défaut sur le champ du mot de passe génère les résultats suivants :

```
mon_array.sortOn("motDePasse")
// Bob
// abcd
// barb
// catchy
```

L'application d'un tri non sensible à la casse sur le champ du mot de passe génère les résultats suivants :

```
mon_array.sortOn("motDePasse", Array.CASEINSENSITIVE)
// abcd
// barb
// Bob
// catchy
```

L'application d'un tri décroissant non sensible à la casse sur le champ du mot de passe génère les résultats suivants :

```
mon_array.sortOn("motDePasse", 1|2)
// catchy
// Bob
// barb
// abcd
```

L'application d'un tri par défaut sur le champ d'âge génère les résultats suivants :

```
mon_array.sortOn("age")
// 29
// 3
// 35
// 4
```

L'application d'un tri numérique sur le champ d'âge génère les résultats suivants :

```
mon_array.sortOn("age", 16)
// 3
// 4
// 29
// 35
```

L'application d'un tri numérique décroissant sur le champ d'âge génère les résultats suivants :

```
mon_array.sortOn("age", 18)
// 35
// 29
// 4
// 3
```

L'application d'un tri modifie les éléments du tableau comme suit :

```
// Avant le tri
// mon_array[0].age = 29;
// mon_array[1].age = 3;
// mon_array[2].age = 35;
// mon_array[3].age = 4;

// Après un tri ne définissant pas la valeur 8 pour option
mon_array.sortOn("age", Array.NUMERIC);
// mon_array[0].age = 3;
// mon_array[1].age = 4;
// mon_array[2].age = 29;
// mon_array[3].age = 35;
```

L'application d'un tri qui renvoie un tableau d'index ne modifie pas les éléments du tableau :

```
// Avant le tri
// mon_array[0].age = 29;
// mon_array[1].age = 3;
// mon_array[2].age = 35;
// mon_array[3].age = 4;

// Après un tri qui renvoie un tableau contenant des valeurs d'index
// Notez que le tableau d'origine reste inchangé.
// Vous pouvez ensuite utiliser le tableau renvoyé pour afficher les
  informations triées
// sans modifier le tableau d'origine.
var indexArray:Array = mon_array.sortOn("age", Array.RETURNINDEXEDARRAY);
// mon_array[0].age = 29;
// mon_array[1].age = 3;
// mon_array[2].age = 35;
// mon_array[3].age = 4;
```

Exemple

Cet exemple crée un nouveau tableau et le trie en fonction des champs `nom` et `ville` : le premier tri utilise `nom` comme première valeur de tri et `ville` comme la seconde. Le second tri utilise `ville` comme première valeur de tri et `nom` comme la seconde.

```
var enr_array = new Array();
enr_array.push( { nom: "john", ville: "omaha", codePostal: 68144 } );
enr_array.push( { nom: "john", ville: "kansas city", codePostal: 72345 } );
enr_array.push( { nom: "bob", ville: "omaha", codePostal: 94010 } );
for(i=0; i<enr_array.length; i++) {
 trace(enr_array[i].nom + ", " + enr_array[i].ville);
}
// a le résultat suivant
// john, omaha
// john, kansas city
// bob, omaha

rec_array.sortOn( [ "nom", "ville" ]);
for(i=0; i<enr_array.length; i++) {
 trace(enr_array[i].nom + ", " + enr_array[i].ville);
}
// a le résultat suivant
// bob, omaha
// john, kansas city
// john, omaha

rec_array.sortOn( ["ville", "nom" ]);
for(i=0; i<enr_array.length; i++) {
 trace(enr_array[i].nom + ", " + enr_array[i].ville);
}
// a le résultat suivant
// john, kansas city
// bob, omaha
// john, omaha
```

Consultez également

`| (OR au niveau du bit)`, `Array.sort`

Array.splice()

Disponibilité

Flash Player 5.

Usage

```
mon_array.splice(debut, nombreAsupprimer [, valeur0, valeur1...valeurN])
```

Paramètres

debut L'index de l'élément dans le tableau dans lequel commence l'insertion ou la suppression.

nombreAsupprimer Le nombre d'éléments à supprimer. Ce nombre comprend l'élément spécifié dans le paramètre *debut*. Si aucune valeur n'est spécifiée pour *nombreAsupprimer*, la méthode supprime toutes les valeurs en commençant par l'élément *debut* jusqu'au dernier élément du tableau. Si la valeur est 0, aucun élément n'est supprimé.

valeur Paramètre facultatif spécifiant les valeurs à insérer dans le tableau au point d'insertion spécifié dans le paramètre *start*.

Renvoie

Rien.

Description

Méthode : ajoute et supprime des éléments d'un tableau. Cette méthode modifie le tableau sans en faire de copie.

Array.toString()

Disponibilité

Flash Player 5.

Usage

mon_array.toString()

Paramètres

Aucun.

Renvoie

Une chaîne.

Description

Méthode : renvoie une valeur chaîne représentant les éléments de l'objet Array spécifié. Chaque élément du tableau, à compter de l'index 0 et jusqu'à l'index *mon_array*.length-1, est converti en chaîne concaténée séparée par des virgules.

Exemple

L'exemple suivant crée mon_array, le convertit en une chaîne et affiche 1,2,3,4,5 dans le panneau de sortie.

```
mon_array = new Array();
mon_array[0] = 1;
mon_array[1] = 2;
mon_array[2] = 3;
mon_array[3] = 4;
mon_array[4] = 5;
trace(mon_array.toString());
```

Array.unshift()

Disponibilité

Flash Player 5.

Usage

mon_array.unshift(*valeur1,valeur2,...valeurN*)

Paramètres

valeur1,...valeurN Un ou plusieurs nombres, éléments ou variables devant être insérés au début du tableau.

Renvoie

La nouvelle longueur du tableau.

Description

Méthode : ajoute un ou plusieurs éléments au début d'un tableau et renvoie la nouvelle longueur du tableau.

Array()

Disponibilité

Flash Player 6

Usage

```
Array()
Array( [élément0 [, élément1 , élément2,...élémentN ] ])
```

Paramètres

élément Un ou plusieurs éléments à placer dans le tableau.

Renvoie

Un tableau.

Description

Fonction de conversion : crée un nouveau tableau vide ou convertit des éléments spécifiés en un tableau. L'utilisation de cette fonction revient à créer un tableau à l'aide du constructeur de tableau (consultez *Constructeur de l'objet Array*, page 292).

asfunction

Disponibilité

Flash Player 5.

Usage

```
asfunction:fonction,"paramètre"
```

Paramètres

fonction Un identifiant pour une fonction.

paramètre Une chaîne qui est transmise à la fonction nommée dans le paramètre *fonction*.

Renvoie

Rien.

Description

Protocole : un protocole spécial pour les adresses URL dans les champs de texte HTML. Dans les champs de texte HTML, un hyperlien peut être créé avec la balise HTML `A`. L'attribut `HREF` de la balise `A` contient une URL qui peut être pour un protocole standard comme HTTP, HTTPS ou FTP. Le protocole `asfunction` est un protocole supplémentaire spécifique à Flash, selon lequel le lien invoque une fonction ActionScript.

Exemple

Dans cet exemple, la fonction `MaFonc()` est définie dans les trois premières lignes de code. L'objet TextField `monChampDeTexte` est associé à un champ de texte HTML. Le texte « Cliquez-moi ! » est un hyperlien à l'intérieur du champ de texte. La fonction `MaFonc()` est appelée lorsque l'utilisateur clique sur l'hyperlien :

```
function maFonc(param){
  trace("Vous avez cliqué sur moi !" Le paramètre était "+arg);
}
DehtmlText ="<A HREF=\"asfunction:MaFonc,Foo \">Cliquez sur moi!</A>";
```

Lorsque l'hyperlien est activé, le résultat suivant est affiché dans le panneau de sortie :

```
Vous m'avez cliqué !  Le paramètre était unParam
```

Classe Boolean

Disponibilité

Flash Player 5 (est devenu un objet natif dans Flash Player 6, améliorant ainsi les performances de manière significative).

Description

La classe Boolean est un objet enveloppe ayant la même fonctionnalité que l'objet Boolean standard de JavaScript. Utilisez l'objet Boolean pour récupérer le type de données primitif ou la représentation chaîne d'un objet Boolean.

Vous devez utiliser le constructeur `new Boolean()` pour créer un objet Boolean avant d'en appeler les méthodes.

Méthodes de la classe Boolean

Méthode	Description
`Boolean.toString`	Renvoie la représentation chaîne (`"true"` ou `"false"`) de l'objet Boolean.
`Boolean.valueOf()`	Renvoie le type de valeur primitif de l'objet Boolean spécifié.

Constructeur de la classe Boolean

Disponibilité

Flash Player 5.

Usage

```
new Boolean([x])
```

Paramètres

x Toute expression. Ce paramètre est facultatif.

Renvoie

Rien.

Description

Constructeur : crée un objet Boolean. Si vous omettez le paramètre *x*, l'objet Boolean est initialisé avec la valeur `false`. Si vous spécifiez une valeur pour le paramètre *x*, la méthode l'évalue et renvoie le résultat sous forme de valeur booléenne selon les règles de la fonction Boolean().

Exemple

Le code suivant crée un objet booléen vide appelé `maValeurBooléenne`.

```
maValeurBooléenne = new Boolean();
```

Boolean.toString

Disponibilité

Flash Player 5.

Usage

```
maValeurBooléenne.toString()
```

Paramètres

Aucun.

Renvoie

Une valeur booléenne.

Description

Méthode : renvoie la représentation chaîne, (`"true"` ou `"false"`), de l'objet Boolean.

Boolean.valueOf()

Disponibilité

Flash Player 5.

Usage

maValeurBooléenne.valueOf()

Paramètres

Aucun.

Renvoie

Une valeur booléenne.

Description

Méthode : renvoie `true` si le type de valeur primitive de l'objet Boolean défini est vrai, `false` s'il est sur faux.

Exemple

```
var x:Boolean = new Boolean();
trace(x.valueOf());   // faux
x = (6==3+3);
trace(x.valueOf());   // vrai
```

Boolean()

Disponibilité

Flash Player 5 ; comportement mis à jour dans Flash Player 7.

Usage

```
Boolean(expression)
```

Paramètres

expression Une expression à convertir en valeur booléenne.

Renvoie

Une valeur booléenne ou *expression* de la valeur, comme décrit ci-dessous.

Description

Fonction : convertit le paramètre *expression* en valeur booléenne et renvoie une valeur comme suit :

Si *expression* est une valeur booléenne, la valeur renvoyée est *expression*.

Si *expression* est un nombre, la valeur renvoyée est `true` si le nombre n'est pas zéro ; sinon, la valeur renvoyée est `false`.

Si *expression* est une chaîne, la valeur renvoyée est comme suit :

- Dans les fichiers publiés pour Flash Player 6 ou une version antérieure, la chaîne est tout d'abord convertie en un numéro ; la valeur est `true` si le numéro est non zéro, sinon `false`.
- Dans les fichiers publiés pour Flash Player 7 ou une version ultérieure, le résultat est `true` si la chaîne présente une longueur supérieure à zéro ; la valeur est `false` pour une chaîne vide.

Si *expression* est `undefined`, la valeur renvoyée est false.

Si *expression* est un clip ou un objet, la valeur renvoyée est `true`.

Consultez également

Classe Boolean

break

Disponibilité

Flash Player 4.

Usage

```
break
```

Paramètres

Aucun.

Renvoie

Rien.

Description

Instruction : apparaît dans une boucle (`for`, `for..in`, `do while` ou `while`) ou dans un bloc d'instructions associé à un cas particulier dans une action `switch`. L'action `break` indique à Flash d'ignorer le reste du corps de la boucle, de stopper l'action de boucle et d'exécuter l'instruction qui suit l'instruction de boucle. Lors de l'emploi d'une action `break`, l'interprète de Flash ignore le reste des instructions dans ce bloc `case` et passe à la première instruction suivant l'action `switch` le contenant. Utilisez l'action `break` pour sortir d'une série de boucles imbriquées.

Exemple

L'exemple suivant utilise l'action `break` pour sortir d'une boucle sans fin.

```
i = 0;
while (true) {
  if (i >= 100) {
    break;
  }
  i++;
}
```

Consultez également

`break`, `for`, `for..in`, `do while`, `while`, `switch`, `case`

Classe Button

Disponibilité

Flash Player 6.

Description

Tous les symboles bouton d'un fichier SWF sont des occurrences de l'objet Button. Vous pouvez donner un nom d'occurrence à un bouton dans l'inspecteur des propriétés et utiliser les méthodes et propriétés de la classe Button pour le manipuler avec ActionScript. Les noms d'occurrence de bouton sont affichés dans l'explorateur d'animations et dans la boîte de dialogue Insérer un chemin cible du panneau Actions.

La classe Button hérite de la Classe Object.

Méthodes de la classe Button

Méthode	Description
Button.getDepth	Renvoie la profondeur d'une occurrence de bouton.

Propriétés de la classe Button

Propriété	Description
Button._alpha	La valeur de transparence d'une occurrence de bouton.
Button.enabled	Indique si un bouton est actif.
Button._focusrect	Indique si un bouton avec focus est encadré d'un rectangle jaune.
Button._height	La hauteur d'une occurrence de bouton, en pixels.
Button._highquality	Le niveau d'un anti-aliasing appliqué au fichier SWF en cours.
Button.menu	Associe un objet ContextMenu à l'objet button.
Button._name	Le nom d'occurrence d'une occurrence de bouton.
Button._parent	Une référence au clip ou à l'objet contenant le clip ou objet courant.
Button._quality	Indique la qualité de rendu du fichier SWF.
Button._rotation	Le degré de rotation d'une occurrence de bouton.
Button._soundbuftime	Nombre de secondes nécessaires au pré-chargement d'un son.
Button.tabEnabled	Indique si un bouton est inclus dans l'ordre de tabulation automatique.
Button.tabIndex	Indique l'ordre de tabulation d'un objet.
Button._target	Le chemin cible d'une occurrence de bouton.
Button.trackAsMenu	Indique si d'autres boutons peuvent recevoir des événements de relâchement du bouton de la souris.
Button._url	L'URL du fichier SWF créateur de l'occurrence de bouton.
Button.useHandCursor	Indique si le curseur de main est affiché lorsque la souris passe au-dessus d'un bouton.
Button._visible	Une valeur booléenne déterminant si l'occurrence d'un bouton est masquée ou visible.
Button._width	La largeur d'une occurrence de bouton, en pixels.
Button._x	La coordonnée *x* d'une occurrence de bouton.
Button._xmouse	La coordonnée *x* du curseur par rapport à une occurrence de bouton.
Button._xscale	La valeur spécifiant le pourcentage de redimensionnement horizontal d'une occurrence de bouton.
Button._y	La coordonnée *y* d'une occurrence de bouton.
Button._ymouse	La coordonnée *y* du curseur par rapport à une occurrence de bouton.
Button._yscale	La valeur spécifiant le pourcentage de redimensionnement vertical d'une occurrence de bouton.

Gestionnaire d'événement de la classe Button

Gestionnaire d'événement	Description
`Button.onDragOut`	Invoqué lorsqu'une pression est exercée sur le bouton de la souris alors que le pointeur se trouve sur le bouton et qu'il passe ensuite en dehors du bouton.
`Button.onDragOver`	Invoqué lorsque l'utilisateur appuie sur le bouton de la souris et la fait glisser en dehors du bouton, puis revient au-dessus de celui-ci.
`Button.onKeyUp`	Invoqué lorsqu'une touche est relâchée.
`Button.onKillFocus`	Invoqué lorsque le focus est retiré d'un bouton.
`Button.onPress`	Invoqué lorsque le bouton de la souris est enfoncé alors que le pointeur au-dessus d'un bouton.
`Button.onRelease`	Invoqué lorsque le bouton de la souris est relâché alors que le pointeur au-dessus d'un bouton.
`Button.onReleaseOutside`	Invoqué lorsque la souris est relâchée alors que le pointeur se trouve au dehors du bouton après l'enfoncement du bouton pendant que le pointeur est à l'intérieur du bouton.
`Button.onRollOut`	Invoqué lorsque le pointeur passe à l'extérieur d'un bouton.
`Button.onRollOver`	Invoqué lorsque le pointeur de la souris passe au-dessus d'un bouton.
`Button.onSetFocus`	Invoqué lorsqu'un bouton a le focus de saisie et qu'une touche est relâchée.

Button._alpha

Disponibilité

Flash Player 6.

Usage

mon_btn.`_alpha`

Description

Propriété : la valeur de transparence alpha du bouton spécifié par *mon_btn*. Les valeurs valides vont de 0 (transparence complète) à 100 (opacité complète). La valeur par défaut est 100. Les objets d'un bouton avec `_alpha` défini sur 0 sont actifs, même s'ils sont invisibles.

Exemple

Le code suivant définit la propriété `_alpha` d'un bouton nommé `star_btn` à 30 % lorsque l'utilisateur clique sur ce bouton :

```
on(release) {
  star_btn._alpha = 30;
}
```

Consultez également

`MovieClip._alpha`, `TextField._alpha`

Button.enabled

Disponibilité

Flash Player 6.

Usage

mon_btn.enabled

Description

Propriété : une valeur booléenne spécifiant si un bouton est activé. La valeur par défaut est `true`.

Button._focusrect

Disponibilité

Flash Player 6.

Usage

mon_btn._focusrect

Description

Propriété : une valeur booléenne spécifiant si un rectangle jaune apparaît autour du bouton avec focus clavier. Cette propriété peut annuler la propriété `_focusrect` globale.

Button.getDepth

Disponibilité

Flash Player 6.

Usage

mon_btn.getDepth()

Renvoie

Un entier.

Description

Méthode : renvoie la profondeur d'une occurrence de bouton.

Button._height

Disponibilité

Flash Player 6.

Usage

mon_btn._height

Description

Propriété : hauteur du bouton, en pixels.

Exemple

L'exemple de code suivant définit la hauteur et la largeur d'un bouton lorsque l'utilisateur clique sur la souris :

```
mon_btn._width = 200;
mon_btn._height = 200;
```

Button._highquality

Disponibilité

Flash Player 6.

Usage

mon_btn._highquality

Description

Propriété (globale) : spécifie le niveau d'anti-aliasing appliqué au fichier SWF en cours. Spécifiez 2 (qualité maximum) pour appliquer une qualité élevée avec le lissage bitmap toujours actif. Spécifiez 1 (qualité élevée) pour appliquer l'anti-aliasing ; cela permettra de lisser les bitmaps si le fichier SWF ne contient pas d'animation. Spécifiez 0 (faible qualité) pour empêcher l'anti-aliasing.

Consultez également

_quality

Button.menu

Disponibilité

Flash Player 7.

Utilisation

mon_bouton.menu = *menuContextuel*

Paramètres

menuContextuel Un objet ContextMenu.

Description

Propriété : associe le *menuContextuel* de l'objet ContextMenu au bouton *mon_bouton*. La classe ContextMenu vous permet de modifier le menu contextuel qui apparaît quand l'utilisateur clique avec le bouton droit de la souris (Windows) ou enfonce la touche Contrôle (Macintosh) dans Flash Player.

Exemple

L'exemple suivant affecte un objet ContextMenu à un objet Button nommé `enregistrer_btn`. L'objet ContextMenu contient une seule commande de menu (étiqueté « Enregistrer » avec une fonction gestionnaire de rappel appelée `doSave` (masquée).

```
var menu_cm = new ContextMenu();
menu_cm.customItems.push(new ContextMenuItem("Enregistrer...", doSave));
function doSave(menu, obj) {
  // "Enregistrer" code ici
}
enregistrer_btn.menu = menu_cm;
```

Voir aussi

Classe ContextMenu, Classe ContextMenuItem, MovieClip.menu, TextField.menu

Button._name

Disponibilité

Flash Player 6.

Usage

mon_btn._name

Description

Propriété : nom de l'occurrence du bouton spécifié par *mon_btn*.

Button.onDragOut

Disponibilité

Flash Player 6.

Usage

```
mon_btn.onDragOut = function() {
  // vos instructions
}
```

Paramètres

Aucun.

Renvoie

Rien.

Description

Gestionnaire d'événement : invoqué lorsqu'une pression est exercée sur le bouton de la souris alors que le pointeur se trouve sur le bouton et qu'il passe ensuite en dehors du bouton.

Vous devez définir une fonction exécutée lorsque le gestionnaire d'événement est invoqué.

Button.onDragOver

Disponibilité

Flash Player 6.

Usage

```
mon_btn.onDragOver = function() {
  // vos instructions
}
```

Paramètres

Aucun.

Renvoie

Rien.

Description

Gestionnaire d'événement : invoqué lorsque l'utilisateur appuie sur le bouton de la souris et la fait glisser en dehors du bouton, puis revient au-dessus de celui-ci.

Vous devez définir une fonction exécutée lorsque le gestionnaire d'événement est invoqué.

Exemple

L'exemple suivant définit une fonction pour le gestionnaire `onKeyDown` qui envoie une action `trace()` au panneau de sortie :

```
mon_btn.onDragOver = function () {
  trace ("onDragOver appelé");
};
```

Consultez également

`Button.onKeyUp`

Button.onKeyDown

Disponibilité

Flash Player 6.

Usage

```
mon_btn.onKeyDown = function() {
  // vos instructions
}
```

Paramètres

Aucun.

Renvoie

Rien.

Description

Gestionnaire d'événement : invoqué lorsqu'un bouton a le focus clavier et qu'une touche est enfoncée. Le gestionnaire d'événement `onKeyDown` est invoqué sans paramètres. Vous pouvez utiliser `Key.getAscii()` et `Key.getCode()` afin de déterminer quelle touche a été enfoncée.

Vous devez définir une fonction exécutée lorsque le gestionnaire d'événement est invoqué.

Exemple

Dans l'exemple suivant, une fonction envoyant une action `trace()` au panneau de sortie est définie pour le gestionnaire `onKeyDown`.

```
mon_btn.onKeyDown = function () {
  trace ("onKeyDown appelé");
};
```

Consultez également

`Button.onKeyUp`

Button.onKeyUp

Disponibilité

Flash Player 6.

Usage

```
mon_btn.onKeyUp = function() {
  // vos instructions
}
```

Paramètres

Aucun.

Renvoie

Rien.

Description

Gestionnaire d'événement : invoqué lorsqu'un bouton a le focus de saisie et qu'une touche est relâchée. Le gestionnaire d'événement `onKeyUp` est invoqué sans paramètres. Vous pouvez utiliser `Key.getAscii()` et `Key.getCode()` afin de déterminer quelle touche a été enfoncée.

Vous devez définir une fonction exécutée lorsque le gestionnaire d'événement est invoqué.

Exemple

Dans l'exemple suivant, une fonction envoyant une action `trace()` au panneau de sortie est définie pour le gestionnaire `onKeyPress`.

```
mon_btn.onKeyUp = function () {
  trace ("onKeyUp appelé");
};
```

Button.onKillFocus

Disponibilité

Flash Player 6.

Usage

```
mon_btn.onKillFocus = function (nouveauFocus) {
  // vos instructions
}
```

Paramètres

`nouveauFocus` L'objet recevant le focus.

Renvoie

Rien.

Description

Gestionnaire d'événement : invoqué lorsqu'un bouton perd le focus clavier. La méthode `onKillFocus` reçoit un paramètre, *`nouveauFocus`*, qui est un objet représentant le nouvel objet recevant le focus. Si aucun objet ne reçoit le focus, *`nouveauFocus`* contient la valeur `null`.

Button.onPress

Disponibilité

Flash Player 6.

Usage

```
mon_btn.onPress = function() {
  // vos instructions
}
```

Paramètres

Aucun.

Renvoie

Rien.

Description

Gestionnaire d'événement : invoqué lorsqu'un bouton est enfoncé. Vous devez définir une fonction exécutée lorsque le gestionnaire d'événement est invoqué.

Exemple

Dans l'exemple suivant, une fonction envoyant une action `trace()` au panneau de sortie est définie pour le gestionnaire `onPress`.

```
mon_btn.onPress = function () {
  trace ("onPress appelé");
};
```

Button.onRelease

Disponibilité

Flash Player 6.

Usage

```
mon_btn.onRelease = function() {
  // vos instructions
}
```

Paramètres

Aucun.

Renvoie

Rien.

Description

Gestionnaire d'événement : invoqué lorsqu'un bouton est relâché. Vous devez définir une fonction exécutée lorsque le gestionnaire d'événement est invoqué.

Exemple

Dans l'exemple suivant, une fonction envoyant une action `trace()` au panneau de sortie est définie pour le gestionnaire `onRelease`.

```
mon_btn.onRelease = function () {
  trace ("onRelease appelé");
};
```

Button.onReleaseOutside

Disponibilité

Flash Player 6.

Usage

```
mon_btn.onReleaseOutside = function() {
  // vos instructions
}
```

Paramètres

Aucun.

Renvoie

Rien.

Description

Gestionnaire d'événement : après que le pointeur de la souris ait été placé à l'intérieur du bouton et le bouton enfoncé, ce gestionnaire est invoqué au moment où le pointeur se trouve en dehors du bouton et que la souris est relâchée.

Vous devez définir une fonction exécutée lorsque le gestionnaire d'événement est invoqué.

Exemple

Dans l'exemple suivant, une fonction envoyant une action `trace()` au panneau de sortie est définie pour le gestionnaire `onReleaseOutside`.

```
mon_btn.onReleaseOutside = function () {
  trace ("onReleaseOutside appelé");
};
```

Button.onRollOut

Disponibilité

Flash Player 6.

Usage

```
mon_btn.onRollOut = function() {
  // vos instructions
}
```

Paramètres

Aucun.

Renvoie

Rien.

Description

Gestionnaire d'événement : invoqué lorsque le pointeur ne survole plus la zone d'un bouton. Vous devez définir une fonction exécutée lorsque le gestionnaire d'événement est invoqué.

Exemple

Dans l'exemple suivant, une fonction envoyant une action `trace()` au panneau de sortie est définie pour le gestionnaire `onRollOut`.

```
mon_btn.onRollOut = function () {
  trace ("onRollOut appelé");
};
```

Button.onRollOver

Disponibilité

Flash Player 6.

Usage

```
mon_btn.onRollOver = function() {
  // vos instructions
}
```

Paramètres

Aucun.

Renvoie

Rien.

Description

Gestionnaire d'événement : invoqué lorsque le pointeur passe au-dessus d'un bouton. Vous devez définir une fonction exécutée lorsque le gestionnaire d'événement est invoqué.

Exemple

Dans l'exemple suivant, une fonction envoyant une action `trace()` au panneau de sortie est définie pour le gestionnaire `onRollOver`.

```
mon_btn.onRollOver = function () {
  trace ("onRollOver appelé");
};
```

Button.onSetFocus

Disponibilité

Flash Player 6.

Usage

```
mon_btn.onSetFocus = function(ancienFocus){
  // vos instructions
}
```

Paramètres

`ancienFocus` L'objet devant perdre le focus clavier.

Renvoie

Rien.

Description

Gestionnaire d'événement : invoqué lorsqu'un bouton reçoit le focus clavier. Le paramètre *`ancienFocus`* est l'objet perdant le focus. Par exemple, si l'utilisateur appuie sur la touche Tab pour faire passer le focus de saisie d'un champ de texte à un bouton, *`ancienFocus`* contient l'occurrence de champ de texte.

Si aucun objet n'a précédemment reçu le focus, *`ancienFocus`* contient une valeur `null`.

Button._parent

Disponibilité

Flash Player 6.

Usage

mon_btn._parent.*property*

_parent.*propriété*

Description

Propriété : une référence au clip ou à l'objet contenant le clip ou objet courant. L'objet courant est celui contenant le code ActionScript faisant référence à _parent.

Utilisez _parent pour spécifier un chemin relatif aux clips ou objets qui se trouvent au-dessus du clip ou objet actuel. Vous pouvez utiliser _parent pour monter de plusieurs niveaux dans la liste d'affichage, comme dans l'exemple suivant :

```
_parent._parent._alpha = 20;
```

Consultez également

MovieClip._parent, _root, targetPath

Button._quality

Disponibilité

Flash Player 6.

Usage

mon_btn._quality

Description

Propriété (globale) : définit ou récupère la qualité de rendu utilisée pour un fichier SWF. Les polices de périphérique sont toujours aliasées et ne sont donc pas affectées par la propriété _quality.

Remarque : Bien qu'il soit possible de spécifier cette propriété pour un objet Button, il s'agit d'une propriété globale ; vous pouvez tout simplement spécifier sa valeur comme _quality. Pour plus d'informations, consultez _quality.

Button._rotation

Disponibilité

Flash Player 6.

Usage

mon_btn.`_rotation`

Description

Propriété : la rotation du bouton, exprimée en degrés, à partir de son orientation d'origine. Les valeurs de 0 à 180 représentent une rotation dans le sens horaire ; les valeurs de 0 à -180 représentent une rotation dans le sens antihoraire. Les valeurs en dehors de cette plage sont ajoutées à ou soustraites de 360 pour obtenir une valeur comprise dans la plage. Par exemple, l'instruction `mon_btn._rotation = 450` revient à l'instruction `mon_btn._rotation = 90`.

Consultez également

`MovieClip._rotation`, `TextField._rotation`

Button._soundbuftime

Disponibilité

Flash Player 6.

Usage

monBouton.`_soundbuftime`

Description

Propriété (globale) : un entier spécifiant le nombre de secondes de mise en tampon d'un son avant sa lecture en flux continu.

Remarque : Bien qu'il soit possible de spécifier cette propriété pour un objet Button, il s'agit d'une propriété globale ; vous pouvez tout simplement spécifier sa valeur comme `_soundbuftime`. Pour plus d'informations, consultez `_soundbuftime`.

Button.tabEnabled

Disponibilité

Flash Player 6.

Usage

mon_btn.tabEnabled

Description

Propriété : spécifie si *mon_btn* est inclus dans l'ordre de tabulation automatique. Valeur `undefined` par défaut.

Si la propriété `tabEnabled` est `undefined` ou `true`, l'objet est inclus dans l'ordre de tabulation automatique. Si la propriété `tabIndex` est également définie avec une valeur, l'objet est également inclus dans l'ordre de tabulation automatique. Si `tabEnabled` est `false`, l'objet n'est pas inclus dans l'ordre de tabulation automatique ou personnalisé, même si la propriété `tabIndex` est définie.

Consultez également

`Button.tabIndex`, `mon_mc.tabEnabled`, `TextField.tabEnabled`

Button.tabIndex

Disponibilité

Flash Player 6.

Usage

mon_btn.tabIndex

Description

Propriété : permet de personnaliser l'ordre de tabulation des objets d'un fichier SWF. Vous pouvez définir la propriété tabIndex, qui est undefined par défaut, pour une occurrence de bouton, clip ou champ de texte.

Si l'un des objets affichés dans le fichier SWF contient une propriété tabIndex, l'ordre de tabulation automatique est désactivé, et l'ordre de tabulation est alors calculé en fonction des propriétés tabIndex des objets du fichier SWF. L'ordre de tabulation personnalisé n'inclut que les objets possédant des propriétés tabIndex.

La propriété tabIndex peut être un entier non-négatif. Les objets sont placés dans l'ordre correspondant à leurs propriétés tabIndex, dans un ordre croissant. Un objet dont la valeur tabIndex est 1 précède un objet dont la valeur tabIndex est de 2. Si deux objets ont la même valeur tabIndex, celui qui précède l'autre dans l'ordre de tabulation est undefined.

L'ordre de tabulation personnalisé défini par la propriété tabIndex est *flat*. Cela signifie que la relation hiérarchique des objets du fichier SWF n'a pas importance. Tous les objets du fichier SWF possédant des 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 précède l'autre dans l'ordre de tabulation est undefined. Vous ne devriez pas utiliser la même valeur tabIndex pour plusieurs objets.

Consultez également

Button.tabEnabled, MovieClip.tabChildren, mon_mc.tabEnabled, MovieClip.tabIndex, TextField.tabIndex

Button._target

Disponibilité

Flash Player 6.

Usage

mon_btn._target

Description

Propriété (lecture seule) : renvoie le chemin cible de l'occurrence du bouton spécifiée par *mon_btn*.

Consultez également

targetPath

Button.trackAsMenu

Disponibilité

Flash Player 6.

Usage

mon_btn.trackAsMenu

Description

Propriété : une valeur booléenne qui indique si d'autres boutons ou clips peuvent recevoir des événements de relâchement de bouton de souris. Cela vous permet de créer des menus. Vous pouvez définir la propriété `trackAsMenu` pour n'importe quel objet de bouton ou clip. Si la propriété `trackAsMenu` n'est pas définie, le comportement par défaut est `false`.

Vous pouvez changer la propriété `trackAsMenu` à tout moment, le bouton modifié prenant immédiatement le nouveau comportement.

Consultez également

MovieClip.trackAsMenu

Button._url

Disponibilité

Flash Player 6.

Usage

mon_btn._url

Description

Propriété (lecture seule) : récupère l'URL du fichier SWF créateur de ce bouton.

Button.useHandCursor

Disponibilité

Flash Player 6.

Usage

mon_btn.useHandCursor

Description

Propriété : une valeur booléenne qui, lorsqu'elle est définie sur la valeur `true` (par défaut), indique si un curseur en forme de main s'affiche lorsque la souris passe au-dessus d'un bouton. Si cette propriété est définie sur la valeur `false`, le curseur de flèche est utilisé.

Vous pouvez changer la propriété `useHandCursor` à tout moment, le bouton modifié prenant immédiatement le nouveau comportement. La propriété `useHandCursor` peut être lue à partir d'un objet prototype.

Button._visible

Disponibilité

Flash Player 6.

Usage

mon_btn.`_visible`

Description

Propriété : une valeur booléenne indiquant si le bouton spécifié par *mon_btn* est visible. Les boutons qui ne sont pas visibles (propriété `_visible` définie sur `false`) sont désactivés.

Consultez également

`MovieClip._visible`, `TextField._visible`

Button._width

Disponibilité

Flash Player 6.

Usage

mon_btn.`_width`

Description

Propriété : la largeur du bouton, en pixels.

Exemple

L'exemple suivant définit les propriétés de hauteur et de largeur d'un bouton.

```
mon_btn._width=200;
mon_btn._height=200;
```

Consultez également

`MovieClip._width`

Button._x

Disponibilité

Flash Player 6.

Usage

mon_btn.*_x*

Description

Propriété : un entier définissant la coordonnée *x* du bouton par rapport aux coordonnées locales du clip parent. Si un bouton se trouve dans le scénario principal, son système de coordonnées fait référence au coin supérieur gauche de la scène, sous la forme (0, 0). Si le bouton se trouve dans un clip qui a subi des transformations, le bouton est dans le système de coordonnées local du clip le contenant. Donc, pour un clip ayant pivoté de 90 degrés dans le sens inverse des aiguilles d'une montre, le bouton inclus hérite d'un système de coordonnées qui a pivoté de 90 degrés dans le sens inverse des aiguilles d'une montre. Les coordonnées du bouton font référence à la position du point d'alignement.

Consultez également

Button._xscale, Button._y, Button._yscale

Button._xmouse

Disponibilité

Flash Player 6.

Usage

mon_btn._xmouse

Description

Propriété (lecture seule) : renvoie la coordonnée *x* de la position de la souris par rapport au bouton.

Consultez également

Button._ymouse

Button._xscale

Disponibilité

Flash Player 6.

Usage

mon_btn.`_xscale`

Description

Propriété : le redimensionnement horizontal du bouton appliqué à partir de son point d'alignement, exprimé sous forme de pourcentage. Le point d'alignement par défaut est (0,0).

Le redimensionnement du système de coordonnées locales affecte les paramètres de propriété `_x` et `_y`, définis en pixels. Par exemple, si le clip parent est dimensionné à 50 %, la définition de la propriété `_x` déplace un objet du bouton de la moitié du nombre de pixels d'un fichier SWF à 100 %.

Consultez également

`Button._x`, `Button._y`, `Button._yscale`

Button._y

Disponibilité

Flash Player 6.

Usage

mon_btn.`_y`

Description

Propriété : 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 fait référence au coin supérieur gauche de la scène, sous la forme (0, 0). Si le bouton se trouve dans un autre clip qui a subi des transformations, le bouton est dans le système de coordonnées local du clip le contenant. Donc, pour un clip ayant pivoté de 90 degrés dans le sens inverse des aiguilles d'une montre, le bouton inclus hérite d'un système de coordonnées qui a pivoté de 90 degrés dans le sens inverse des aiguilles d'une montre. Les coordonnées du bouton font référence à la position du point d'alignement.

Consultez également

`Button._x`, `Button._xscale`, `Button._yscale`

Button._ymouse

Disponibilité

Flash Player 6.

Usage

mon_btn._ymouse

Description

Propriété (lecture seule) : indique la coordonnée *y* de la position de la souris par rapport au bouton.

Consultez également

Button._xmouse

Button._yscale

Disponibilité

Flash Player 6.

Usage

mon_btn._yscale

Description

Propriété : le redimensionnement vertical du bouton appliqué à partir de son point d'alignement, exprimé sous forme de pourcentage. Le point d'alignement par défaut est (0,0).

Consultez également

Button._y, Button._x, Button._xscale

call()

Disponibilité

Flash Player 4. Cette action est déconseillée dans Flash 5 ; Macromedia recommande l'utilisation de l'action `function`.

Usage

```
call(image)
```

Paramètres

image L'étiquette ou le numéro d'une image du scénario.

Renvoie

Rien.

Description

Action à éviter : exécute le script dans l'image appelée sans déplacer la tête de lecture jusqu'à cette image. Les variables locales n'existent plus après l'exécution du script.

Consultez également

`function`, `Function.call`

Classe Camera

Disponibilité

Flash Player 6.

Description

La classe Camera est principalement conçue pour une utilisation avec Macromedia Flash Communication Server, mais elle peut être utilisée de façon limitée sans le serveur.

La classe Camera permet de capturer une vidéo à partir d'une caméra vidéo connectée à un ordinateur qui exécute Macromedia Flash Player, par exemple, pour contrôler une vidéo issue d'une webcam associée à votre système local. (Flash fournit les mêmes fonctionnalités audio. Pour plus d'informations, consultez l'entrée Classe Microphone.)

Pour créer ou référencer un objet Camera, utilisez `Camera.get()`.

Méthodes de la classe Camera

Méthode	Description
`Camera.get()`	Renvoie un objet Camera par défaut ou spécifié, ou `null` si la caméra n'est pas disponible.
`Camera.setMode()`	Définit les aspects du mode de capture de la caméra, tels que la hauteur, la largeur et les images par seconde.
`Camera.setMotionLevel()`	Spécifie les mouvements nécessaires pour invoquer `Camera.onActivity(true)` et le temps qui doit s'écouler sans mouvement avant l'invocation de `Camera.onActivity(false)`.
`Camera.setQuality()`	Un entier qui spécifie la bande passante maximale que la vidéo actuelle sortante peut utiliser, en octets par seconde.

Propriétés de la classe Camera

Propriété (lecture seule)	Description
`Camera.activityLevel`	La quantité de mouvements que la caméra détecte.
`Camera.bandwidth`	La bande passante maximale que la vidéo actuelle sortante peut utiliser, en octets.
`Camera.currentFps`	Le taux auquel la caméra capture les données, en images par seconde.
`Camera.fps`	Le taux auquel vous souhaitez que la caméra capture les données, en images par seconde.
`Camera.height`	La hauteur de la capture actuelle, en pixels.
`Camera.index`	L'index de la caméra, comme indiqué dans le tableau renvoyé par `Camera.names`.
`Camera.motionLevel`	La quantité de mouvements nécessaire pour invoquer `Camera.onActivity(true)`.
`Camera.motionTimeOut`	Le nombre de millisecondes entre le moment où la caméra arrête de détecter les mouvements et le moment où `Camera.onActivity(false)` est invoqué.
`Camera.muted`	Une valeur booléenne qui spécifie si l'utilisateur a autorisé ou refusé l'accès à la caméra.
`Camera.name`	Le nom de la caméra tel qu'il est spécifié par le matériel de la caméra.
`Camera.names`	Propriété de classe : un tableau de chaînes indiquant les noms de tous les appareils de capture vidéo disponibles, y compris les cartes vidéo et les caméras.
`Camera.quality`	Un entier spécifiant le niveau nécessaire de qualité d'image, tel que déterminé par le montant de compression appliqué à chaque image vidéo.
`Camera.width`	La largeur de la capture actuelle, en pixels.

Gestionnaires d'événement de la classe Camera

Gestionnaire d'événement	Description
`Camera.onActivity`	Invoqué lorsque la caméra commence ou arrête de détecter des mouvements.
`Camera.onStatus`	Invoqué lorsque l'utilisateur autorise ou refuse l'accès à la caméra.

Constructeur de la classe Camera

Pour plus d'informations, consultez `Camera.get()`.

Camera.activityLevel

Disponibilité

Flash Player 6.

Usage

active_cam.activityLevel

Description

Propriété (lecture seule) : une valeur numérique spécifiant la quantité de mouvements que la caméra détecte. Les valeurs vont de 0 (aucun mouvement détecté) à 100 (nombreux mouvements détectés). La valeur de cette propriété peut vous aider à déterminer si vous devez transmettre un paramètre à `Camera.setMotionLevel()`.

Si la caméra est disponible mais qu'elle n'est pas encore utilisée car `Video.attachVideo()` n'a pas encore été appelé, cette propriété est définie sur -1.

Si vous transmettez en continu une vidéo locale non compressée, cette propriété n'est définie que si vous avez affecté une fonction au gestionnaire d'événement `Camera.onActivity`. Dans le cas contraire, elle est undefined.

Consultez également

`Camera.motionLevel`, `Camera.setMotionLevel()`

Camera.bandwidth

Disponibilité

Flash Player 6.

Usage

active_cam.bandwidth

Description

Propriété (lecture seule) : un entier qui spécifie la bande passante maximale que la vidéo actuelle sortante peut utiliser, en octets par seconde. Une valeur 0 signifie que la vidéo Flash peut utiliser autant de bande passante que nécessaire pour conserver la qualité d'image souhaitée.

Pour définir cette propriété, utilisez Camera.setQuality().

Exemple

L'exemple suivant charge un autre fichier SWF si la bande passante de la caméra est de 32 kilo-octets ou plus.

```
if(maCam.bandwidth >= 32768){
  loadMovie("splat.swf",_root.hiddenvar);
}
```

Consultez également

Camera.setQuality()

Camera.currentFps

Disponibilité

Flash Player 6.

Usage

active_cam.currentFps

Description

Propriété (lecture seule) : le taux auquel la caméra capture les données, en images par seconde. Cette propriété ne peut pas être définie. Cependant, vous pouvez utiliser la méthode Camera.setMode() pour définir une propriété associée —Camera.fps—qui spécifie la cadence maximale à laquelle vous souhaitez que la caméra capture les données.

Consultez également

Camera.fps, Camera.setMode()

Camera.fps

Disponibilité

Flash Player 6.

Usage

active_cam.fps

Description

Propriété (lecture seule) : le taux maximum auquel vous souhaitez que la caméra capture les données, en images par seconde. Le taux maximal possible dépend des capacités de la caméra : si la caméra ne supporte pas la valeur que vous avez définie ici, la cadence ne sera pas atteinte.

- Pour définir la valeur souhaitée de cette propriété, utilisez `Camera.setMode()`.
- Pour définir le taux auquel la caméra capture actuellement les données, utilisez la propriété `Camera.currentFps`.

Exemple

L'exemple suivant définit le taux fps de la caméra active, `maCam.fps` à la valeur fournie par le champ de texte de l'utilisateur `this.config.txt_fps`.

```
if (this.config.txt_fps != undefined) {
    maCam.setMode(maCam.width, maCam.height, this.config.txt_fps, false);
}
```

Remarque : La fonction setMode ne garantit pas la définition fps demandée ; elle définit le fps demandé ou le fps le plus rapide disponible.

Consultez également

`Camera.currentFps`, `Camera.setMode()`

Camera.get()

Disponibilité

Flash Player 6.

Usage

```
Camera.get([index])
```

Remarque : La syntaxe correcte est `Camera.get()`. Pour affecter l'objet Camera à une variable, utilisez une syntaxe telle que `active_cam=Camera.get()`.

Paramètres

index Un entier de base zéro facultatif qui spécifie la caméra à récupérer, comme l'indique le tableau renvoyé par la propriété `Camera.names`. Pour récupérer la caméra par défaut (ce qui est recommandé pour la plupart des applications), omettez ce paramètre.

Renvoie

- Si *index* n'est pas spécifié, cette méthode renvoie une référence à la caméra par défaut ou, si elle est utilisée par une autre application, à la première caméra disponible. Si plus d'une caméra est installée, l'utilisateur peut spécifier la caméra par défaut dans le panneau Camera Settings de Flash Player. Si aucune caméra n'est disponible ou installée, la méthode renvoie `null`.
- Si *index* est spécifié, cette méthode renvoie une référence demandée ou `null` si elle n'est pas disponible.

Description

Méthode : renvoie une référence à un objet Camera pour la capture de vidéo. Pour commencer à capturer la vidéo, vous devez lier l'objet Camera à un objet Video (consultez `Video.attachVideo()`).

Contrairement aux objets créés à l'aide du constructeur `new`, les appels multiples à `Camera.get()` font référence à la même caméra. Ainsi, si votre script contient les lignes `first_cam = Camera.get()` et `second_cam = Camera.get()`, `first_cam` et `second_cam` font référence à la même caméra (par défaut).

De manière générale, vous ne devez pas transmettre une valeur pour *index* ; utilisez tout simplement `Camera.get()` pour renvoyer une référence à la caméra par défaut. A l'aide du panneau Camera Settings (abordé plus loin dans cette même section), l'utilisateur peut spécifier la caméra par défaut que Flash doit utiliser. Si vous transmettez une valeur pour *index*, vous pourrez tenter de faire référence à une caméra autre que la caméra préférée de l'utilisateur. *index* ne sera utilisé que rarement, par exemple, si votre application capture de la vidéo à partir de deux caméras en même temps.

Lorsqu'un fichier SWF essaie d'accéder à la caméra renvoyée par `Camera.get()`, Flash Player affiche le panneau de contrôle de l'accès qui permet à l'utilisateur de choisir s'il doit autoriser ou refuser l'accès à la caméra. (Vérifiez que la scène mesure au moins 215 x 138 pixels ; c'est la taille minimale requise par Flash pour afficher la boîte de dialogue.)

Lorsque l'utilisateur répond à cette boîte de dialogue, le gestionnaire d'événement `Camera.onStatus` renvoie un objet information indiquant la réponse. Afin de déterminer si l'utilisateur a refusé ou accepté l'accès à la caméra sans traiter le gestionnaire d'événement, utilisez la propriété `Camera.muted`.

Pour spécifier des paramètres de contrôle de l'accès permanents pour un domaine particulier, l'utilisateur peut également cliquer avec le bouton droit de la souris (Windows) ou en appuyant sur la touche Contrôle (Macintosh) pendant la lecture d'un fichier SWF, puis choisir Paramètres, ouvrir le panneau de contrôle de l'accès et sélectionner Mémoriser.

Vous ne pouvez pas utiliser ActionScript pour définir la valeur Autoriser ou Refuser pour un utilisateur mais vous pouvez afficher le panneau de contrôle de l'accès pour l'utilisateur en utilisant `System.showSettings(0)`. Si l'utilisateur sélectionne Mémoriser, Flash Player ferme le panneau de contrôle de l'accès pour les animations de ce domaine.

Si `Camera.get` renvoie `null`, soit la caméra est utilisée par une autre application, soit aucune caméra n'est installée sur le système. Pour déterminer s'il existe des caméras installées, utilisez `Camera.names.length`. Pour afficher le panneau Paramètres de la caméra Flash Player permettant à l'utilisateur de choisir la caméra devant être référencée par `Camera.get()`, utilisez `System.showSettings(3)`.

L'opération consistant à analyser le matériel afin de détecter la présence de caméras prend du temps. Si Flash trouve au moins une caméra, il n'analyse plus le matériel pendant toute la durée de l'occurrence du lecteur. Cependant, si Flash ne trouve pas de caméra, il analyse le matériel chaque fois que `Camera.get` est appelé. Cette étape s'avère utile si un utilisateur a oublié de connecter la caméra ; si votre fichier SWF fournit un bouton Nouvelle tentative qui appelle `Camera.get`, Flash peut trouver la caméra sans que l'utilisateur n'ait à redémarrer le fichier SWF.

Exemple

L'exemple suivant capture et affiche la vidéo localement dans un objet Video nommé `ma_video` sur la scène.

```
var ma_cam = Camera.get();
ma_video.attachVideo(maCam);
```

Consultez également

`Camera.index`, `Camera.muted`, `Camera.names`, `Camera.onStatus`, `Camera.setMode()`, `System.showSettings()`, `Video.attachVideo()`

Camera.height

Disponibilité

Flash Player 6.

Usage

`cam_active``.height`

Description

Propriété (lecture seule) : la hauteur de capture actuelle, en pixels. Pour définir la valeur de cette propriété, utilisez `Camera.setMode()`.

Exemple

La ligne de code suivante met à jour un champ de texte dans l'interface utilisateur avec la valeur de la hauteur actuelle.

```
mon_txt._height = myCam.height;
```

Veuillez également consulter l'exemple pour `Camera.setMode()`.

Consultez également

`Camera.setMode()`, `Camera.width`

Camera.index

Disponibilité

Flash Player 6.

Usage

active_cam.index

Description

Propriété (lecture seule) : un entier basé sur zéro qui spécifie l'index de la caméra, comme mentionné dans le tableau renvoyé par Camera.names.

Exemple

L'exemple suivant récupère la caméra ayant la valeur index.

```
ma_cam = Camera.get(index);
```

Consultez également

Camera.get(), Camera.names

Camera.motionLevel

Disponibilité

Flash Player 6.

Usage

active_cam.motionLevel

Description

Propriété (lecture seule) : une valeur numérique qui spécifie la quantité de mouvements nécessaire pour invoquer Camera.onActivity(true). Les valeurs valides sont comprises entre 0 et 100. La valeur par défaut est 50.

La vidéo peut être affichée, quelle que soit la valeur de la propriété motionLevel. Pour plus d'informations, consultez Camera.setMotionLevel().

Consultez également

Camera.activityLevel, Camera.onActivity, Camera.onStatus, Camera.setMotionLevel()

Camera.motionTimeOut

Disponibilité

Flash Player 6.

Usage

active_cam.motionTimeOut

Description

Propriété (lecture seule) : le nombre de millisecondes entre le moment où la caméra arrête de détecter les mouvements et le moment où Camera.onActivity(false) est invoqué. La valeur par défaut est 2 000 (2 secondes).

Pour définir cette valeur, utilisez Camera.setMotionLevel().

Exemple

L'exemple suivant définit le nombre de millisecondes entre le moment où la caméra arrête de détecter les mouvements et le moment où Camera.onActivity(false) est invoqué à 1 000 millisecondes ou une seconde.

```
if(ma_cam.motionTimeOut >= 1000){
  ma_cam.setMotionLevel(myCam.motionLevel, 1000);
}
```

Consultez également

Camera.onActivity, Camera.setMotionLevel()

Camera.muted

Disponibilité

Flash Player 6.

Usage

active_cam.muted

Description

Propriété (lecture seule) : une valeur booléenne spécifiant si l'utilisateur a refusé l'accès à la caméra (true) ou permis l'accès à la caméra (false) dans le panneau des paramètres de contrôle de l'accès de Flash Player. Lorsque cette valeur change, Camera.onStatus est invoqué. Pour plus d'informations, consultez Camera.get().

Consultez également

Camera.get(), Camera.onStatus

Camera.name

Disponibilité

Flash Player 6.

Usage

active_cam.name

Description

Propriété (lecture seule) : une chaîne qui spécifie le nom de la caméra actuelle, tel que renvoyé par le matériel de la caméra.

Exemple

L'exemple suivant affiche le nom de la caméra par défaut dans le panneau de sortie. Sous Windows, ce nom est le même que le nom de l'appareil apparaissant dans la page de propriétés des scanners et des caméras.

```
ma_cam = Camera.get();
trace("Le nom de la caméra est : " + ma_cam.name);
```

Consultez également

Camera.get(), Camera.names

Camera.names

Disponibilité

Flash Player 6.

Usage

```
Camera.names
```

Remarque : La syntaxe correcte est `Camera.names`. Pour affecter la valeur de renvoi à une variable, utilisez une syntaxe telle que *`cam_array`* `= Camera.names`. Pour déterminer le nom de la caméra actuelle, utilisez *`active_cam`*`.name`.

Description

Propriété de classe (lecture seule) : récupère un tableau de chaînes répertoriant le nom de toutes les caméras disponibles sans afficher le panneau des paramètres de contrôle de l'accès de Flash Player. Ce tableau se comporte comme tout autre tableau ActionScript : il fournit implicitement l'index basé sur zéro de chaque caméra et le nombre de caméras dans le système (au moyen de `Camera.names.length`). Pour plus d'informations, consultez l'entrée Classe Array.

L'appel de la propriété `Camera.names` requiert un examen attentif du matériel : c'est pourquoi la construction du tableau peut prendre plusieurs secondes. Dans la plupart des cas, vous pouvez tout simplement utiliser la caméra par défaut.

Exemple

L'exemple suivant utilise la caméra par défaut à moins qu'il n'existe plusieurs caméras disponibles, auquel cas l'utilisateur peut choisir la caméra par défaut.

```
cam_array = Camera.names;
if (tableau_cam.length == 1){
ma_cam = Camera.get();
}
else {
  System.showSettings(3);
ma_cam = Camera.get();
}
```

Consultez également

`Camera.get()`, `Camera.index`, `Camera.name`

Camera.onActivity

Disponibilité

Flash Player 6.

Usage

```
active_cam.onActivity = function(activité) {
  // vos instructions
}
```

Paramètres

activité Une valeur booléenne définie sur `true` lorsque la caméra commence à détecter les mouvements et sur `false` lorsqu'elle s'arrête.

Renvoie

Rien.

Description

Gestionnaire d'événement : invoqué lorsque la caméra commence ou arrête de détecter le mouvement. Si vous souhaitez répondre à ce gestionnaire d'événement, vous devez créer une fonction afin de traiter sa valeur *activité*.

Pour spécifier la quantité de mouvements nécessaire pour invoquer `Camera.onActivity(true)`, ainsi que le temps devant s'écouler sans aucune activité avant l'invocation de `Camera.onActivity(false)`, utilisez `Camera.setMotionLevel()`.

Exemple

L'exemple suivant affiche `true` ou `false` dans le panneau de sortie lorsque la caméra commence ou arrête de détecter les mouvements.

```
// Suppose qu'un objet Video nommé "monObjetVidéo" est sur la scène
ma_cam = Camera.get();
monObjetVidéo.attachVideo(ma_cam);
ma_cam.setMotionLevel(10, 500);
ma_cam.onActivity = function(mode)
{
  trace(mode);
}
```

Consultez également

`Camera.onActivity`, `Camera.setMotionLevel()`

Camera.onStatus

Disponibilité

Flash Player 6.

Usage

```
active_cam.onStatus = function(objetInfo) {
  // vos instructions
}
```

Paramètres

objetInfo Un paramètre défini selon le message d'état.

Renvoie

Rien.

Description

Gestionnaire d'événement : invoqué lorsque l'utilisateur autorise ou refuse l'accès à la caméra. Si vous souhaitez répondre à ce gestionnaire d'événement, vous devez créer une fonction afin de traiter l'objet information généré par la caméra.

Lorsqu'un fichier SWF essaie d'accéder à la caméra, Flash Player affiche le panneau de contrôle de l'accès qui permet à l'utilisateur de choisir s'il doit autoriser ou refuser l'accès à la caméra.

- Si l'utilisateur autorise l'accès, la propriété `Camera.muted` est définie sur `false` ; ce gestionnaire est invoqué avec un objet information dont la propriété `code` est `"Camera.Unmuted"` et dont la propriété `niveau` est `"Etat"`.
- Si l'utilisateur refuse l'accès, la propriété `Camera.muted` est définie sur `true` ; ce gestionnaire est invoqué avec un objet information dont la propriété `code` est `"Camera.Muted"` et dont la propriété `niveau` est `"Etat"`.

Afin de déterminer si l'utilisateur a refusé ou accepté l'accès à la caméra sans traiter le gestionnaire d'événement, utilisez la propriété `Camera.muted`.

Remarque : Si l'utilisateur choisit d'autoriser ou de refuser définitivement l'accès à tous les fichiers SWF à partir d'un domaine spécifique, ce gestionnaire n'est pas invoqué pour les fichiers SWF de ce domaine, à moins que l'utilisateur ne modifie ultérieurement les paramètres de contrôle de l'accès. Pour plus d'informations, consultez `Camera.get()`.

Exemple

Le gestionnaire d'événement suivant affiche un message dès que l'utilisateur autorise ou refuse l'accès à la caméra.

```
maCam = Camera.get();
monObjetVidéo.attachVideo(maCam);
maCam.onStatus = function(infoMsg) {
  if(infoMsg.code == "Caméra.Désactivée"){
    trace("L'utilisateur refuse l'accès à la caméra");
  }
else
    trace("L'utilisateur autorise l'accès à la caméra");
}
// Change la valeur Autoriser ou Refuser pour invoquer la fonction
System.showSettings(0);
```

Consultez également

`Camera.get()`, `Camera.muted`

Camera.quality

Disponibilité

Flash Player 6.

Usage

`active_cam``.quality`

Description

Propriété (lecture seule) : un entier spécifiant le niveau nécessaire de qualité d'image, tel que déterminé par le montant de compression appliqué à chaque image vidéo. Les valeurs de qualité valides sont comprises entre 1 (qualité minimale, compression maximale) et 100 (qualité maximale, pas de compression). La valeur par défaut est 0, qui signifie que la qualité d'image peut varier en fonction des besoins afin d'éviter un excès de bande passante disponible.

Consultez également

`Camera.setQuality()`

Camera.setMode()

Disponibilité

Flash Player 6.

Usage

active_cam.setMode(*largeur*, *hauteur*, *fps* [,T*aillefavor*])

Paramètres

largeur La largeur de capture demandée, en pixels. La valeur par défaut est 160.

hauteur La hauteur de capture demandée, en pixels. La valeur par défaut est 120.

fps Le taux requis auquel la caméra doit capturer les données, en images par seconde. La valeur par défaut est 15.

Taillefavor (facultatif) : une valeur booléenne qui spécifie comment manipuler la largeur, la hauteur et la cadence si la caméra n'a aucun mode natif correspondant aux conditions requises. La valeur par défaut est `true` : cela signifie que la conservation de la taille de la capture est favorisée ; l'utilisation de ce paramètre sélectionne le mode qui correspond le mieux aux valeurs *largeur* et *hauteur*, bien qu'une telle action affecte négativement les performances en réduisant la cadence. Afin d'optimiser la cadence aux dépens de la hauteur et de la largeur, définissez le paramètre `Taillefavor` sur *false*.

Renvoie

Rien.

Description

Méthode : définit le mode de capture de la caméra sur le mode natif qui correspond le mieux aux conditions requises. Si la caméra n'a pas de mode natif qui correspond à tous les paramètres définis, Flash sélectionne un mode capture qui correspond le mieux au mode requis. Cette manipulation peut nécessiter le rognage de l'image et l'omission d'images.

Par défaut, Flash omet les images de sorte que la taille de l'image soit conservée. Pour minimiser le nombre d'images omises, même s'il faut pour cela réduire la taille de l'image, définissez le paramètre `Taillefavor` sur *false*.

Lors du choix du mode natif, Flash tente de conserver les proportions demandées si possible. Par exemple, si vous communiquez la commande *active_cam*.setMode(400,400,30) et que les valeurs largeur et hauteur maximales disponibles sur la caméra sont 320 et 288, Flash définit à la fois la largeur et la hauteur sur 288 ; en définissant ces propriétés sur la même valeur, Flash conserve les proportions 1:1 que vous avez demandées.

Pour déterminer les valeurs affectées à ces propriétés dès que Flash a sélectionné le mode qui correspond le mieux aux valeurs requises, utilisez `Camera.width`, `Camera.height` et `Camera.fps`.

Exemple

L'exemple suivant définit la largeur, la hauteur et fps en fonction des entrées de l'utilisateur, s'il clique sur le bouton. Le paramètre facultatif *Taillefavor* n'est pas inclus car la valeur par défaut `true` fournit les paramètres les plus proches des préférences de l'utilisateur sans sacrifier la qualité d'image. Cependant, il se peut que fps soit sacrifié. L'interface utilisateur est alors mise à jour avec les nouveaux paramètres.

```
on (press)
{
  // Définit la largeur, la hauteur et fps en fonction des entrées de
  l'utilisateur.
  _root.myCam.setMode(txt_width, my_txt._height, txt_fps);

  // Mettre à jour le champ de texte de l'utilisateur avec les nouveaux
  paramètres.
  _root.txt_width = maCam.width;
_root.txt_height = maCam.height;
  _root.txt_fps = maCam.fps;
}
```

Consultez également

`Camera.currentFps`, `Camera.fps`, `Camera.height`, `Camera.width`

Camera.setMotionLevel()

Disponibilité

Flash Player 6.

Usage

active_cam.setMotionLevel(*sensibilité* [, *timeout*])

Paramètres

sensibilité Une valeur numérique qui spécifie la quantité de mouvements nécessaire pour invoquer Camera.onActivity(true). Les valeurs valides sont comprises entre 0 et 100. La valeur par défaut est 50.

timeout Un paramètre numérique facultatif qui spécifie le nombre de millisecondes devant s'écouler sans activité avant que Flash ne considère que l'activité est stoppée et invoque le gestionnaire d'événement Camera.onActivity(false). La valeur par défaut est 2 000 (2 secondes).

Renvoie

Rien.

Description

Méthode : spécifie les mouvements nécessaires pour invoquer Camera.onActivity(true). Définit de façon facultative le nombre de millisecondes devant s'écouler sans activité avant que Flash ne considère que les mouvements sont stoppés et invoque Camera.onActivity(false).

Remarque : La vidéo peut être affichée, quelle que soit la valeur du paramètre *sensibilité*. Ce paramètre ne détermine que le moment où et dans quelle circonstance Camera.onActivity est invoqué ; il ne détermine pas si la vidéo est en cours de capture ou d'affichage.

- Afin d'empêcher la caméra de détecter tout mouvement, définissez le paramètre de *sensibilité* sur 100 ; Camera.onActivity n'est jamais invoqué. (Vous n'utiliserez probablement cette valeur que pour des tests, par exemple, pour désactiver temporairement les actions définies devant se produire lorsque Camera.onActivity est invoqué.)
- Pour déterminer le nombre de mouvements en cours que la caméra détecte, utilisez la propriété Camera.activityLevel.

Les valeurs de sensibilité aux mouvements correspondent directement aux valeurs d'activité. L'absence totale de mouvements correspond à une valeur d'activité de 0. Un mouvement constant correspond à une valeur d'activité de 100. Votre valeur d'activité est inférieure à votre valeur de sensibilité de mouvement lorsque vous êtes immobile ; en mouvement, les valeurs d'activité dépassent fréquemment la valeur de sensibilité de mouvement.

Cette méthode a le même objectif que Microphone.setSilenceLevel() ; les deux méthodes permettent de déterminer à quel moment le gestionnaire d'événement onActivity doit être invoqué. Cependant, ces méthodes ont un impact très différent sur la publication en flux continus :

- `Microphone.setSilenceLevel()` est conçu pour optimiser la bande passante. Lorsqu'un flux continu audio est considéré comme silencieux, aucune donnée audio n'est envoyée. Un message unique est envoyé, indiquant le début du silence.
- `Camera.setMotionLevel()` est conçu pour détecter les mouvements et n'affecte pas l'utilisation de la bande passante. Même lorsqu'un flux continu vidéo ne détecte pas de mouvement, la vidéo est envoyée.

Exemple

L'exemple suivant envoie des messages au panneau de sortie lorsque l'activité de vidéo commence ou s'arrête. Modifiez la valeur de la sensibilité de mouvement définie à 30 (en l'augmentant ou en la diminuant) afin de voir comment les différentes valeurs affectent la détection du mouvement.

```
// Suppose qu'un objet Video nommé "monObjetVidéo" est sur la scène
c = Camera.get();
x = 0;
function motion(mode)
{
  trace(x + ": " + mode);
  x++;
}
c.onActivity = function(mode) {motion(mode);};
c.setMotionLevel(30, 500);
monObjetVidéo.attachVideo(c);
```

Consultez également

`Camera.activityLevel`, `Camera.motionLevel`, `Camera.motionTimeOut`, `Camera.onActivity`

Camera.setQuality()

Disponibilité

Flash Player 6.

Usage

active_cam.setQuality(*bandePassante*, *qualitéDimage*)

Paramètres

bandePassante Un entier qui spécifie la bande passante maximale que la vidéo actuelle sortante peut utiliser, en octets par seconde. Pour indiquer que la vidéo Flash peut utiliser autant de bande passante que nécessaire pour conserver la valeur *qualitéDimage*, définissez *bandePassante* sur la valeur 0. La valeur par défaut est 16384.

qualitéDimage Un entier spécifiant le niveau nécessaire de qualité d'image, tel que déterminé par le montant de compression appliqué à chaque image vidéo. Les valeurs de qualité acceptables vont de 1 (qualité minimale, compression maximale) à 100 (qualité maximale, pas de compression). Pour indiquer que la qualité d'image peut varier en fonction des besoins et pour éviter une bande passante excessive, définissez *qualitéDimage* sur la valeur 0. La valeur par défaut est 0.

Renvoie

Rien.

Description

Méthode : définit la bande passante maximale par seconde ou la qualité d'image nécessaire pour la vidéo sortante actuelle. En principe, cette méthode n'est applicable que si vous transmettez la vidéo à l'aide de Flash Communication Server.

Utilisez cette méthode pour spécifier quel élément de la vidéo sortante est le plus important pour votre application (utilisation de la bande passante ou qualité d'image).

- Pour indiquer que l'utilisation de la bande passante est prioritaire, définissez une valeur pour *bandePassante* et paramétrez *qualitéDimage* sur 0. Flash transmettra ainsi la vidéo à la meilleure qualité possible dans la limite de la bande passante spécifiée. Si nécessaire, Flash réduira la qualité d'image afin d'éviter de dépasser la bande passante spécifiée. En général, la qualité décroît à mesure que le mouvement s'accroît.
- Pour indiquer que la qualité est prioritaire, définissez *bandePassante* sur 0 et indiquez une valeur numérique pour *qualitéDimage*. Flash utilise autant de bande passante que nécessaire afin de conserver la qualité spécifiée. Si nécessaire, Flash réduira la cadence pour conserver la qualité d'image. En général, l'utilisation de bande passante augmente à mesure que le mouvement augmente.
- Pour spécifier que la bande passante et la qualité sont aussi importantes l'une que l'autre, transmettez des valeurs numériques aux deux paramètres. Flash transmet une vidéo qui atteint la qualité requise et qui ne dépasse pas la bande passante spécifiée. Si nécessaire, Flash réduira la cadence afin de conserver la qualité d'image sans dépasser la bande passante spécifiée.

Exemple

Les exemples suivantes illustrent comment utiliser cette méthode afin de contrôler l'utilisation de la bande passante et la qualité d'image.

```
// Assurer qu'un maximum de 8 192 (8 K/secondes) est utilisé pour l'envoi de la
   vidéo
active_cam.setQuality(8192,0);

// Assurer qu'un maximum de 8 192 (8 K/secondes) est utilisé pour l'envoi de la
   vidéo
// avec une qualité minimale de 50
active_cam.setQuality(8192,50);

// Assurer une qualité minimale de 50, quelle que soit la bande passante
   utilisée
active_cam.setQuality(0,50);
```

Consultez également

`Camera.bandwidth`, `Camera.quality`

Camera.width

Disponibilité

Flash Player 6.

Usage

`active_cam``.width`

Description

Propriété (lecture seule) : la largeur de capture actuelle, en pixels. Pour définir la valeur souhaitée de cette propriété, utilisez `Camera.setMode()`.

Exemple

La ligne de code suivante met à jour un champ de texte dans l'interface utilisateur avec la valeur de la hauteur actuelle.

```
monChampDeTexte.text=maCam.width;
```

Veuillez également consulter l'exemple pour `Camera.setMode()`.

Consultez également

`Camera.height`

case

Disponibilité

Flash Player 4.

Usage

```
case expression: instructions
```

Paramètres

expression Toute expression.

instructions Toute instruction.

Renvoie

Rien.

Description

Instruction : définit une condition pour l'action `switch`. Les instructions du paramètre *instructions* sont exécutées si le paramètre *expression* qui suit le mot-clé `case` est égal au paramètre *expression* de l'action `switch` avec une égalité stricte (===).

L'action `case` utilisée en dehors d'une instruction `switch` produit une erreur et la compilation du script échouera si elle est utilisée comme telle.

Consultez également

`break`, `default`, `=== (égalité stricte)`, `switch`

chr

Disponibilité

Flash Player 4. Cette fonction est déconseillée dans Flash 5 ; utilisez plutôt `String.fromCharCode()`.

Usage

```
chr(nombre)
```

Paramètres

nombre Un nombre de code ASCII.

Renvoie

Rien.

Description

Fonction de chaîne : convertit les codes ASCII en caractères.

Exemple

L'exemple suivant convertit le nombre 65 en la lettre *A* et l'affecte à la variable `maVar`.

```
maVar = chr(65);
```

Consultez également

`String.fromCharCode()`

class

Disponibilité

Flash Player 6.

Usage

```
[dynamic] class nomDeLaClasse  [ extends superClasse ]
                [ implements nomDinterface [, nomDinterface... ] ]
{
  // définition de la classe ici
}
```

Remarque : Pour utiliser ce mot-clé, vous devez définir ActionScript 2.0 et Flash Player 6 ou version ultérieure dans l'onglet Flash de la boîte de dialogue Paramètres de publication de votre fichier FLA. Ce mot-clé n'est supporté que lorsqu'il est utilisé dans des fichiers de scripts externes et non dans des scripts écrits dans le panneau Actions.

Paramètres

nomDeLaClasse Le nom pleinement qualifié de la classe.

superClasse Facultatif : le nom de la classe que *nomDeLaClasse* étend (qui hérite de).

nomDinterface Facultatif : le nom de l'interface dont *nomDeLaClasse* doit implémenter les méthodes.

Description

Instruction : définit une classe personnalisée, qui permet d'instancier des objets qui partagent les méthodes et propriétés définies. Par exemple, si vous développez un système de suivi de factures, vous pouvez créer une classe facture définissant les méthodes et les propriétés de chaque facture. Vous devez alors utiliser la commande `new invoice()` pour créer des objets facture.

Le nom de la classe doit être identique au nom du fichier externe contenant la classe. Par exemple, si vous nommez une classe *Etudiant*, le fichier définissant la classe doit être nommé *Etudiant.as*.

Le nom de classe doit être pleinement qualifié dans le fichier dans lequel il est déclaré, c'est-à-dire qu'il doit référencer le répertoire dans lequel il est stocké. Par exemple, pour créer une classe nommée ClasseRequise stockée dans le répertoire myClasses/education/curriculum, vous devez déclarer la classe dans le fichier ClasseRequise.as comme suit :

```
classe myClasses.education.curriculum.ClasseRequise {
}
```

Pour cela, il est préférable de planifier la structure de votre répertoire avant de créer les classes. Si vous décidez de déplacer les fichiers de classe après les avoir créés, vous devrez modifier les instructions de déclaration de la classe pour référencer leur nouvel emplacement.

Vous ne pouvez pas imbriquer les définitions de classe ; autrement dit, vous ne pouvez pas définir de classe supplémentaire au sein d'une définition de classe.

Pour signaler les objets pouvant ajouter des propriétés dynamiques et y accéder au moment de l'exécution, spécifiez le mot-clé `dynamic` avant l'instruction de la classe. Pour créer des classes basées sur les interfaces, utilisez le mot-clé `implements`. Pour créer des sous-classes, utilisez le mot-clé `extends`. (Une classe peut étendre une classe seulement, mais peut implémenter plusieurs interfaces.) Vous pouvez utiliser `implements` et `extends` dans une seule et même instruction.

```
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    // pas OK
```

Pour plus d'informations, consultez *Création et utilisation de classes*, page 169.

Exemple

L'exemple suivant crée une classe nommée Plant. Son constructeur prend deux paramètres.

```
// Nom de fichier Plant.as
class Plant {
  // Définissez les noms et les types de propriété
  var typeDeFeuille:String;
  var saisonDeFloraison:String;
  // La ligne qui suit est le constructeur
  // parce qu'elle a le même nom que la classe
  function Plant (param_typeDeFeuille:String, param_saisonDeFloraison:String)
  {
    // Affectez des valeurs transmises aux propriétés lorsque le nouvel objet
  Plant est créé
    typeDeFeuille = param_typeDeFeuille;
    saisonDeFloraison = param_saisonDeFloraison;
  }
  // Créez des méthodes pour renvoyer les valeurs de propriété, parce qu'il
  s'agit de la meilleure façon,
  // comparé au référencement direct d'une propriété d'une classe
  function getLeafType():String {return le typeDeFeuille};
  function getBloomSeason():String {return la saisonDeFloraison};
}
```

Dans un fichier de script externe ou dans le panneau Actions, utilisez l'opérateur `new` pour créer un objet Plant.

```
var pin:Plant = new Plant("Persistant","N/A");
// Confirmez que les paramètres ont été transmis correctement
trace(pin.getLeafType());
trace(pin.getBloomSeason());
```

Consultez également

`dynamic`, `extends`, `implements`, `import`, `interface`, `new`, `Object.registerClass()`

clearInterval()

Disponibilité

Flash Player 6.

Usage

```
clearInterval( idDintervalle )
```

Paramètres

idDintervalle Un objet renvoyé à partir d'un appel à `setInterval()`.

Renvoie

Rien.

Description

Fonction : annule un appel à `setInterval()`.

Exemple

L'exemple suivant définir d'abord, puis annule un appel d'intervalle :

```
function callback() {
        trace("intervalle appelé");
}
var idDintervalle;
idDintervalle = setInterval( callback, 1000 );

// plus tard
clearInterval( idDintervalle );
```

Consultez également

`setInterval()`

Classe Color

Disponibilité

Flash Player 5.

Description

La classe Color vous permet de définir la valeur de la couleur RVB et de récupérer les valeurs une fois qu'elles ont été définies.

Vous devez utiliser le constructeur `new Color()` pour créer un objet Color avant d'en appeler les méthodes.

Méthodes de la classe Color

Méthode	Description
`Color.getRGB()`	Renvoie la valeur numérique RVB définie par le dernier appel `setRGB()`.
`Color.getTransform()`	Renvoie les informations de transformation définies par le dernier appel `setTransform()`.
`Color.setRGB()`	Définit la représentation hexadécimale de la valeur RVB pour un objet Color.
`Color.setTransform()`	Définit la transformation de couleur pour un objet Color.

Constructeur de la classe Color

Disponibilité

Flash Player 5.

Usage

```
new Color(cible)
```

Paramètres

`cible` Le nom d'occurrence d'un clip.

Renvoie

Rien.

Description

Constructeur : crée un objet Color pour le clip spécifié par le paramètre *`cible`*. Vous pouvez alors utiliser les méthodes de cet objet Color pour changer la couleur du clip cible.

Exemple

L'exemple suivant crée un objet Color nommé `ma_color` pour le clip `monClip_mc` et définit sa valeur RVB :

```
ma_color = new Color(mon_mc);
ma_color.setRGB(0xff9933);
```

Color.getRGB()

Disponibilité

Flash Player 5.

Usage

ma_color.getRGB()

Paramètres

Aucun.

Renvoie

Un nombre représentant la valeur numérique RVB pour la couleur spécifiée.

Description

Méthode : renvoie les valeurs numériques définies par le dernier appel `setRGB()`.

Exemple

Le code suivant récupère la valeur RVB de l'objet couleur `ma_color`, la convertit en une chaîne hexadécimale et l'affecte à la variable `value`.

```
value = ma_color.getRGB().toString(16);
```

Consultez également

`Color.setRGB()`

Color.getTransform()

Disponibilité

Flash Player 5.

Usage

ma_color.getTransform()

Paramètres

Aucun.

Renvoie

Un objet dont les propriétés contiennent le décalage actuel et les valeurs de pourcentage de la couleur spécifiée.

Description

Méthode : renvoie la valeur de transformation définie par le dernier appel de `Color.setTransform()`.

Consultez également

`Color.setTransform()`

Color.setRGB()

Disponibilité

Flash Player 5.

Usage

ma_color.setRGB(0x*RRGGBB*)

Paramètres

0x*RRGGBB* La couleur hexadécimale ou RVB à définir. *RR*, *VV* et *BB* consistent chacun en deux chiffres hexadécimaux spécifiant le décalage de chaque composante de couleur. Le 0x indique au compilateur ActionScript que le nombre est une valeur hexadécimale.

Description

Méthode : spécifie une couleur RVB pour un objet Color. L'appel de cette méthode annule les paramètres Color.setTransform() précédents.

Renvoie

Rien.

Exemple

Cet exemple définit la valeur de couleur RVB pour le clip mon_mc. Pour voir ce code fonctionner, placez un clip sur la scène avec le nom d'occurrence mon_mc. Placez ensuite le code suivant sur l'image 1 du scénario principal et choisissez Contrôle > Tester l'animation.

```
ma_color = new Color(mon_mc);
ma_color.setRGB(0x993366);
```

Consultez également

Color.setTransform()

Color.setTransform()

Disponibilité

Flash Player 5.

Usage

ma_color.setTransform(*objetDeTransformationDeCouleur*)

Paramètres

objetDeTransformationDeCouleur Un objet créé à l'aide du constructeur `new Object`. Cette occurrence de Classe Object doit avoir les propriétés suivantes, qui spécifient les valeurs de transformation de couleur : `ra`, `rb`, `ga`, `gb`, `ba`, `bb`, `aa`, `ab`. Ces propriétés sont expliquées ci-dessous.

Renvoie

Rien.

Description

Méthode : définit les informations de transformation de couleur pour un objet Color. Le paramètre *objetDeTransformationDeCouleur* est un objet générique que vous créez à partir du constructeur `new Object`. Il a des paramètres spécifiant les valeurs de pourcentage et de décalage pour les composantes rouge, verte, bleue et alpha (transparence) d'une couleur, entrées suivant le format *0xRRVVBBAA*.

Les paramètres pour un objet de transformation de couleur correspondent aux paramètres définis dans la boîte de dialogue Effet avancé et sont définis comme suit :

- *ra* est le pourcentage de la composante rouge (-100 à 100).
- *rb* est le décalage de la composante rouge (-255 à 255).
- *ga* est le pourcentage de la composante verte (-100 à 100).
- *gb* est le décalage de la composante verte (-255 à 255).
- *ba* est le pourcentage de la composante bleue (-100 à 100).
- *bb* est le décalage de la composante bleue (-255 à 255).
- *aa* est le pourcentage pour alpha (-100 à 100).
- *ab* est le décalage pour alpha (-255 à 255).

Vous créez un paramètre *objetDeTransformationDeCouleur* comme suit :

```
maTransformationDeCouleur = new Object();
maTransformationDeCouleur.ra = 50;
maTransformationDeCouleur.rb = 244;
maTransformationDeCouleur.ga = 40;
maTransformationDeCouleur.gb = 112;
maTransformationDeCouleur.ba = 12;
maTransformationDeCouleur.bb = 90;
maTransformationDeCouleur.aa = 40;
maTransformationDeCouleur.ab = 70;
```

Vous pouvez également utiliser la syntaxe suivante pour créer un paramètre *objetDeTransformationDeCouleur* :

```
maTransformationDeCouleur = { ra: '50', rb: '244', ga: '40', gb: '112', ba:
  '12', bb: '90', aa: '40', ab: '70'}
```

Exemple

Cet exemple crée un nouvel objet Color pour un fichier SWF cible, crée un objet générique appelé `maTransformationDeCouleur` avec les propriétés définies ci-dessus et utilise la méthode `setTransform()` pour transmettre *`objetDeTransformationDeCouleur`* à un objet Color. Pour utiliser ce code dans un document Flash (FLA), placez-le sur l'image 1 du scénario principal et placez un clip sur la scène avec le nom d'occurrence `mon_mc`, comme dans le code suivant :

```
// Créez un objet Color appelé ma_color pour la cible mon_mc
ma_color = new Color(mon_mc);
// Créez un objet de transformation de couleur appelé
  maTransformationDeCouleur
// avec l'objet générique Object
maTransformationDeCouleur = new Object();
// définir les valeurs de maTransformationDeCouleur
maTransformationDeCouleur = { ra: '50', rb: '244', ga: '40', gb: '112', ba:
  '12', bb: '90', aa: '40', ab: '70'};
// associer l'objet de transformation de couleur à l'objet Color
// créé pour mon_mc
ma_color.setTransform(maTransformationDeCouleur);
```

Classe ContextMenu

Disponibilité

Flash Player 7.

Description

La classe ContextMenu fournit un contrôle d'exécution des éléments du menu contextuel de Flash Player, qui apparaît lorsqu'un utilisateur clique avec le bouton droit (Windows) ou appuie sur la touche Contrôle (Macintosh) dans Flash Player. Vous pouvez utiliser les méthodes et les propriétés de la classe ContextMenu pour ajouter des éléments de menu personnalisés, contrôler l'affichage des éléments du menu contextuel intégré (par exemple Zoom avant et Imprimer) ou créer des copies de menus.

Vous pouvez associer un objet ContextMenu à un bouton spécifique, un clip ou un objet de champ de texte. Pour ce faire, utilisez la propriété `menu` de la classe Button, MovieClip ou TextField. Pour plus d'informations sur la propriété `menu`, consultez `Button.menu`, `MovieClip.menu` et `TextField.menu`.

Pour ajouter de nouveaux éléments à un objet ContextMenu, vous créez un objet ContextMenuItem, puis vous ajoutez cet objet au tableau `ContextMenu.customItems`. Pour plus d'informations sur la création d'éléments de menu contextuel, consultez l'entrée Classe ContextMenuItem.

Flash Player dispose de trois types de menus contextuels : le menu Standard (qui apparaît lorsque vous cliquez avec le bouton droit de la souris dans Flash Player), le menu Edition (qui apparaît lorsque vous cliquez avec le bouton droit de la souris sur un champ de texte sélectionnable ou modifiable) et le menu Erreur (qui apparaît lorsque le chargement d'un fichier SWF a échoué dans Flash Player.) Seuls les menus Standard et Edition peuvent être modifiés à l'aide de la classe ContextMenu.

Les éléments de menu personnalisés apparaissent toujours en haut du menu contextuel Flash Player, au-dessus de tout élément de menu intégré ; une barre de séparation distingue les éléments de menu intégrés des éléments personnalisés. Un menu contextuel peut contenir jusqu'à 15 éléments personnalisés.

Vous devez utiliser le constructeur `new ContextMenu()` pour créer un objet ContextMenu avant d'en appeler les méthodes.

Méthodes de la classe ContextMenu

Méthode	Description
`ContextMenu.copy()`	Renvoie une copie de l'objet ContextMenu spécifié.
`ContextMenu.hideBuiltInItems()`	Masque la plupart des éléments intégrés dans le menu contextuel de Flash Player.

Propriétés de la classe ContextMenu

Propriété	Description
`ContextMenu.builtInItems`	Un objet dont les membres correspondent aux éléments intégrés du menu contextuel.
`ContextMenu.customItems`	Un tableau, non défini par défaut, qui contient des objets ContextMenuItem.

Gestionnaires d'événement de la classe ContextMenu

Propriété	Description
`ContextMenu.onSelect`	Invoqué avec l'affichage du menu.

Constructeur de la classe ContextMenu

Disponibilité

Flash Player 7.

Usage

```
new ContextMenu ([fonctionDeRappel])
```

Paramètres

`fonctionDeRappel` Une référence à une fonction appelée lorsque l'utilisateur clique avec le bouton droit ou appuie sur la touche Contrôle, avant l'affichage du menu. Ce paramètre est facultatif.

Renvoie

Rien.

Description

Constructeur : crée un nouvel objet ContextMenu. Facultativement, à la création de l'objet, vous pouvez définir un identifiant pour un gestionnaire d'événements. La fonction spécifiée est appelée lorsque l'utilisateur invoque le menu contextuel, mais *avant* l'affichage de ce dernier. Cette information est utile pour personnaliser le contenu des menus en fonction de l'état de l'application ou du type d'objet (clip, champ de texte ou bouton) sur lequel l'utilisateur clique avec le bouton droit ou pour lequel il appuie sur la touche Contrôle. (Pour obtenir un exemple de création d'un gestionnaire d'événements, consultez `ContextMenu.onSelect`.)

Exemple

L'exemple suivant masque tous les objets intégrés dans le menu contextuel. (Cependant, les éléments Paramètres et A propos de apparaissent encore, car il est impossible de les désactiver.)

```
var nouveauMenu = new ContextMenu();
nouveauMenu.hideBuiltInItems();
_root.menu = nouveauMenu;
```

Dans cet exemple, le gestionnaire d'événements spécifié, `menuHandler`, active ou désactive un élément du menu personnalisé (à l'aide du tableau `ContextMenu.customItems`) en fonction de la valeur de la variable booléenne nommée `showItem`. Si cette dernière est `false`, l'élément du menu personnalisé est désactivé ; dans le cas contraire, il est activé.

```
var afficherElement = false; // Changez la valeur en true pour visualiser
  l'effet
mon_cm = new ContextMenu(gestionnaireDeMenu);
mon_cm.customItems.push(new ContextMenuItem("Bonjour", itemHandler));
function menuHandler(obj, menuObj) {
  if (showItem == false) {
    menuObj.customItems[0].enabled = false;
  } else {
    menuObj.customItems[0].enabled = true;
  }
}
function itemHandler(obj, item) {
}
_root.menu = mon_cm;
```

Consultez également

`Button.menu`, `ContextMenu.onSelect`, `ContextMenu.customItems`, `ContextMenu.hideBuiltInItems()`, `MovieClip.menu`, `TextField.menu`

ContextMenu.builtInItems

Disponibilité

Flash Player 7.

Usage

mon_cm.builtInItems

Description

Propriété : un objet ayant les propriétés booléennes suivantes : `save`, `zoom`, `quality`, `play`, `loop`, `rewind`, `forward_back` et `print`. La définition de ces variables sur `false` ôte les éléments correspondants du menu de l'objet ContextMenu spécifié. Ces propriétés sont énumérables et sont définies sur `true` par défaut.

Exemple

Dans cet exemple, les éléments intégrés Quality et Print sont désactivés pour l'objet ContextMenu `mon_cm`, qui est lié au scénario principal du fichier SWF.

```
var mon_cm = new ContextMenu ();
mon_cm.builtInItems.quality=false;
mon_cm.builtInItems.print=false;
_root.menu = mon_cm;
```

Dans l'exemple suivant, une boucle `for..in` énumère tous les noms et les valeurs des éléments intégrés du menu de l'objet ContextMenu, `mon_cm`.

```
mon_cm = new ContextMenu();
for(eachProp dans mon_cm.builtInItems) {
  var nomProp = eachProp;
  var valeurProp = mon_cm.builtInItems[nomProp];
  trace(nomProp + ": " + valeurProp;
}
```

ContextMenu.copy()

Disponibilité

Flash Player 7.

Usage

mon_cm.copy()

Paramètres

Aucun.

Renvoie

Un objet ContextMenu.

Description

Méthode : crée une copie de l'objet ContextMenu spécifié. La copie hérite de toutes les propriétés de l'objet du menu original.

Exemple

Cet exemple crée une copie de l'objet ContextMenu nommé `mon_cm` dont les éléments intégrés sont masqués et ajoute un élément du menu avec le texte « Enregistrer... ». Il crée ensuite une copie de `mon_cm` et l'affecte à la variable `clone_cm`, qui hérite de toutes les propriétés du menu original.

```
mon_cm = new ContextMenu();
mon_cm.hideBuiltInItems();
mon_cm.customItems.push(new ContextMenuItem("Enregistrer...", saveHandler);
function saveHandler (obj, menuItem) {
  saveDocument(); // fonction personnalisée (masquée)
}
clone_cm = mon_cm.copy();
```

ContextMenu.customItems

Disponibilité

Flash Player 7.

Usage

mon_cm.customItems

Description

Propriété : un tableau d'objets ContextMenuItem. Chaque objet du tableau représente un menu contextuel que vous avez défini. Utilisez cette propriété pour ajouter, ôter ou modifier ces éléments personnalisés du menu.

Pour ajouter des nouveaux éléments de menu, vous devez d'abord créer un nouvel objet ContextMenuItem, puis l'ajouter au tableau *menu_mc*.customItems (à l'aide de Array.push(), par exemple). Pour plus d'informations sur la création d'éléments de menu, consultez l'entrée Classe ContextMenuItem.

Exemple

L'exemple suivant crée un nouvel élément de menu personnalisé nommé élémentDuMenu_cm avec la légende « Envoyer un courriel » et un gestionnaire de rappel nommé emailHandler (masqué). Le nouvel élément de menu est ensuite ajouté à l'objet ContextMenu, mon_cm, à l'aide du tableau customItems. Enfin, le nouveau menu est attaché à un clip nommé courriel_mc.

```
var mon_cm = new ContextMenu();
var élémentDuMenu_cm = new ContextMenuItem("Envoyer un courriel",
  emailHandler);
mon_cm.customItems.push(élémentDuMenu_cm);
email_mc.menu = mon_cm;
```

Consultez également

Button.menu, Classe ContextMenu, MovieClip.menu, TextField.menu

ContextMenu.hideBuiltInItems()

Disponibilité

Flash Player 7.

Usage

```
mon_cm.hideBuiltInItems();
```

Paramètres

Aucun.

Renvoie

Rien.

Description

Méthode : masque tous les éléments intégrés du menu (excepté Paramètres) dans l'objet ContextMenu spécifié. Si Flash Debug Player est en cours d'exécution, l'élément de menu Débogage apparaît, mais il est grisé pour les fichiers SWF dont le débogage à distance est désactivé.

Cette méthode masque seulement les éléments du menu qui apparaissent dans le menu contextuel standard ; elle n'affecte pas les éléments apparaissant dans les menus d'édition ou d'erreur. Pour plus d'informations sur les différents types de menus, veuillez vous référer à l'entrée Classe ContextMenu.

Cette méthode consiste à définir tous les membres booléens de *mon_cm*.`builtInItems` sur `false`. Vous pouvez choisir de rendre visible un élément intégré en définissant son membre correspondant dans *mon_cm*.`builtInItems` sur `true` (comme dans l'exemple suivant).

Exemple

L'exemple suivant crée un nouvel objet ContextMenu nommé `mon_cm` dont les éléments intégrés du menu sont masqués, excepté pour Print. L'objet du menu est lié au scénario principal.

```
mon_cm = new ContextMenu();
mon_cm.hideBuiltInItems();
mon_cm.builtInItems.print = true;
_root.menu = mon_cm;
```

ContextMenu.onSelect

Disponibilité

Flash Player 7.

Usage

```
mon_cm.onSelect = function (item:Object, item_menu:ContextMenu) {
  // votre code ici
}
```

Paramètres

élément Une référence à l'objet (clip, bouton ou champ de texte sélectionnable) que le pointeur de la souris survolait lorsque le menu contextuel de Flash Player a été invoqué et dont la propriété `menu` est définie en tant qu'objet ContextMenu valide.

menuElément Une référence à l'objet ContextMenu affecté à la propriété `menu` de l'*objet*.

Renvoie

Rien.

Description

Gestionnaire d'événement : appelé quand un utilisateur invoque le menu contextuel de Flash Player, mais avant l'affichage de ce dernier. Vous pouvez ainsi personnaliser le contenu du menu contextuel en fonction de l'état actuel de l'application.

Vous pouvez également spécifier le gestionnaire de rappel pour un objet ContextMenu quand vous construisez un nouvel objet ContextMenu. Pour plus d'informations, consultez l'entrée Classe ContextMenu.

Exemple

L'exemple suivant détermine sur quel type d'objet le menu contextuel a été invoqué.

```
mon_cm = new ContextMenu();
menuHandler = function (obj:Object, menu:ContextMenu) {
  if(obj instanceof MovieClip) {
    trace("Clip : " + obj);
  }
  if(obj instanceof TextField) {
    trace("Champ de texte : " + obj);
  }
  if(obj instanceof TextField) {
    trace("Bouton : " + obj);
  }
}
mon_cm.onSelect = menuHandler;
```

Classe ContextMenuItem

Disponibilité

Flash Player 7.

Description

Utilisez la classe ContextMenuItem afin de créer des éléments de menus personnalisés à afficher dans le menu contextuel de Flash Player. Chaque objet ContextMenuItem a une légende (texte) affichée dans le menu contextuel, ainsi qu'un gestionnaire de rappel (une fonction) invoqué quand l'élément du menu sélectionné. Pour ajouter un nouvel élément à un menu contextuel, vous devez l'ajouter dans le tableau `customItems` de l'objet ContextMenu.

Vous pouvez activer ou désactiver des éléments spécifiques du menu, rendre les éléments visibles ou invisibles, ou encore changer la légende ou le gestionnaire de rappel associé à un élément du menu.

Les éléments de menu personnalisés apparaissent en haut du menu contextuel, au-dessus des éléments intégrés. Une barre de séparation distingue toujours les éléments personnalisés des éléments intégrés. Vous pouvez ajouter jusqu'à 15 éléments personnalisés au menu contextuel de Flash Player. Chaque élément doit contenir au moins un caractère visible ; les caractères de contrôle, sauts de lignes et autres espaces blancs sont ignorés. Aucun élément ne peut contenir plus de 100 caractères. Les éléments identiques aux éléments intégrés ou à d'autres éléments personnalisés sont ignorés, que l'élément correspondant soit visible ou non. Les éléments du menu sont comparés sans tenir compte de la casse, de la ponctuation et des espaces blancs.

Aucun des mots suivants ne peut apparaître dans un élément personnalisé : *Macromedia, Flash Player*, *Settings*.

Méthode de la classe ContextMenuItem

Méthode	Description
`ContextMenuItem.copy()`	Renvoie une copie de l'objet ContextMenuItem spécifié.

Propriétés de la classe ContextMenuItem

Propriété	Description
`ContextMenuItem.caption`	Spécifie le texte affiché dans l'élément du menu.
`ContextMenuItem.enabled`	Spécifie si l'élément du menu est activé ou désactivé.
`ContextMenuItem.separatorBefore`	Spécifie si une barre de séparation doit apparaître au-dessus de l'élément du menu.
`ContextMenuItem.visible`	Spécifie si l'élément du menu est visible ou non.

Gestionnaire d'événement de la classe ContextMenuItem

Gestionnaire d'événement	Description
`ContextMenuItem.onSelect`	Invoqué quand l'élément du menu est sélectionné.

Constructeur de la classe ContextMenuItem

Disponibilité

Flash Player 7.

Usage

```
new ContextMenuItem(légende, fonctionDeRappel, [séparateurAvant,] [activé,]
  [visible] ] ] )
```

Paramètres

légende Une chaîne qui spécifie le texte associé à l'élément du menu.

fonctionDeRappel Une fonction que vous définissez et qui est appelée lorsque l'élément de menu est sélectionné.

séparateurAvant Une valeur booléenne qui indique si une barre de séparation doit apparaître au-dessus de l'élément du menu dans le menu contextuel. Ce paramètre est facultatif ; sa valeur par défaut est `false`.

activé Une valeur booléenne qui indique si l'élément du menu est activé ou désactivé dans le menu contextuel. Ce paramètre est facultatif ; sa valeur par défaut est `true`.

visible Une valeur booléenne qui indique si l'élément du menu est visible ou invisible. Ce paramètre est facultatif ; sa valeur par défaut est `true`.

Renvoie

Rien.

Description

Constructeur : crée un nouvel objet ContextMenuItem qui peut être ajouté au tableau `ContextMenu.customItems`.

Exemple

Cet exemple ajoute les éléments du menu Démarrer et Arrêter, séparés par une barre, à l'objet ContextMenu `mon_cm`. La fonction `startHandler()` est appelée quand Démarrer est sélectionné dans le menu contextuel ; `stopHandler()` est appelée quand Arrêter est sélectionné. L'objet ContextMenu du menu est appliqué au scénario principal.

```
mon_cm = new ContextMenu();
mon_cm.customItems.push(new ContextMenuItem("Démarrer", startHandler));
mon_cm.customItems.push(new ContextMenuItem("Arrêter", stopHandler, true));
function itemHandler(obj, item) {
  trace("Arrêt...");
}
function itemHandler(obj, item) {
  trace("Démarrage...");
}
_root.menu = mon_cm;
```

ContextMenuItem.caption

Disponibilité

Flash Player 7.

Usage

élémentDuMenu_cmi.caption

Description

Propriété : une chaîne qui spécifie la légende de l'élément du menu (texte) affichée dans le menu contextuel.

Exemple

Cet exemple affiche la légende de l'élément du menu sélectionné (Mettre le jeu en pause) dans le panneau de sortie.

```
mon_cm = new ContextMenu();
élémentDuMenu_cmi = new ContextMenuItem("Mettre le jeu en pause", onPause);
mon_cm.customItems.
function onPause(obj, menuItem) {
  trace("Vous choisissez : " + menuItem.caption);
}
```

ContextMenuItem.copy()

Disponibilité

Flash Player 7.

Usage

élémentDuMenu_cmi.copy();

Renvoie

Un objet ContextMenuItem.

Description

Méthode : crée et renvoie une copie de l'objet ContextMenuItem spécifié. La copie inclut toutes les propriétés de l'objet original.

Exemple

Cet exemple crée un nouvel objet ContextMenuItem nommé `original_cmi` avec le texte Pause en tant que légende et un gestionnaire d'événement défini sur la fonction `onPause`. L'exemple crée ensuite une copie de l'objet ContextMenuItem et l'affecte à la variable `copier_cmi`.

```
original_cmi = new ContextMenuItem("Pause", onPause);
function onPause(obj, menu) {
  _root.stop();
}
original_cmi.visible = false;
copy_cmi = orig_cmi.copy();
```

ContextMenuItem.enabled

Disponibilité

Flash Player 7.

Usage

```
élémentDuMenu_cmi.enabled
```

Description

Propriété : une valeur booléenne qui indique si l'élément du menu spécifié est activé ou désactivé. Par défaut, cette propriété est `true`.

Exemple

L'exemple suivant crée un nouvel élément du menu contextuel, puis désactive l'élément du menu.

```
var saveMenuItem = new ContextMenuItem("Enregistrer...", doSave);
saveMenuItem.enabled = false;
```

ContextMenuItem.onSelect

Disponibilité

Flash Player 7.

Usage

```
élémentDuMenu_cmi.onSelect = function (obj, élémentDuMenu) {
  // vos instructions
}
```

Paramètres

obj Une référence au clip (ou scénario), au bouton ou au champ de texte modifiable sur lequel l'utilisateur a cliqué avec le bouton droit de la souris ou en appuyant sur la touche Contrôle.

élémentDuMenu Une référence à l'objet ContextMenuItem sélectionné.

Renvoie

Rien.

Description

Gestionnaire d'événement : invoqué lorsque l'élément du menu spécifié est sélectionné à partir du menu contextuel de Flash Player. Le gestionnaire de rappel spécifié reçoit deux paramètres : *obj*, une référence à l'objet situé sous le pointeur de la souris quand l'utilisateur a invoqué le menu contextuel Flash Player, et *élémentDuMenu*, une référence à l'objet ContextMenuItem représentant l'élément du menu sélectionné.

Exemple

L'exemple suivant affecte une fonction au gestionnaire `onSelect` pour un objet ContextMenuItem nommé `démarrer_cmi`. La fonction affiche la légende de l'élément du menu sélectionné.

```
démarrer_cmi.onSelect = function (obj, item) {
    trace("Vous choisissez : " + item.caption);
}
```

Consultez également

`ContextMenu.onSelect`

ContextMenuItem.separatorBefore

Disponibilité

Flash Player 7.

Usage

`élémentDuMenu_cmi``.separatorBefore`

Description

Propriété : une valeur booléenne qui indique si une barre de séparation doit apparaître au-dessus de l'élément du menu spécifié. Cette propriété est `false` par défaut.

Remarque : Une barre de séparation apparaît toujours entre les éléments du menu personnalisés et les éléments intégrés.

Exemple

Cet exemple crée trois éléments du menu étiquetés Ouvrir, Enregistrer et Imprimer. Une barre de séparation sépare les éléments Enregistrer et Imprimer. Les éléments du menu sont ensuite ajoutés au tableau `customItems` de l'objet ContextMenu. Enfin, le menu est associé au scénario principal du fichier SWF.

```
mon_cm = new ContextMenu();
ouvrir_cmi = new ContextMenuItem("Ouvrir", itemHandler);
enregistrer_cmi = new ContextMenuItem("Enregistrer", itemHandler);
imprimer_cmi = new ContextMenuItem("Imprimer", itemHandler);
imprimer_cmi.separatorBefore = true;
mon_cm.customItems.push(ouvrir_cmi, enregistrer_cmi, imprimer_cmi);
function itemHandler(obj, menuItem) {
    trace("Vous choisissez : " + menuItem.caption);
};
_root.menu = mon_cm;
```

Consultez également

`ContextMenu.onSelect`

ContextMenuItem.visible

Disponibilité

Flash Player 7.

Usage

élémentDuMenu_cmi.visible

Description

Propriété : une valeur booléenne qui indique si le menu spécifié est visible quand le menu contextuel Flash Player s'affiche. Par défaut, cette propriété est `true`.

continue

Disponibilité

Flash Player 4.

Usage

```
continue
```

Paramètres

Aucun.

Renvoie

Rien.

Description

Instruction : apparaît dans plusieurs types d'instructions de boucle et se comporte différemment dans chaque type de boucle.

Dans une boucle `while`, `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.

Dans une boucle `do while`, `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.

Dans une boucle `for`, `continue` oblige l'interprète de Flash à ignorer le reste du corps de la boucle et à atteindre l'évaluation de la post-expression `for` de la boucle.

Dans une boucle `for...in`, `continue` oblige l'interprète de Flash à ignorer le reste du corps de la boucle et à retourner au début de la boucle, où la valeur suivante dans l'énumération est traitée.

Consultez également

`do while`, `for`, `for..in`, `while`

Classe CustomActions

Disponibilité

Flash Player 6.

Description

Les méthodes de la classe CustomActions permettent à un fichier SWF lu dans l'outil de programmation Flash de gérer toutes les actions personnalisées qui sont enregistrées à l'aide de l'outil de programmation. Un fichier SWF peut installer et désinstaller des actions personnalisées, récupérer la définition XML d'une action personnalisée et récupérer la liste des actions personnalisées enregistrées.

Vous pouvez utiliser ces méthodes pour développer des fichiers SWF qui constituent des extensions de l'outil de programmation Flash. Une telle extension pourrait, par exemple, utiliser le protocole d'application Flash pour naviguer dans un référentiel UDDI et télécharger des services web dans la boîte à outils Actions.

Méthodes de la classe CustomActions

Méthode	Description
`CustomActions.get()`	Lit le contenu du fichier XML de définition d'actions personnalisées.
`CustomActions.install()`	Installe un nouveau fichier XML de définition d'actions personnalisées.
`CustomActions.list()`	Renvoie une liste de toutes les actions personnalisées enregistrées.
`CustomActions.uninstall`	Supprime un fichier XML de définition d'actions personnalisées.

CustomActions.get()

Disponibilité

Flash Player 6.

Usage

```
CustomActions.get(définitionDactionsPersonnalisées)
```

Paramètres

définitionDactionsPersonnalisées Le nom de la définition d'action personnalisée à récupérer.

Renvoie

Si la définition personnalisée de l'action XML est située, renvoie une chaîne ; dans le cas contraire, renvoie `undefined`.

Description

Méthode : lit le contenu du fichier XML de définition d'actions personnalisées nommé *définitionDactionsPersonnalisées*.

Le nom du fichier de définition doit être un nom de fichier simple, sans l'extension .xml et sans séparateurs de répertoires (« : :», « :/ :» ou « :\ :»).

Si le fichier de définition spécifié par *définitionDactionsPersonnalisées* est introuvable, une valeur de `undefined` est renvoyée. Si la définition XML des actions personnalisées spécifiée par le paramètre *définitionDactionsPersonnalisées* est trouvée, elle est lue en totalité et renvoyée sous forme de chaîne.

CustomActions.install()

Disponibilité

Flash Player 6.

Usage

```
CustomActions.install(définitionDactionsPersonnalisées,
  définitionXMLPersonnalisée)
```

Paramètres

définitionDactionsPersonnalisées Le nom de la définition d'action personnalisée à installer.

définitionXMLPersonnalisée Le texte de la définition XML à installer.

Renvoie

Une valeur booléenne `false` si une erreur se produit lors de l'installation : sinon, une valeur `true` est retournée pour indiquer que l'installation de l'action personnalisée est réussie.

Description

Méthode : installe un nouveau fichier XML de définition d'actions personnalisées indiqué par le paramètre *définitionDactionsPersonnalisées*. Le contenu du fichier est spécifié par la chaîne *définitionXMLPersonnalisée*.

Le nom du fichier de définition doit être un nom de fichier simple, sans l'extension .xml et sans séparateurs de répertoires (« : :», « :/ :» ou « :\ :»).

Si un fichier d'actions personnalisées existe déjà avec le nom *définitionDactionsPersonnalisées*, il est écrasé.

Si le répertoire Configuration/ActionsPanel/CustomActions n'existe pas lorsque cette méthode est invoquée, il est créé.

CustomActions.list()

Disponibilité

Flash Player 6.

Usage

```
CustomActions.list()
```

Paramètres

Aucun.

Renvoie

Un tableau.

Description

Méthode : renvoie un objet Array contenant les noms de toutes les actions personnalisées qui sont enregistrées avec l'outil de programmation Flash. Les éléments du tableau sont de simples noms, sans extension .xml et sans séparateurs de répertoires (par exemple, « : », « / » ou « \ »). Si aucune action personnalisée n'est enregistrée, la méthode `list()` renvoie un tableau de longueur zéro. Si une erreur se produit, la méthode `list()` renvoie la valeur `undefined`.

CustomActions.uninstall

Disponibilité

Flash Player 6.

Usage

```
CustomActions.uninstall(définitionDactionsPersonnalisées)
```

Paramètres

`définitionDactionsPersonnalisées` Le nom de la définition d'action personnalisée à désinstaller.

Renvoie

Une valeur booléenne `false` si aucune action personnalisée nommée *`définitionDactionsPersonnalisées`* n'est rencontrée. Si les actions personnalisées ont été correctement retirées, une valeur de `true` est renvoyée.

Description

Méthode : supprime le fichier XML de définition d'actions personnalisées nommé *`définitionDactionsPersonnalisées`*.

Le nom du fichier de définition doit être un nom de fichier simple, sans l'extension .xml et sans séparateurs de répertoires (« : :», « :/ :» ou « :\ :»).

Classe Date

Disponibilité

Flash Player 5.

Description

La classe Date permet de récupérer les valeurs de dates et d'heures en fonction du temps universel (Temps moyen de Greenwich, maintenant appelé Temps Universel Coordonné) ou en fonction du système d'exploitation sur lequel Flash Player est exécuté. Les méthodes de la classe Date ne sont pas statiques, mais ne s'appliquent qu'à l'objet Date individuel spécifié à l'appel de la méthode. La méthode `Date.UTC()`est une exception : c'est une méthode statique.

La classe Date gère l'heure d'été différemment suivant le système d'exploitation et la version de Flash Player. Flash Player 6 et les versions suivantes gèrent l'heure d'été sur les systèmes d'exploitation suivants comme suit :

- Windows – l'objet Date ajuste automatiquement son résultat pour refléter l'heure d'été. L'objet Date détecte si l'heure d'été est employée à l'endroit en question et, si c'est le cas, détecte les dates et heures de transition de l'heure d'hiver à l'heure d'été. Cependant, les dates de transition actuellement en vigueur sont appliquées aux dates passées et futures, et l'écart dû à l'heure d'été peut donc être calculé de façon incorrecte pour les dates passées lorsque le lieu en question avait des dates de transition différentes.
- Mac OS X – l'objet Date ajuste automatiquement son résultat pour refléter l'heure d'été. La base de données d'informations de fuseau horaire sous Mac OS X est utilisée pour déterminer si un écart dû à l'heure d'été devrait être appliqué à une date ou à une heure présente ou passée.

Flash Player 5 gère l'heure d'été sur les systèmes d'exploitation suivants comme suit :

- Windows : les réglementations américaines relatives à l'heure d'été sont toujours appliquées. Cela entraîne des transitions incorrectes en Europe et dans d'autres régions qui utilisent l'heure d'été mais dont les transitions diffèrent par rapport aux Etats-Unis. Flash détecte correctement si l'heure d'été est employée dans les paramètres régionaux en question.

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 même section.

Méthodes de la classe Date

Méthode	Description
`Date.getDate`	Renvoie le jour du mois, en fonction de l'heure locale.
`Date.getDay()`	Renvoie le jour de la semaine, en fonction de l'heure locale.
`Date.getFullYear`	Renvoie l'année (quatre chiffres), en fonction de l'heure locale.
`Date.getHours()`	Renvoie l'heure, en fonction de l'heure locale.
`Date.getMilliseconds()`	Renvoie les millisecondes, en fonction de l'heure locale.
`Date.getMinutes()`	Renvoie les minutes, en fonction de l'heure locale.
`Date.getMonth`	Renvoie le mois, en fonction de l'heure locale.
`Date.getSeconds`	Renvoie les secondes, en fonction de l'heure locale.

Méthode	Description
`Date.getTime()`	Renvoie le nombre de millisecondes écoulées depuis le premier janvier 1970 à minuit, temps universel.
`Date.getTimezoneOffset()`	Renvoie la différence, en minutes, entre l'heure locale de l'ordinateur et le temps universel.
`Date.getUTCDate()`	Renvoie le jour (date) du mois, en fonction du temps universel.
`Date.getUTCDay()`	Renvoie le jour de la semaine, en fonction du temps universel.
`Date.getUTCFullYear()`	Renvoie l'année (quatre chiffres), en fonction du temps universel.
`Date.getUTCHours()`	Renvoie l'heure, en fonction du temps universel.
`Date.getUTCMilliseconds()`	Renvoie les millisecondes, en fonction du temps universel.
`Date.getUTCMinutes()`	Renvoie les minutes, en fonction du temps universel.
`Date.getUTCMonth()`	Renvoie le mois, en fonction du temps universel.
`Date.getUTCSeconds()`	Renvoie les secondes, en fonction du temps universel.
`Date.getYear()`	Renvoie l'année, en fonction de l'heure locale.
`Date.setDate()`	Définit le jour du mois, en fonction de l'heure locale. Renvoie les nouvelles informations horaires, en millisecondes.
`Date.setFullYear()`	Définit l'année complète, en fonction de l'heure locale. Renvoie les nouvelles informations horaires, en millisecondes.
`Date.setHours()`	Définit l'heure, en fonction de l'heure locale. Renvoie les nouvelles informations horaires, en millisecondes.
`Date.setMilliseconds()`	Définit les millisecondes, en fonction de l'heure locale. Renvoie les nouvelles informations horaires, en millisecondes.
`Date.setMinutes()`	Définit les minutes, en fonction de l'heure locale. Renvoie les nouvelles informations horaires, en millisecondes.
`Date.setMonth()`	Définit le mois, en fonction de l'heure locale. Renvoie les nouvelles informations horaires, en millisecondes.
`Date.setSeconds()`	Définit les secondes, en fonction de l'heure locale. Renvoie les nouvelles informations horaires, en millisecondes.
`Date.setTime()`	Définit la date, en millisecondes. Renvoie les nouvelles informations horaires, en millisecondes.
`Date.setUTCDate()`	Définit la date, en fonction du temps universel. Renvoie les nouvelles informations horaires, en millisecondes.
`Date.setUTCFullYear()`	Définit l'année, en fonction du temps universel. Renvoie les nouvelles informations horaires, en millisecondes.
`Date.setUTCHours()`	Définit l'heure, en fonction du temps universel. Renvoie les nouvelles informations horaires, en millisecondes.
`Date.setUTCMilliseconds()`	Définit les millisecondes, en fonction du temps universel. Renvoie les nouvelles informations horaires, en millisecondes.

Méthode	Description
Date.setUTCMinutes()	Définit les minutes, en fonction du temps universel. Renvoie les nouvelles informations horaires, en millisecondes.
Date.setUTCMonth()	Définit le mois, en fonction du temps universel. Renvoie les nouvelles informations horaires, en millisecondes.
Date.setUTCSeconds()	Définit les secondes, en fonction du temps universel. Renvoie les nouvelles informations horaires, en millisecondes.
Date.setYear()	Définit l'année, en fonction de l'heure locale.
Date.toString()	Renvoie une valeur de chaîne représentant la date et l'heure stockées dans l'objet Date spécifié.
Date.UTC()	Renvoie le nombre de millisecondes écoulées entre le premier janvier 1970 à minuit, temps universel, et l'heure spécifiée.

Constructeur de la classe Date

Disponibilité

Flash Player 5.

Usage

```
new Date()
new Date(année, mois [, date [, heure [, minute [, seconde [, milliseconde
  ]]]]])
```

Paramètres

année Une valeur entre 0 et 99 indique une année entre 1900 et 1999, sinon, les quatre chiffres de l'année doivent être spécifiés.

mois Un entier compris entre 0 (janvier) et 11 (décembre).

date Un entier compris entre 1 et 31. Ce paramètre est facultatif.

heure Un entier compris entre 0 (minuit) et 23 (23h00).

minute Un entier compris entre 0 et 59. Ce paramètre est facultatif.

seconde Un entier compris entre 0 et 59. Ce paramètre est facultatif.

milliseconde Un entier compris entre 0 et 999. Ce paramètre est facultatif.

Renvoie

Rien.

Description

Objet : construit un nouvel objet Date contenant les informations horaires actuelles ou la date spécifiée.

Exemple

L'exemple suivant récupère l'heure et la date actuelles.

```
maintenant_date = new Date();
```

L'exemple suivant crée un nouvel objet Date pour l'anniversaire de Gary, le 12 août 1974. (Le paramètre du mois étant basé sur zéro, l'exemple utilise 7 pour le mois et non 8)

```
annivGary_date = new Date (74, 7, 12);
```

L'exemple suivant crée un nouvel objet Date, concatène la valeur renvoyée de `Date.getMonth`, `Date.getDate` et `Date.getFullYear` et les affiche dans le champ de texte spécifié par la variable `date_str`.

```
aujourdhui_date = new Date();
date_str = ((aujourdhui_date.getMonth() + 1) + "/" + aujourdhui_date.getDate()
  + "/" + aujourdhui_date.getFullYear());
```

Date.getDate

Disponibilité

Flash Player 5.

Usage

ma_date`.getDate()`

Paramètres

Aucun.

Renvoie

Un entier.

Description

Méthode : renvoie le jour du mois (un entier compris entre 1 et 31) de l'objet Date spécifié, en fonction de l'heure locale. L'heure locale est déterminée par le système d'exploitation sur lequel Flash Player est exécuté.

Date.getDay()

Disponibilité

Flash Player 5.

Usage

ma_date`.getDay()`

Paramètres

Aucun.

Renvoie

Un entier.

Description

Méthode : renvoie le jour de la semaine (0 pour dimanche, 1 pour lundi, et ainsi de suite) de l'objet Date spécifié, en fonction de l'heure locale. L'heure locale est déterminée par le système d'exploitation sur lequel Flash Player est exécuté.

Date.getFullYear

Disponibilité

Flash Player 5.

Usage

```
ma_date.getFullYear()
```

Paramètres

Aucun.

Renvoie

Un entier.

Description

Méthode : renvoie l'année complète (un nombre à quatre chiffres, tel que 2002) de l'objet Date spécifié, en fonction de l'heure locale. L'heure locale est déterminée par le système d'exploitation sur lequel Flash Player est exécuté.

Exemple

L'exemple suivant utilise le constructeur pour créer un nouvel objet Date et envoie la valeur renvoyée par la méthode `getFullYear` au panneau de sortie :

```
ma_date = new Date();
trace(ma_date.getFullYear());
```

Date.getHours()

Disponibilité

Flash Player 5.

Usage

```
ma_date.getHours()
```

Paramètres

Aucun.

Renvoie

Un entier.

Description

Méthode : renvoie l'heure (un entier compris entre 0 et 23) de l'objet Date spécifié, en fonction de l'heure locale. L'heure locale est déterminée par le système d'exploitation sur lequel Flash Player est exécuté.

Date.getMilliseconds()

Disponibilité

Flash Player 5.

Usage

ma_date.getMilliseconds()

Paramètres

Aucun.

Renvoie

Un entier.

Description

Méthode : renvoie les millisecondes (un entier compris entre 0 et 999) de l'objet Date spécifié, en fonction de l'heure locale. L'heure locale est déterminée par le système d'exploitation sur lequel Flash Player est exécuté.

Date.getMinutes()

Disponibilité

Flash Player 5.

Usage

ma_date.getMinutes()

Paramètres

Aucun.

Renvoie

Un entier.

Description

Méthode : renvoie les minutes (un entier compris entre 0 et 59) de l'objet Date spécifié, en fonction de l'heure locale. L'heure locale est déterminée par le système d'exploitation sur lequel Flash Player est exécuté.

Date.getMonth

Disponibilité

Flash Player 5.

Usage

```
ma_date.getMonth()
```

Paramètres

Aucun.

Renvoie

Un entier.

Description

Méthode : renvoie le mois (0 pour janvier, 1 pour février, et ainsi de suite) de l'objet Date spécifié, en fonction de l'heure locale. L'heure locale est déterminée par le système d'exploitation sur lequel Flash Player est exécuté.

Date.getSeconds

Disponibilité

Flash Player 5.

Usage

```
ma_date.getSeconds()
```

Paramètres

Aucun.

Renvoie

Un entier.

Description

Méthode : renvoie les secondes (un entier compris entre 0 et 59) de l'objet Date spécifié, en fonction de l'heure locale. L'heure locale est déterminée par le système d'exploitation sur lequel Flash Player est exécuté.

Date.getTime()

Disponibilité

Flash Player 5.

Usage

ma_date.getTime()

Paramètres

Aucun.

Renvoie

Un entier.

Description

Méthode : renvoie le nombre de millisecondes écoulées depuis le premier janvier 1970 à minuit, temps universel, pour l'objet Date spécifié. Utilisez cette méthode pour représenter un instant spécifique dans le temps lors d'une comparaison entre deux ou plusieurs objets Date.

Date.getTimezoneOffset()

Disponibilité

Flash Player 5.

Usage

ma_date.getTimezoneOffset()

Paramètres

Aucun.

Renvoie

Un entier.

Description

Méthode : renvoie la différence, en minutes, entre l'heure locale de l'ordinateur et le temps universel.

Exemple

L'exemple suivant renvoie la différence entre l'heure d'été locale de San Francisco et le temps universel. L'heure d'été est prise en compte dans le résultat renvoyé uniquement si la date définie dans l'objet Date se produit durant l'heure d'été.

```
trace(new Date().getTimezoneOffset());

// 420 est affiché dans le panneau de sortie
// (7 heures * 60 minutes/heure = 420 minutes)
// cet exemple est donné dans le fuseau horaire de la Californie (PDT, GMT-
  0700).
// le résultat peut varier en fonction du lieu et de la date.
```

Date.getUTCDate()

Disponibilité

Flash Player 5.

Usage

ma_date.getUTCDate()

Paramètres

Aucun.

Renvoie

Un entier.

Description

Méthode : renvoie le jour du mois (un entier compris entre 1 et 31) de l'objet Date spécifié, en fonction du temps universel.

Date.getUTCDay()

Disponibilité

Flash Player 5.

Usage

ma_date.getUTCDay()

Paramètres

Aucun.

Renvoie

Un entier.

Description

Méthode : renvoie le jour de la semaine (0 pour dimanche, 1 pour lundi, et ainsi de suite) de l'objet Date spécifié, en fonction du temps universel.

Date.getUTCFullYear()

Disponibilité

Flash Player 5.

Usage

```
ma_date.getUTCFullYear()
```

Paramètres

Aucun.

Renvoie

Un entier.

Description

Méthode : renvoie l'année complète (quatre chiffres) de l'objet Date spécifié, en fonction du temps universel.

Date.getUTCHours()

Disponibilité

Flash Player 5.

Usage

```
ma_date.getUTCHours()
```

Paramètres

Aucun.

Renvoie

Un entier.

Description

Méthode : renvoie les heures de l'objet Date spécifié, en fonction du temps universel.

Date.getUTCMilliseconds()

Disponibilité

Flash Player 5.

Usage

```
ma_date.getUTCMilliseconds()
```

Paramètres

Aucun.

Renvoie

Un entier.

Description

Méthode : renvoie les millisecondes de l'objet Date spécifié, en fonction du temps universel.

Date.getUTCMinutes()

Disponibilité

Flash Player 5.

Usage

```
ma_date.getUTCMinutes()
```

Paramètres

Aucun.

Renvoie

Un entier.

Description

Méthode : renvoie les minutes de l'objet Date spécifié, en fonction du temps universel.

Date.getUTCMonth()

Disponibilité

Flash Player 5.

Usage

ma_date.getUTCMonth()

Paramètres

Aucun.

Renvoie

Un entier.

Description

Méthode : renvoie le mois (0 pour janvier, 1 pour février, et ainsi de suite) de l'objet Date spécifié, en fonction du temps universel.

Date.getUTCSeconds()

Disponibilité

Flash Player 5.

Usage

ma_date.getUTCSeconds()

Paramètres

Aucun.

Renvoie

Un entier.

Description

Méthode : renvoie les secondes dans l'objet Date spécifié, en fonction du temps universel.

Date.getYear()

Disponibilité

Flash Player 5.

Usage

```
ma_date.getYear()
```

Paramètres

Aucun.

Renvoie

Un entier.

Description

Méthode : renvoie l'année de l'objet Date spécifié, en fonction de l'heure locale. L'heure locale est déterminée par le système d'exploitation sur lequel Flash Player est exécuté. L'année est l'année complète moins 1900. Par exemple, l'an 2000 est représenté par 100.

Consultez également

`Date.getFullYear`

Date.setDate()

Disponibilité

Flash Player 5.

Usage

```
ma_date.setDate(date)
```

Paramètres

date Un entier compris entre 1 et 31.

Renvoie

Un entier.

Description

Méthode : définit le jour du mois pour l'objet Date spécifié, en fonction de l'heure locale et renvoie les nouvelles informations, en millisecondes. L'heure locale est déterminée par le système d'exploitation sur lequel Flash Player est exécuté.

Date.setFullYear()

Disponibilité

Flash Player 5.

Usage

```
ma_date.setFullYear(année [, mois [, date]] )
```

Paramètres

année Un nombre à quatre chiffres spécifiant une année. Les nombres à deux chiffres ne représentent pas les années : par exemple, 99 n'est pas l'année 1999, mais l'année 99.

mois Un entier compris entre 0 (janvier) et 11 (décembre). Ce paramètre est facultatif.

date Un nombre entre 1 et 31. Ce paramètre est facultatif.

Renvoie

Un entier.

Description

Méthode : définit l'année pour l'objet Date spécifié, en fonction de l'heure locale et renvoie les nouvelles informations, en millisecondes. Si les paramètres *mois* et *date* sont spécifiés, ils sont également définis sur l'heure locale. L'heure locale est déterminée par le système d'exploitation sur lequel Flash Player est exécuté.

L'appel de cette méthode ne modifie pas les autres champs de l'objet Date spécifié, mais `Date.getUTCDay()` et `Date.getDay()` peuvent rapporter une nouvelle valeur si le jour de la semaine change suite à l'appel de cette méthode.

Date.setHours()

Disponibilité

Flash Player 5.

Usage

```
ma_date.setHours(heure)
```

Paramètres

heure Un entier compris entre 0 (minuit) et 23 (23h00).

Renvoie

Un entier.

Description

Méthode : définit les heures pour l'objet Date spécifié, en fonction de l'heure locale et renvoie les nouvelles informations, en millisecondes. L'heure locale est déterminée par le système d'exploitation sur lequel Flash Player est exécuté.

Date.setMilliseconds()

Disponibilité

Flash Player 5.

Usage

ma_date.setMilliseconds(*milliseconde*)

Paramètres

milliseconde Un entier compris entre 0 et 999.

Renvoie

Un entier.

Description

Méthode : définit les millisecondes pour l'objet Date spécifié, en fonction du temps universel et renvoie les nouvelles informations, en millisecondes. L'heure locale est déterminée par le système d'exploitation sur lequel Flash Player est exécuté.

Date.setMinutes()

Disponibilité

Flash Player 5.

Usage

ma_date.setMinutes(*minute*)

Paramètres

minute Un entier compris entre 0 et 59.

Renvoie

Un entier.

Description

Méthode : définit les minutes pour l'objet Date spécifié, en fonction du temps universel et renvoie les nouvelles informations, en millisecondes. L'heure locale est déterminée par le système d'exploitation sur lequel Flash Player est exécuté.

Date.setMonth()

Disponibilité

Flash Player 5.

Usage

ma_date.setMonth(*mois* [, *date*])

Paramètres

mois Un entier compris entre 0 (janvier) et 11 (décembre).

date Un entier compris entre 1 et 31. Ce paramètre est facultatif.

Renvoie

Un entier.

Description

Méthode : définit le mois pour l'objet Date spécifié, en fonction de l'heure locale et renvoie les nouvelles informations, en millisecondes. L'heure locale est déterminée par le système d'exploitation sur lequel Flash Player est exécuté.

Date.setSeconds()

Disponibilité

Flash Player 5.

Usage

ma_date.setSeconds(*seconde*)

Paramètres

seconde Un entier compris entre 0 et 59.

Renvoie

Un entier.

Description

Méthode : définit les secondes pour l'objet Date spécifié, en fonction de l'heure locale et renvoie les nouvelles informations, en millisecondes. L'heure locale est déterminée par le système d'exploitation sur lequel Flash Player est exécuté.

Date.setTime()

Disponibilité

Flash Player 5.

Usage

ma_date.setTime(*millisecondes*)

Paramètres

millisecondes Une valeur entière où 0 est le 1er janvier 1970 à 0h00 GMT.

Renvoie

Un entier.

Description

Méthode : définit la date pour l'objet Date spécifié en millisecondes depuis le 1er janvier 1970 à minuit et renvoie les nouvelles informations horaires, en millisecondes.

Date.setUTCDate()

Disponibilité

Flash Player 5.

Usage

ma_date.setUTCDate(*date*)

Paramètres

date Un entier compris entre 1 et 31.

Renvoie

Un entier.

Description

Méthode : définit la date pour l'objet Date spécifié, en fonction du temps universel et renvoie les nouvelles informations, 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 rapporter une nouvelle valeur si le jour de la semaine change suite à l'appel de cette méthode.

Date.setUTCFullYear()

Disponibilité

Flash Player 5.

Usage

```
ma_date.setUTCFullYear(année [, mois [, date]] )
```

Paramètres

année L'année spécifiée à l'aide d'un nombre à quatre chiffres, tel que 2000.

mois Un entier compris entre 0 (janvier) et 11 (décembre). Ce paramètre est facultatif.

date Un entier compris entre 1 et 31. Ce paramètre est facultatif.

Renvoie

Un entier.

Description

Méthode : définit l'année pour l'objet Date spécifié (*ma_date*), en fonction de l'heure locale et renvoie les nouvelles informations, en millisecondes.

En option, 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 rapporter une nouvelle valeur si le jour de la semaine change suite à l'appel de cette méthode.

Date.setUTCHours()

Disponibilité

Flash Player 5.

Usage

```
ma_date.setUTCHours(heure [, minute [, seconde [, milliseconde]]])
```

Paramètres

heure Un entier compris entre 0 (minuit) et 23 (23h00).

minute Un entier compris entre 0 et 59. Ce paramètre est facultatif.

seconde Un entier compris entre 0 et 59. Ce paramètre est facultatif.

milliseconde Un entier compris entre 0 et 999. Ce paramètre est facultatif.

Renvoie

Un entier.

Description

Méthode : définit l'heure pour l'objet Date spécifié, en fonction du temps universel et renvoie les nouvelles informations, en millisecondes.

Date.setUTCMilliseconds()

Disponibilité

Flash Player 5.

Usage

```
ma_date.setUTCMilliseconds(milliseconde)
```

Paramètres

milliseconde Un entier compris entre 0 et 999.

Renvoie

Un entier.

Description

Méthode : définit les millisecondes pour l'objet Date spécifié, en fonction du temps universel et renvoie les nouvelles informations, en millisecondes.

Date.setUTCMinutes()

Disponibilité

Flash Player 5.

Usage

```
ma_date.setUTCMinutes(minute [, seconde [, milliseconde]])
```

Paramètres

minute Un entier compris entre 0 et 59.

seconde Un entier compris entre 0 et 59. Ce paramètre est facultatif.

milliseconde Un entier compris entre 0 et 999. Ce paramètre est facultatif.

Renvoie

Un entier.

Description

Méthode : définit la minute pour l'objet Date spécifié, en fonction du temps universel et renvoie les nouvelles informations, en millisecondes.

Date.setUTCMonth()

Disponibilité

Flash Player 5.

Usage

```
ma_date.setUTCMonth(mois [, date ])
```

Paramètres

mois Un entier compris entre 0 (janvier) et 11 (décembre).

date Un entier compris entre 1 et 31. Ce paramètre est facultatif.

Renvoie

Un entier.

Description

Méthode : définit le mois et, éventuellement, le jour (*date*) pour l'objet Date spécifié, en fonction du temps universel et renvoie les nouvelles informations, 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 rapporter une nouvelle valeur si le jour de la semaine change suite à la spécification d'une valeur pour le paramètre *date*.

Date.setUTCSeconds()

Disponibilité

Flash Player 5.

Usage

```
ma_date.setUTCSeconds(seconde [, milliseconde]))
```

Paramètres

seconde Un entier compris entre 0 et 59.

milliseconde Un entier compris entre 0 et 999. Ce paramètre est facultatif.

Renvoie

Un entier.

Description

Méthode : définit les secondes pour l'objet Date spécifié, en fonction du temps universel et renvoie les nouvelles informations, en millisecondes.

Date.setYear()

Disponibilité

Flash Player 5.

Usage

ma_date.setYear(*année*)

Paramètres

année Si *année* est un entier compris entre 0 et 99, setYear définit l'année comme 1900 + *année*. Sinon, l'année est la valeur du paramètre année.

Renvoie

Un entier.

Description

Méthode : définit l'année pour l'objet Date spécifié, en fonction de l'heure locale et renvoie les nouvelles informations, en millisecondes. L'heure locale est déterminée par le système d'exploitation sur lequel Flash Player est exécuté.

Date.toString()

Disponibilité

Flash Player 5.

Usage

ma_date.toString()

Paramètres

Aucun.

Renvoie

Une chaîne.

Description

Méthode : renvoie une valeur chaîne pour l'objet Date spécifié dans un format lisible et renvoie les nouvelles informations, en millisecondes.

Exemple

L'exemple suivant renvoie les informations de l'objet Date annéeDeNaissance_date sous forme de chaîne.

```
var annéeDeNaissance_date = new Date(74, 7, 12, 18, 15);
trace (annéeDeNaissance_date.toString());
```

Résultat (pour l'heure standard du Pacifique) :

```
Mois août 12 18:15:00 GMT-0700 1974
```

Date.UTC()

Disponibilité

Flash Player 5.

Usage

```
Date.UTC(année, mois [, date [, heure [, minute [, seconde [, milliseconde
  ]]]]])
```

Paramètres

année Un nombre à quatre chiffres, tel que 2000.

mois Un entier compris entre 0 (janvier) et 11 (décembre).

date Un entier compris entre 1 et 31. Ce paramètre est facultatif.

heure Un entier compris entre 0 (minuit) et 23 (23h00).

minute Un entier compris entre 0 et 59. Ce paramètre est facultatif.

seconde Un entier compris entre 0 et 59. Ce paramètre est facultatif.

milliseconde Un entier compris entre 0 et 999. Ce paramètre est facultatif.

Renvoie

Un entier.

Description

Méthode : renvoie le nombre de millisecondes écoulées entre le premier janvier 1970 à minuit, temps universel, et la date spécifiée dans les paramètres. Ceci est une méthode statique invoquée avec le constructeur d'objet Date et non avec un objet Date spécifique. Cette méthode vous permet de créer un objet Date qui adopte le temps universel, alors que le constructeur Date adopte l'heure locale.

Exemple

L'exemple suivant crée un nouvel objet Date `annivGary_date`, défini en fonction du temps universel. Il s'agit de la variante en temps universel de l'exemple utilisé pour la méthode constructeur `new Date`.

```
annivGary_date = new Date(Date.UTC(1974, 7, 12));
```

default

Disponibilité

Flash Player 6.

Usage

```
default: instructions
```

Paramètres

instructions Toute instruction.

Renvoie

Rien.

Description

Instruction : définit la casse par défaut pour une action `switch`. Les instructions sont exécutées si le paramètre *Expression* de l'action `switch` n'est pas égal (avec égalité stricte) à l'un des paramètres *Expression* qui suit les mots-clés `case` pour une action `switch` donnée.

Une action `switch` n'est pas nécessaire pour avoir une hauteur de casse par défaut (`default`). Une hauteur de casse par défaut (`default`) n'a pas besoin d'être la dernière dans la liste. L'utilisation d'une action `default` en dehors d'une action `switch` est une erreur, et la compilation du script échoue.

Exemple

Dans l'exemple suivant, l'expression A n'est pas égale aux expressions B ou D. L'instruction qui suit le mot-clé default est donc exécutée et l'action `trace()` est envoyée au panneau de sortie.

```
switch ( A ) {
  case B:
    C;
    break
  case D:
    E;
    break
  default
    trace ("aucune casse spécifique");
}
```

Consultez également

`switch`, `case`, `break`

delete

Disponibilité

Flash Player 5.

Usage

```
delete référence
```

Paramètres

référence Le nom de la variable ou de l'objet à éliminer.

Renvoie

Une valeur booléenne.

Description

Opérateur : détruit l'objet ou la variable spécifié par le paramètre *référence* et renvoie `true` si l'objet a été supprimé avec succès : sinon, renvoie `false`. Cet opérateur est utile pour libérer de la mémoire utilisée par des scripts. Quoique `delete` soit un opérateur, il est généralement utilisé comme instruction, comme dans l'exemple suivant :

```
delete x;
```

L'opérateur `delete` peut échouer et renvoyer `false` si le paramètre *référence* n'existe pas ou ne peut pas être effacé. Les objets et propriétés prédéfinis, et les variables déclarées avec `var`, ne peuvent pas être supprimés. Vous ne pouvez pas utiliser l'opérateur `delete` pour retirer des clips.

Exemple

Usage 1 : l'exemple suivant crée un objet, l'utilise et le supprime une fois qu'il n'est plus nécessaire.

```
compte = new Object();
compte.nom = 'Jon';
compte.solde = 10000;

delete compte;
```

Usage 2 : l'exemple suivant supprime une propriété d'un objet.

```
// créer l'objet "compte"
compte = new Object();
// affecter le nom de propriété au compte
compte.nom = 'Jon';
// supprimer la propriété
delete compte.nom;
```

Usage 3 : l'exemple suivant illustre une autre façon de supprimer une propriété d'objet.

```
// créer un objet Array de longueur 0
mon_array = new Array();
// ajouter un élément au tableau. // Array.length est maintenant 1
mon_array[0] = "abc";
// ajouter un autre élément au tableau. // Array.length est maintenant 2
mon_array[1] = "def";
// ajouter un autre élément au tableau. // Tableau.length est maintenant 3
mon_array[1] = "ghi";
// mon_array[2] est supprimé, mais Array.length n'est pas changé
delete monTableau[2];
trace(mon_array.length);
```

Usage 4 : l'exemple suivant illustre le comportement de `delete` sur les références d'objets.

```
// créer un objet et affecter la variable ref1
// pour faire référence à l'objet
ref1 = new Object();
ref1.nom = "Jody";
// copier la variable de référence dans une nouvelle variable
// et supprimer ref1
ref2 = ref1;
delete ref1;
```

Si `ref1` n'avait pas été copié dans `ref2`, l'objet aurait été supprimé à la suppression de `ref1`, car il n'y aurait plus eu de référence à l'objet. Si vous supprimiez `ref2`, il n'y aurait plus de référence à l'objet, qui serait détruit et la mémoire qu'il utilisait serait à nouveau disponible.

Consultez également

`var`

do while

Disponibilité

Flash Player 4.

Usage

```
do {
   instruction(s)
} while (condition)
```

Paramètres

`condition` La condition à évaluer.

`instruction(s)` La ou les instructions à exécuter tant que le paramètre *`condition`* est évalué comme `true`.

Renvoie

Rien.

Description

Instruction : exécute les instructions, puis évalue la condition dans une boucle, tant que la condition est `true`.

Consultez également

`break`, `continue`

duplicateMovieClip()

Disponibilité

Flash Player 4.

Usage

```
duplicateMovieClip(cible, nouveauNom, profondeur)
```

Paramètres

`cible` Le chemin cible de l'animation à dupliquer.

`nouveauNom` Un identifiant unique pour le clip dupliqué.

`profondeur` Un niveau de profondeur unique pour le clip dupliqué. Le niveau de profondeur est un ordre d'empilement des clips dupliqués. Cet ordre d'empilement est similaire à l'ordre d'empilement des calques dans le scénario, les clips de niveau de profondeur plus faible étant masqués sous des clips d'ordre d'empilement plus élevé. Vous devez affecter à chaque clip dupliqué un niveau de profondeur unique pour l'empêcher de remplacer des fichiers SWF sur des profondeurs occupées.

Renvoie

Une référence au clip dupliqué.

Description

Fonction : crée une occurrence de clip pendant la lecture du fichier SWF. Dans les clips dupliqués, la tête de lecture commence toujours à l'image 1, quel que soit l'endroit où se trouve la tête de lecture dans le clip original (ou « : parent : »). Les variables du clip parent ne sont pas copiées dans le clip dupliqué. Si le clip parent est effacé, le clip dupliqué l'est également. Utilisez l'action ou la méthode `removeMovieClip()` pour supprimer une occurrence de clip créée avec `duplicateMovieClip()`.

Consultez également

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

dynamic

Disponibilité

Flash Player 6.

Usage

```
dynamic class nomDeLaClasse  [ extends superClasse ]
                [ implements nomDinterface [, nomDinterface... ] ]
{
  // définition de la classe ici
}
```

Remarque : Pour utiliser ce mot-clé, vous devez définir ActionScript 2.0 et Flash Player 6 ou version ultérieure dans l'onglet Flash de la boîte de dialogue Paramètres de publication de votre fichier FLA. Ce mot-clé n'est supporté que lorsqu'il est utilisé dans des fichiers de scripts externes et non dans des scripts écrits dans le panneau Actions.

Description

Mot-clé : définit que les objets basés sur la classe spécifiée peuvent ajouter des propriétés dynamiques et y accéder au moment de l'exécution.

La vérification du type sur les classes dynamiques est moins stricte que sur les classes non dynamiques, car les membres auxquels vous accédez dans la définition et les occurrences de classe ne sont pas comparés à ceux définis dans le domaine des classes. Il est toutefois toujours possible de vérifier le type de retour et les types de paramètres de ces fonctions de membre de classe. Ce comportement s'avère particulièrement utile pour la manipulation des objets MovieClip, car il existe plusieurs façons d'ajouter de manière dynamique des propriétés et des objets à un clip, par exemple `MovieClip.createEmptyMovieClip()` et `MovieClip.createTextField()`.

Les sous-classes de classes dynamiques sont également dynamiques.

Pour plus d'informations, consultez *Création de classes dynamiques*, page 182.

Exemple

Dans l'exemple suivant, la classe B a été marquée comme dynamique ; ainsi, l'appel d'une fonction non déclarée ne génèrera pas d'erreur au moment de la compilation.

```
// dans B.as
dynamic class B extends class_A {
  function B() {
    /*il s'agit du constructeur*/
  }
  function m():Number {return 25;}
  function o(s:String):Void {trace(s);}
}
// dans C.as
class C extends class_A {
  function C() {
    /*il s'agit du constructeur*/
  }
  function m():Number {return 25;}
  function o(s:String):Void {trace(s);}
}
// dans un autre script
var var1 = B.n();   // pas d'erreur
var var2 = C.n()    // erreur puisqu'il n'existe pas de fonction n dans C.as
```

Consultez également

`class`, `extends`

else

Disponibilité

Flash Player 4.

Usage

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

Paramètres

`condition` Une expression évaluée en tant que `true` ou `false`.

`instruction(s)` Une série alternative d'instructions à exécuter si la condition spécifiée dans l'instruction `if` est `false`.

Renvoie

Rien.

Description

Instruction : spécifie les instructions à exécuter si la condition de l'instruction `if` renvoie `false`.

Consultez également

`if`

else if

Disponibilité

Flash Player 4.

Usage

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

Paramètres

condition Une expression évaluée en tant que `true` ou `false`.

instruction(s) Une série alternative d'instructions à exécuter si la condition spécifiée dans l'instruction `if` est `false`.

Renvoie

Rien.

Description

Instruction : évalue une condition et spécifie les instructions à exécuter si la condition de l'instruction `if` initiale renvoie `false`. Si la condition `else if` renvoie `true`, l'interprète de Flash exécute les instructions suivant la condition entre accolades ({}). Si la condition `else if` est `false`, Flash ignore les instructions entre accolades et exécute les instructions suivant les accolades. Utilisez l'action `else if` pour créer une logique de branchement dans vos scripts.

Exemple

L'exemple suivant utilise des actions `else if` pour vérifier si chaque côté d'un objet se trouve à l'intérieur d'une limite spécifique :

```
// si l'objet sort des limites,
// le renvoyer et inverser sa vitesse de déplacement
     if (this._x>limiteDroite) {
       this._x = limiteDroite;
       xInc = -xInc;
     } else if (this._x<limiteGauche) {
       this._x = limiteGauche;
       xInc = -xInc;
     } else if (this._y>limiteInf) {
       this._y = limiteInf;
       yInc = -yInc;
     } else if (this._y<limiteSup) {
       this._y = limiteSup;
       yInc = -yInc;
     }
```

Consultez également

`if`

#endinitclip

Disponibilité

Flash Player 6.

Usage

```
#endinitclip
```

Paramètres

Aucun.

Renvoie

Rien.

Description

Directive du compilateur : indique la fin d'un bloc d'actions d'initialisation de composant.

Exemple

```
#initclip
...les actions d'initialisation de composant sont placées ici...
#endinitclip
```

Consultez également

`#initclip`

eq (égal à – spécifique à la chaîne)

Disponibilité

Flash Player 4. Cet opérateur est déconseillé dans Flash 5 et remplacé par l'opérateur `==` `(égalité)`.

Usage

`expression1` `eq` *`expression2`*

Paramètres

`expression1, expression2` Nombres, chaînes ou variables.

Renvoie

Rien.

Description

Opérateur de comparaison : compare l'égalité de deux expressions et renvoie `true` si la représentation chaîne de *`expression1`* est égale à la représentation chaîne de *`expression2`* : sinon, l'opération renvoie `false`.

Consultez également

`== (égalité)`

Classe Error

Disponibilité

Flash Player 7.

Description

Contient les informations concernant une erreur qui s'est produite dans un script. Vous créez une objet Error à l'aide de la fonction constructeur `Error`. Généralement, vous « émettez » un nouvel objet Error à partir d'un bloc de code `try` qui est ensuite « saisi » par un bloc de code `catch` ou `finally`.

Vous pouvez également créer une sous-classe de la classe Error et émettre des occurrences de cette sous-classe.

Méthode de la classe Error

Méthode	Description
`Error.toString()`	Renvoie la représentation chaîne d'un objet Error.

Propriétés de la classe Error

Propriété	Description
`Error.message`	Une chaîne qui contient un message d'erreur associé à une erreur.
`Error.name`	Une chaîne qui contient le nom de l'objet Error.

Constructeur de la classe Error

Disponibilité

Flash Player 7.

Usage

```
new Error([message])
```

Paramètres

`message` Une chaîne associée à l'objet Error ; ce paramètre est facultatif.

Renvoie

Rien.

Description

Constructeur : crée un nouvel objet Error. Si le *`message`* est spécifié, sa valeur est affectée à la propriété `Error.message` de l'objet.

Exemple

Dans l'exemple suivant, une fonction émet une erreur (avec un message spécifié) si les deux chaînes auxquelles elle est transmise ne sont pas identiques.

```
function compareStrings(string_1, string_2) {
  if(string_1 != string_2) {
    throw new Error("Les chaînes ne correspondent pas.");
  }
}
try
  compareStrings("Chien","chien");
} catch (e) {
  trace(e.toString());
}
```

Consultez également

throw, try..catch..finally

Error.message

Disponibilité

Flash Player 7.

Usage

monErreur.message

Description

Propriété : contient le message associé à l'objet Error. Par défaut, la valeur de la propriété est "Error". Vous pouvez spécifier une propriété message quand vous créez un nouvel objet Error en transmettant la chaîne d'erreur à la fonction du constructeur Error.

Consultez également

throw, try..catch..finally

Error.name

Disponibilité

Flash Player 7.

Usage

monErreur.name

Description

Propriété : contient le nom de l'objet Error. Par défaut, la valeur de la propriété est "Error".

Consultez également

throw, try..catch..finally

Error.toString()

Disponibilité

Flash Player 7.

Usage

mon_err.toString()

Renvoie

Une chaîne.

Description

Méthode : renvoie la chaîne "Error" par défaut ou la valeur contenue dans Error.message, si défini.

Consultez également

Error.message, throw, try..catch..finally

escape

Disponibilité

Flash Player 5.

Usage

escape(*expression*)

Paramètres

expression L'expression à convertir en chaîne et à coder dans un format de code URL.

Renvoie

Rien.

Description

Fonction : convertit le paramètre en une chaîne et l'encode dans un format d'URL encodée où tous les caractères non-alphanumériques sont échappés avec des séquences % hexadécimales.

Exemple

L'exécution du code suivant donne le résultat Bonjour%7B%5BMonde%5D%7D.

```
escape("Bonjour{[Monde]}");
```

Consultez également

unescape

eval()

Disponibilité

Flash Player 5 et versions suivantes pour une pleine utilisation de la fonctionnalité. Vous pouvez utiliser la fonction `eval()` lors de l'exportation vers Flash Player 4, mais devez utiliser une notation à barre oblique et ne pouvez accéder qu'aux variables, pas aux propriétés ni aux objets.

Usage

```
eval(expression);
```

Paramètres

`expression` Une chaîne contenant le nom d'une variable, d'une propriété, d'un objet ou d'un clip à récupérer.

Renvoie

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

Description

Fonction : accède aux variables, propriétés, objets ou clips par nom. Si *`expression`* est une variable ou une propriété, la valeur de la variable ou de la propriété est renvoyée. Si *`expression`* est un objet ou un clip, une référence à l'objet ou au clip est renvoyée. Si l'élément nommé dans *`expression`* est introuvable, `undefined` est renvoyé.

Dans Flash 4, la fonction `eval()` était utilisée pour simuler des tableaux ; dans Flash 5 et les versions suivantes, il est recommandé d'utiliser la classe pour simuler des tableaux.

Dans Flash 4, vous pouvez également utiliser la fonction `eval()` pour définir et récupérer dynamiquement la valeur d'une variable ou d'un nom d'occurrence. Cependant, vous pouvez également obtenir le même résultat avec l'opérateur d'accès tableau (`[]`).

Dans Flash 5 et les versions suivantes, vous ne pouvez pas utiliser la fonction `eval()` pour définir et récupérer dynamiquement la valeur d'une variable ou d'un nom d'occurrence car vous ne pouvez pas utiliser la fonction `eval()` dans la partie gauche d'une équation. Par exemple, remplacez le code suivant :

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

par celui-ci :

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

ou :

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

Exemple

L'exemple suivant utilise la fonction `eval()` pour déterminer la valeur de l'expression `"morceau" + x`. Le résultat étant un nom de variable, `morceau3`, `eval()` renvoie la valeur de la variable et l'affecte à `y` :

```
morceau3 = "dangereux";
x = 3;
y = eval("morceau" + x);
trace(y);
// Résultat : dangereux
```

Consultez également

Classe Array

extends

Disponibilité

Flash Player 6.

Usage

```
class nomDeLaClasse extends autreNomDeClasse {}
interface nomDinterface extends autreNomDinterface {}
```

Remarque : Pour utiliser ce mot-clé, vous devez définir ActionScript 2.0 et Flash Player 6 ou version ultérieure dans l'onglet Flash de la boîte de dialogue Paramètres de publication de votre fichier FLA. Ce mot-clé n'est supporté que lorsqu'il est utilisé dans des fichiers de scripts externes et non dans des scripts écrits dans le panneau Actions.

Paramètres

`nomDeLaClasse` Le nom de la classe que vous définissez.

`autreNomDeClasse` Le nom de la classe sur laquelle *`nomDeLaClasse`* est basé.

`nomDinterface` Le nom de l'interface que vous définissez.

`autreNomDinterface` Le nom de l'interface sur laquelle *`nomDinterface`* est basé.

Description

Mot-clé : définit une classe ou une interface qui est une sous-classe d'une autre classe ou interface ; la dernière est une superclasse. La sous-classe hérite de toutes les méthodes, propriétés, fonctions, etc., qui sont définies dans la super-classe.

Pour plus d'informations, consultez *Création de sous-classes*, page 171.

Exemple

Dans la classe B, telle que définie ci-dessous, un appel au constructeur de la classe A sera automatiquement inséré comme première instruction de la fonction du constructeur de B, puisque aucun appel n'existe à cet endroit. (C'est-à-dire qu'il est commenté dans l'exemple.)

```
class B extends class A
{
  function B() { // il s'agit du constructeur
//   super(); // facultatif; inséré durant la compilation si omis
   }
  function m():Number {return 25;}
  function o(s:String):Void {trace(s);}
}
```

Consultez également

`class`, `implements`, `interface`

false

Disponibilité

Flash Player 5.

Usage

```
false
```

Description

Constante : Une valeur booléenne unique représentant l'opposé de `true`.

Consultez également

`true`

_focusrect

Disponibilité

Flash Player 4.

Usage

```
_focusrect = booléen;
```

Description

Propriété (globale) : spécifie si un rectangle jaune apparaît autour du bouton ou du clip avec focus clavier. La valeur par défaut, `true`, affiche un rectangle jaune autour du bouton ou du clip avec focus, à mesure que l'utilisateur appuie sur la touche de tabulation pour parcourir les objets du fichier SWF. Spécifiez `false` si vous ne voulez pas afficher le rectangle jaune. Il s'agit d'une propriété globale qui peut être annulée dans des cas spécifiques.

Consultez également

`Button._focusrect`, `MovieClip._focusrect`

for

Disponibilité

Flash Player 5.

Usage

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

Paramètres

init Une expression à évaluer avant de commencer la séquence de boucle, généralement une expression d'affectation. Une instruction `var` est également autorisée pour ce paramètre.

condition Une expression évaluée en tant que `true` ou `false`. La condition est évaluée avant chaque itération de boucle : la boucle sort lorsque la condition est évaluée comme `false`.

next Une expression à évaluer après chaque itération de boucle ; généralement une expression d'affectation utilisant les opérateurs ++ (incrémentation) ou `--` (décrémentation).

instruction(s) Une instruction ou des instructions à exécuter à l'intérieur du corps de la boucle.

Description

Instruction : une construction de boucle qui évalue l'expression `init` (initialiser) une fois, puis commence une séquence de boucle par laquelle, tant que *condition* est évaluée comme `true`, *instruction* est exécutée et l'expression suivante évaluée.

Certaines propriétés ne peuvent pas être énumérées par les actions `for` ou `for..in`. Par exemple, les méthodes intégrées de la classe Array (comme `Array.sort` et `Array.reverse()`) ne sont pas comprises dans l'énumération d'un objet Array, et les propriétés du clip, comme _x et _y, ne sont pas énumérées. Dans des fichiers de classe externes, les membres d'occurrence ne peuvent pas être énumérés : seuls les membres dynamiques et statiques peuvent être énumérés.

Exemple

L'exemple suivant utilise `for` pour additionner les éléments d'un tableau :

```
mon_array=new Array();
for(i=0; i<10; i++) {
  mon_array [i] = (i + 5)*10;
  trace(mon_array[i]);
}
```

Les résultats suivants sont affichés dans le panneau de sortie :

```
50
60
70
80
90
100
110
120
130
140
```

L'exemple suivant utilise `for` pour effectuer la même action de façon répétée. Dans le code ci-dessous, la boucle `for` additionne les nombres de 1 à 100 :

```
var somme = 0;
  for (var i=1; i<=100; i++) {
    somme = somme + i;
  }
```

Consultez également

`++ (incrémentation)`, `-- (décrémentation)`, `for..in`, `var`

for..in

Disponibilité

Flash Player 5.

Usage

```
for(itérantDeVariable in objet){
  instruction(s);
}
```

Paramètres

`itérantDeVariable` Le nom d'une variable qui joue le rôle d'itérant, référençant chaque propriété d'un objet ou chaque élément d'un tableau.

`objet` Le nom d'un objet à répéter.

`instruction(s)` Une instruction à exécuter pour chaque itération.

Renvoie

Rien.

Description

Instruction : effectue une boucle sur les propriétés d'un objet ou sur les éléments d'un tableau et exécute l'*`instruction`* pour chaque propriété d'un objet.

Certaines propriétés ne peuvent pas être énumérées par les actions `for` ou `for..in`. Par exemple, les méthodes intégrées de la classe Array (comme `Array.sort` et `Array.reverse()`) ne sont pas comprises dans l'énumération d'un objet Array, et les propriétés du clip, comme `_x` et `_y`, ne sont pas énumérées. Dans des fichiers de classe externes, les membres d'occurrence ne peuvent pas être énumérés : seuls les membres dynamiques et statiques peuvent être énumérés.

La construction `for..in` itère sur les propriétés des objets dans la chaîne prototype de l'objet itéré. Si le prototype de l'enfant est `parent`, l'itération sur les propriétés de l'enfant avec `for..in` itérera également sur les propriétés de `parent`.

L'action `for..in` énumère tous les objets de la chaîne prototype d'un objet. Les propriétés de l'objet sont énumérées en premier, puis les propriétés de son prototype immédiat, puis les propriétés du prototype du prototype, et ainsi de suite. L'action `for..in` n'énumère pas deux fois le même nom de propriété. Si l'objet `child` a le prototype `parent` et que tous deux contiennent la propriété `prop`, l'action `for..in` appelée pour `child` énumère `prop` de `child` mais ignore celle de `parent`.

Exemple

L'exemple suivant utilise `for..in` pour itérer sur les propriétés d'un objet :

```
monObjet = { nom:'Tara', age:27, ville:'San Francisco' };
for (nom in monObjet) {
  trace ("monObjet." + nom + " = " + monObjet[nom]);
}
```

Le résultat de cet exemple est le suivant :

```
monObjet.nom = Tara
monObjet.age = 27
monObjet.ville = San Francisco
```

L'exemple suivant utilise l'opérateur `typeof` avec `for..in` pour itérer sur un enfant particulier :

```
for (nom in mon_mc) {
  if (typeof (mon_mc[nom]) = "clip") {
    trace ("J'ai un clip appelé " + nom);
  }
}
```

L'exemple suivant énumère les enfants d'un clip et envoie chacun dans l'image 2 de leurs scénarios respectifs. Le clip `GroupeDeBoutonsRadio` est un parent de plusieurs enfants, `_BoutonRadioRouge_`, `_BoutonRadioVert_` et `_BoutonRadioBleu`.

```
for (nom de variable in GroupeDeBoutonsRadio) {
  GroupeDeBoutonsRadio[nom].gotoAndStop(2);
}
```

fscommand()

Disponibilité

Flash Player 3.

Usage

```
fscommand("commande", "paramètres")
```

Paramètres

`commande` Une chaîne transmise à l'application hôte pour toute utilisation ou une commande transmise à Flash Player.

`paramètres` Une chaîne transmise à l'application hôte pour toute utilisation ou une valeur transmise à Flash Player.

Renvoie

Rien.

Description

Fonction : permet au fichier SWF de communiquer avec Flash Player ou avec le programme qui héberge Flash Player, tel qu'un navigateur web. Vous pouvez également utiliser l'action `fscommand` pour transmettre des messages à Macromedia Director, ou à Visual Basic, Visual C++ et tout autre programme capable d'héberger les contrôles ActiveX.

Usage 1 : pour envoyer un message à Flash Player, vous devez utiliser des commandes et paramètres prédéfinis. Le tableau suivant indique les valeurs que vous pouvez spécifier pour les paramètres *commande* et *paramètres* de l'action `fscommand` pour contrôler un fichier SWF lu dans Flash Player autonome (y compris les projections) :

Commande	Paramètres	Objectif
`quit`	`Aucun`	Ferme la projection.
`fullscreen`	`true` ou `false`	La spécification de `true` définit Flash Player en mode plein écran. La spécification de `false` renvoie le lecteur en affichage normal du menu.
`allowscale`	`true` ou `false`	La spécification de `false` définit le lecteur de sorte que le fichier SWF soit toujours affiché dans sa taille originale et que son échelle ne soit jamais modifiée. La spécification de `true` oblige le fichier SWF à adopter l'échelle 100 % du lecteur.
`showmenu`	`true` ou `false`	La spécification de `true` active le jeu complet des éléments de menu contextuel. La spécification de `false` masque tous les éléments de menu contextuel, à l'exception de A propos de Flash Player.
`exec`	`Chemin de l'application`	Exécute une application depuis la projection.
`trapallkeys`	`true` ou `false`	La spécification de `true` envoie tous les éléments de touche, y compris les raccourcis, au gestionnaire `onClipEvent(keyDown/keyUp)` de Flash Player.

La commande `exec` ne peut contenir que les caractères A–Z, a–z, 0–9, point (.) et trait de soulignement (_). La commande `exec` n'est exécutée que dans le sous-répertoire fscommand. En d'autres termes, si vous utilisez la commande `fscommand exec` pour appeler une application, cette dernière doit résider dans un sous-dosser nommé fscommand.

Usage 2 : pour utiliser l'action `fscommand` pour envoyer un message à un langage de programmation tel que JavaScript dans un navigateur web, vous pouvez transmettre deux paramètres quelconques dans les paramètres *commande* et *paramètres*. Ces paramètres peuvent être des chaînes ou des expressions et sont utilisés dans une fonction JavaScript qui traite l'action `fscommand`.

Dans un navigateur web, l'action `fscommand` appelle la fonction JavaScript `nomanimation_DoFScommand` dans la page HTML contenant le fichier SWF. `moviename` est le nom de Flash Player, tel qu'affecté par l'attribut `NAME` de la balise `EMBED` ou la propriété ID de la balise `OBJECT`. Si vous affectez le nom `monDocument` à Flash Player, la fonction JavaScript appelée est `monDocument_DoFScommand`.

Usage 3 : l'action `fscommand` peut envoyer des messages à Macromedia Director qui sont interprétés par Lingo comme des chaînes, des événements ou un code exécutable Lingo. Si le message est une chaîne ou un événement, vous devez rédiger le code Lingo pour le recevoir depuis l'action `fscommand` et entraîner une action dans Director. Pour plus d'informations, consultez le centre de support de Director à l'adresse www.macromedia.com/support/director.

Usage 4 : en Visual Basic, Visual C++, et dans d'autres programmes pouvant héberger les contrôles ActiveX, fscommand envoie un événement VB avec deux chaînes qui peut être traité dans l'environnement du langage de programmation. Pour plus d'informations, utilisez les mots-clés *méthode Flash* pour effectuer une recherche sur le centre de support de Flash à l'adresse www.macromedia.com/go/flash_support_fr.

Exemple

Usage 1 : dans l'exemple suivant, l'action `fscommand` définit Flash Player de sorte que l'animation soit affichée en mode plein écran lorsque le bouton est relâché.

```
on(release) {
  fscommand("fullscreen", true);
}
```

Usage 2 : l'exemple suivant utilise l'action `fscommand` appliquée à un bouton dans Flash pour ouvrir une boîte de message JavaScript dans une page HTML. Le message même est envoyé à JavaScript en tant que paramètre de `fscommand`.

Vous devez ajouter une fonction à la page HTML contenant le fichier SWF. Cette fonction, *`monDocument`*`_DoFSCommand`, réside dans la page HTML et attend une action `fscommand` dans Flash. Lorsqu'une action `fscommand` est déclenchée dans Flash (par exemple, lorsqu'un utilisateur clique sur le bouton), les chaînes `commande` et `paramètres` sont transmises à la fonction *`monDocument`*`_DoFSCommand`. Vous pouvez utiliser les chaînes transmises dans votre code JavaScript ou VBScript comme vous l'entendez. Dans cet exemple, la fonction contient une instruction conditionnelle `if` qui vérifie si la chaîne de commande est `"fenêtreMessage"`. Le cas échéant, une fenêtre d'alerte JavaScript s'ouvre et affiche le contenu de la chaîne `paramètres`.

```
function monDocument_DoFSCommand(commande, args) {
  if (commande == "fenêtreMessage") {
    alert(args);
  }
}
```

Dans le document Flash, ajoutez l'action `fscommand` à un bouton :

```
fscommand("fenêtreMessage", "Ceci est une fenêtre de message appelée depuis
  Flash.")
```

Vous pouvez également utiliser des expressions pour l'action et les paramètres `fscommand`, comme dans l'exemple suivant :

```
fscommand("fenêtreMessage", "Bonjour, " + nom + ", bienvenue sur notre site
  web!")
```

Pour tester l'animation, choisissez Fichier >Aperçu avant publication > HTML.

Remarque : Si vous publiez votre fichier SWF avec le modèle Flash et FSCommand dans les paramètres de publication HTML, la fonction `monDocument_DoFSCommand` est automatiquement insérée. Les attributs `NAME` et `ID` du fichier SWF seront le nom du fichier. Par exemple, pour le fichier `monDocument.fla`, les attributs seront définis sur monDocument.

function

Disponibilité

Flash Player 5.

Usage

```
function nomDeFonction ([paramètre0, paramètre1,...paramètreN]){
  instruction(s)
}
function ([paramètre0, paramètre1,...paramètreN]){
  instruction(s)
}
```

Paramètres

nomDeFonction Le nom de la nouvelle fonction.

paramètre Un identifiant représentant un paramètre à transmettre à la fonction. Ces paramètres sont facultatifs.

instruction(s) Toute instruction ActionScript que vous avez définie pour le corps de la `fonction`.

Renvoie

Rien.

Description

Instruction : un jeu d'instructions que vous définissez pour effectuer une certaine tâche. Vous pouvez *déclarer*, ou définir, une fonction à un emplacement et l'appeler, ou l'invoquer, depuis différents scripts dans un fichier SWF. Lorsque vous définissez une fonction, vous pouvez également spécifier ses paramètres. Les paramètres sont des supports pour les valeurs sur lesquelles la fonction opère. Vous pouvez transmettre différents paramètres à une fonction à chaque fois que vous l'appelez. Ceci vous permet de réutiliser une fonction dans de nombreuses situations différentes.

Utilisez l'action `return` dans les *instruction(s)* d'une fonction pour obliger une fonction à renvoyer, ou générer, une valeur.

Usage 1 : déclare une `function` avec les *nomDeFonction*, *paramètres* et *instruction(s)* spécifiés. Lorsqu'une fonction est appelée, la déclaration de la fonction est invoquée. La référence en aval est autorisée : au sein d'une même liste d'actions, une fonction peut être déclarée après avoir été appelée. Une déclaration de fonction remplace toute déclaration précédente de cette même fonction. Vous pouvez utiliser cette syntaxe partout où une instruction est autorisée.

Usage 2 : crée une fonction anonyme et la renvoie. Cette syntaxe est utilisée dans les expressions et est particulièrement utile pour installer des méthodes dans des objets.

Exemple

Usage 1 : l'exemple suivant définit la fonction `sqr`, qui accepte un argument et renvoie la valeur `square(x*x)` du paramètre. Si la fonction est déclarée et utilisée dans le même script, la déclaration de la fonction doit apparaître après l'utilisation de la fonction.

```
y=sqr(3);

function sqr(x) {
  return x*x;
}
```

Usage 2 : la fonction suivante définit un objet Cercle :

```
function Cercle(rayon) {
  this.rayon = rayon;
}
```

L'instruction suivante définit une fonction anonyme qui calcule l'aire d'un cercle et l'associe à l'objet `Cercle` comme méthode :

```
Cercle.prototype.area = function () {return Math.PI * this.radius *
  this.radius}
```

Classe Function

Disponibilité

Flash Player 6.

Méthodes de la classe Function

Méthode	Description
`Function.apply()`	Active l'appel d'une fonction par le code ActionScript.
`Function.call`	Invoque la fonction représentée par un objet Function.

Propriétés de la classe Function

Propriété	Description
`Function.prototype`	Fait référence à un objet qui est le prototype d'une classe.

Function.apply()

Disponibilité

Flash Player 6.

Usage

maFonction.apply(*cetObjet*, *objetArguments*)

Paramètres

cetObjet L'objet auquel est appliqué *maFonction*.

objetArguments Un tableau dont les éléments sont transmis à *maFonction* en tant que paramètres.

Renvoie

Toute valeur spécifiée par la fonction appelée.

Description

Méthode : spécifie la valeur de `this` comme devant être utilisée avec toute fonction appelée par ActionScript. Cette méthode spécifie également les paramètres à transmettre à toute fonction appelée. La méthode `apply()` étant une méthode de la classe Function, elle est également une méthode de chaque objet de fonction dans ActionScript.

Les paramètres sont spécifiés sous forme d'objet Array. Ceci est souvent utile lorsque le nombre de paramètres à transmettre n'est pas connu avant l'exécution du script.

Exemple

Les invocations de fonction suivantes sont équivalentes :

```
Math.atan2(1, 0)
Math.atan2.apply(null, [1, 0])
```

Vous pouvez construire un fichier SWF contenant des champs de saisie qui permettent à l'utilisateur d'entrer le nom d'une fonction à invoquer, ainsi que des paramètres zéro ou plus à transmettre à la fonction. Une pression sur un bouton d'appel utiliserait alors la méthode `apply` pour appeler la fonction spécifiant les paramètres.

Dans cet exemple, l'utilisateur spécifie le nom d'une fonction dans un champ de saisie de texte appelé `nomDeFonction`. Le nombre de paramètres est spécifié dans un champ de saisie de texte appelé `nombreParamètres`. Jusqu'à 10 paramètres sont spécifiés dans des champs de texte appelés `paramètre1`, `paramètre2`, jusqu'à `paramètre10`.

```
on (release) {
  callTheFunction();
}
...
function callTheFunction()
{
   var laFonction = eval(nomDeFonction.text);
   var n = Number(nombreParamètres);
   var paramètres = [];
   for (var i = 0; i <n; i++) {
      paramètres.push(eval("paramètre" + i));
   }
   laFonction.apply(null, paramètres);
}
```

Function.call

Disponibilité

Flash Player 6.

Usage

maFonction.call(*cetObjet*, *paramètre1*, ..., *paramètreN*)

Paramètres

cetObjet Spécifie la valeur de `this` dans le corps de la fonction.

paramètre1 Un paramètre à transmettre à *maFonction*. Vous pouvez spécifier zéro ou plusieurs paramètres.

paramètreN

Renvoie

Rien.

Description

Méthode : invoque la fonction représentée par un objet Function. Chaque fonction dans ActionScript est représentée par un objet Function, toutes les fonctions supportent donc cette méthode.

Dans presque tous les cas, l'opérateur d'appel de la fonction (`()`) peut être utilisé au lieu de cette méthode. L'opérateur d'appel de la fonction permet de rédiger le code de façon concise et lisible. Cette méthode est surtout utile lorsque le paramètre `this` de l'invocation de fonction doit être explicitement contrôlé. Normalement, si une fonction est invoquée en tant que méthode d'un objet, dans le corps de la fonction, `this` est défini sur `monObjet`, comme dans l'exemple suivant :

```
monObjet.maMéthode(1, 2, 3);
```

Dans certaines situations, vous pourrez vouloir que `this` désigne autre chose : par exemple, si une fonction doit être invoquée en tant que méthode d'un objet, mais n'est pas effectivement stockée comme méthode de cet objet.

```
monObjet.maMéthode.call(monAutreObjet, 1, 2, 3);
```

Vous pouvez transmettre la valeur `null` pour le paramètre *`cetObjet`* pour invoquer une fonction en tant que fonction ordinaire et pas 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)
```

Exemple

Cet exemple utilise la méthode `Function.call()` pour imposer à une fonction un comportement identique à celui d'une méthode d'un autre objet, sans stocker la fonction dans l'objet.

```
function MonObjet() {
}
function MaMéthode(obj) {
  trace("this == obj? " + (this == obj));
}
var obj = new MonObjet();
MaMéthode.call(obj, obj);
```

L'action `trace()` envoie le code suivant vers le panneau de sortie :

```
this == obj? true
```

Function.prototype

Disponibilité

Flash Player 5. Si vous utilisez ActionScript 2.0, vous ne devez pas utiliser cette propriété : elle représente l'implémentation de l'héritage de ActionScript 1.

Usage

`maFonction``.prototype`

Description

Propriété : dans une fonction constructeur ActionScript 1, la propriété `prototype` fait référence à un objet qui est le prototype de la classe construite. Chaque occurrence de la classe, créée par la fonction constructeur, hérite de toutes les propriétés et méthodes de l'objet prototype.

ge (supérieur ou égal à - spécifique aux chaînes)

Disponibilité

Flash Player 4. Cet opérateur est déconseillé dans Flash 5 et remplacé par l'opérateur >= (supérieur ou égal à).

Usage

`expression1` `ge` *`expression2`*

Paramètres

`expression1`, *`expression2`* Nombres, chaînes ou variables.

Renvoie

Rien.

Description

Opérateur (comparaison) : compare la représentation chaîne de *expression1* avec la représentation chaîne de *expression2* et renvoie `true` si *expression1* est supérieure ou égale à *expression2* : sinon, renvoie `false`.

Consultez également

`>= (supérieur ou égal à)`

get

Disponibilité

Flash Player 6.

Usage

```
function get property() {
  // vos instructions
}
```

Remarque : Pour utiliser ce mot-clé, vous devez définir ActionScript 2.0 et Flash Player 6 ou version ultérieure dans l'onglet Flash de la boîte de dialogue Paramètres de publication de votre fichier FLA. Ce mot-clé n'est supporté que lorsqu'il est utilisé dans des fichiers de scripts externes et non dans des scripts écrits dans le panneau Actions.

Paramètres

propriété Le mot que vous souhaitez utiliser pour faire référence à la propriété à laquelle `get` accède ; cette valeur doit être identique à celle utilisée dans la commande `set` correspondante.

Renvoie

La valeur de la propriété spécifiée par *nomDeProp*.

Description

Mot-clé : permet une « obtention » implicite des propriétés associées aux objets basés sur des classes définies dans des fichiers de classe externes. L'utilisation de méthodes d'obtention implicites vous permet d'accéder aux propriétés d'objets sans accéder directement à ces objets. Les méthodes get/set implicites sont des abréviations syntaxiques de la méthode `Object.addProperty()` dans ActionScript 1.

Pour plus d'informations, consultez *Méthodes get/set implicites*, page 181.

Consultez également

`Object.addProperty()`, `set`

getProperty

Disponibilité

Flash Player 4.

Usage

```
getProperty(mon_mc, propriété)
```

Paramètres

mon_mc Le nom d'occurrence d'un clip dont la propriété est récupérée.

propriété La propriété d'un clip.

Renvoie

La valeur de la propriété spécifiée.

Description

Fonction : renvoie la valeur de la propriété spécifiée pour le clip *mon_mc*.

Exemple

L'exemple suivant récupère la coordonnée de l'axe horizontal (_x) pour le clip mon_mc et l'affecte à la variable mon_mc_x :

```
mon_mc_x = getProperty(_root.mon_mc, _x);
```

getTimer

Disponibilité

Flash Player 4.

Usage

```
getTimer()
```

Paramètres

Aucun.

Renvoie

Le nombre de millisecondes écoulées depuis le démarrage de la lecture du fichier SWF.

Description

Fonction : renvoie le nombre de millisecondes écoulées depuis le démarrage de la lecture du fichier SWF.

getURL()

Disponibilité

Flash 2. Les options `GET` et `POST` ne sont disponibles que dans Flash Player 4 et les versions ultérieures du lecteur.

Usage

```
getURL(url [, fenêtre [, "variables"]])
```

Paramètres

`url` L'URL où se trouve le document à obtenir.

`fenêtre` Un paramètre facultatif spécifiant la fenêtre ou le cadre HTML dans laquelle ou lequel le document doit être chargé. Vous pouvez entrer le nom d'une fenêtre spécifique ou choisir parmi les noms cibles réservés suivants :

- `_self` spécifie l'image courante dans la fenêtre courante.
- `_blank` spécifie une nouvelle fenêtre.
- `_parent` spécifie le parent de l'image courante.
- `_top` spécifie l'image de premier niveau dans la fenêtre courante.

`variables` Une méthode `GET` ou `POST` pour envoyer des variables. Omettez ce paramètre s'il n'y a aucune variable. La méthode `GET` ajoute les variables à la fin de l'URL et est utilisée pour des petits nombres de variables. La méthode `POST` envoie les variables dans un en-tête HTTP distinct et est utilisée pour les longues chaînes de variables.

Renvoie

Rien.

Description

Fonction : charge un document depuis une URL spécifique dans une fenêtre ou transmet les variables à une autre application, à une URL définie. Pour tester cette action, assurez-vous que le fichier à charger se trouve à l'emplacement spécifié. Pour utiliser une URL absolue (par exemple, *`http://www.monServeur.fr`*), il vous faut une connexion de réseau.

Exemple

Cet exemple charge une nouvelle URL dans une fenêtre de navigateur vide. L'action `getURL()` cible la variable `incomingAd` comme paramètre *`url`* de sorte que vous pouvez changer l'URL chargée sans avoir à modifier le fichier SWF. La valeur de la variable `incomingAd` est transmise à Flash plus tôt dans le fichier SWF avec une action `loadVariables()`.

```
on(release) {
  getURL(incomingAd, "_blank");
}
```

Consultez également

`loadVariables()`, `XML.send`, `XML.sendAndLoad`, `XMLSocket.send`

getVersion

Disponibilité

Flash Player 5.

Usage

```
getVersion()
```

Paramètres

Aucun.

Renvoie

Une chaîne contenant la version de Flash Player et les informations de plate-forme.

Description

Fonction : renvoie une chaîne contenant la version de Flash Player et les informations de plate-forme.

La fonction `getVersion` renvoie uniquement les informations pour Flash Player 5 ou les versions ultérieures.

Exemple

L'exemple suivant illustre une chaîne renvoyée par la fonction `getVersion`.

```
WIN 5,0,17,0
```

Ceci indique que la plate-forme est Microsoft Windows et que la version de Flash Player est la version principale 5, avec une version mineure de 17 (5.0r17).

Consultez également

`System.capabilities.os`, `System.capabilities.version`

_global object

Disponibilité

Flash Player 6.

Usage

```
_global.identifiant
```

Paramètres

Aucun.

Renvoie

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

Description

Identifiant : crée des variables, des objets ou des classes globaux. Par exemple, vous pourriez créer une bibliothèque exposée comme objet ActionScript global, comme l'objet Math ou Date. A la différence des variables et fonctions déclarées dans un scénario ou déclarées localement, les variables et fonctions globales sont visibles dans chaque scénario et étendue du fichier SWF, à condition qu'elles ne soient pas masquées par des identifiants portant le même nom dans des étendues internes.

Exemple

L'exemple suivant crée une fonction de haut niveau, `factorial()`, qui est disponible à chaque scénario et étendue d'un fichier SWF :

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

Consultez également

`var`, `Variable set`

gotoAndPlay

Disponibilité

Flash 2.

Usage

```
gotoAndPlay([séquence], image)
```

Paramètres

séquence Une chaîne facultative spécifiant le nom de la séquence vers laquelle la tête de lecture est envoyée.

image Un nombre représentant le numéro de l'image ou une chaîne représentant l'étiquette de l'image vers laquelle la tête de lecture est envoyée.

Renvoie

Rien.

Description

Fonction : envoie la tête de lecture vers l'image spécifiée d'une séquence et lit à partir de cette image. Si aucune séquence n'est spécifiée, la tête de lecture passe à l'image spécifiée de la séquence courante.

Exemple

Lorsque l'utilisateur clique sur un bouton auquel l'action `gotoAndPlay()` est affectée, la tête de lecture est envoyée vers l'image 16 et commence la lecture.

```
on(release) {
  gotoAndPlay(16);
}
```

Consultez également

`MovieClip.gotoAndPlay()`

gotoAndStop()

Disponibilité

Flash 2.

Usage

```
gotoAndStop([séquence], image)
```

Paramètres

séquence Une chaîne facultative spécifiant le nom de la séquence vers laquelle la tête de lecture est envoyée.

image Un nombre représentant le numéro de l'image ou une chaîne représentant l'étiquette de l'image vers laquelle la tête de lecture est envoyée.

Renvoie

Rien.

Description

Fonction : envoie la tête de lecture vers l'image spécifiée d'une séquence et l'arrête. Si aucune séquence n'est spécifiée, la tête de lecture passe à l'image spécifiée de la séquence courante.

Exemple

Lorsque l'utilisateur clique sur un bouton auquel l'action `gotoAndStop()` est affectée, la tête de lecture est envoyée vers l'image 5 dans la séquence actuelle et le fichier SWF arrête la lecture.

```
on(release) {
  gotoAndStop(5);
}
```

Consultez également

`stop()`

gt (supérieur à – spécifique aux chaînes)

Disponibilité

Flash Player 4. Cet opérateur est déconseillé dans Flash 5 et remplacé par le nouvel opérateur > (supérieur à).

Usage

```
expression1 gt expression2
```

Paramètres

expression1, expression2 Nombres, chaînes ou variables.

Description

Opérateur (comparaison) : compare la représentation chaîne de *expression1* avec la représentation chaîne de *expression2* et renvoie `true` si *expression1* est supérieure à *expression2* : sinon, renvoie `false`.

Consultez également

`> (supérieur à)`

_highquality

Disponibilité

Flash Player 4 : déconseillé et remplacé par `_quality`.

Usage

```
_highquality
```

Description

Propriété déconseillée (globale) : spécifie le niveau d'anti-aliasing appliqué au fichier SWF en cours. Spécifiez 2 (qualité maximum) pour appliquer une qualité élevée avec le lissage bitmap toujours actif. Spécifiez 1 (qualité élevée) pour appliquer l'anti-aliasing ; cela permettra de lisser les bitmaps si le fichier SWF ne contient pas d'animation. Spécifiez 0 (faible qualité) pour empêcher l'anti-aliasing.

Exemple

```
_highquality = 1;
```

Consultez également

`_quality`, `toggleHighQuality()`

if

Disponibilité

Flash Player 4.

Usage

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

Paramètres

condition Une expression évaluée en tant que `true` ou `false`.

instruction(s) Les instructions à exécuter si ou lorsque la condition est évaluée comme `true`.

Renvoie

Rien.

Description

Fonction : évalue une condition pour déterminer la prochaine action dans un fichier SWF. Si la 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 suivant les accolades. Utilisez l'action `if` pour créer une logique de branchement dans vos scripts.

Exemple

Dans l'exemple suivant, la condition entre parenthèses évalue la variable `nom` pour déterminer si elle a la valeur littérale `"Erica"`. Le cas échéant, l'action `play()` entre accolades est exécutée.

```
if(nom == "Erica"){
  play();
}
```

L'exemple suivant utilise une action `if` pour évaluer le moment auquel un objet déplaçable du fichier SWF est relâché par l'utilisateur. Si l'objet est relâché moins de 300 millisecondes après avoir été déplacé, la condition est évaluée comme `true` et les instructions entre accolades sont exécutées. Ces instructions définissent des variables pour stocker le nouvel emplacement de l'objet, la force avec laquelle il a été lancé et la vitesse à laquelle il a été lancé. La variable `tempsEnfoncé` est également réinitialisée. Si l'objet est relâché plus de 300 millisecondes après avoir été déplacé, la condition est évaluée comme `false` et aucune des instructions n'est exécutée.

```
if (getTimer()<tempsEnfoncé+300) {
     // si la condition est true,
     // l'objet a été lancé.
     // quelle est la nouvelle position de l'objet ?
     xNouvEmpl = this._x;
     yNouvEmpl = this._y;
     // avec quelle force a-t-il été lancé ?
     xDéplac = xNouvEmpl-xEmpl;
     yDéplac = yNouvEmpl-yEmpl;
     // la définition de la vitesse de l'objet dépend de
     // la distance de son déplacement
     xInc = xDéplac/2;
     yInc = yDéplac/2;
     tempsEnfoncé = 0;
   }
```

Consultez également

`else`

ifFrameLoaded

Disponibilité

Flash Player 3. L'action `ifFrameLoaded` est déconseillée dans Flash 5 ; Macromedia recommande l'utilisation de la propriété `MovieClip._framesloaded`.

Usage

```
ifFrameLoaded([séquence], image) {
  instruction(s);
}
```

Paramètres

`séquence` Une chaîne facultative spécifiant le nom de la séquence devant être chargée.

`image` Le numéro ou l'étiquette de l'image à charger avant l'exécution de la prochaine instruction.

`instruction(s)` Les instructions à exécuter si la séquence (ou la séquence et l'image) spécifiée(s) sont chargées.

Renvoie

Rien.

Description

Action à éviter : vérifie si le contenu d'une image est disponible localement. Utilisez `ifFrameLoaded` pour commencer la lecture d'une animation simple pendant que le reste du fichier SWF est téléchargé sur l'ordinateur local. La différence entre l'utilisation de `_framesloaded` et de `ifFrameLoaded` est que `_framesloaded` vous permet d'ajouter vos propres instructions `if` ou `else`.

Consultez également

`MovieClip._framesloaded`

implements

Disponibilité

Flash Player 6.

Usage

```
maClasse implements interface01 [, interface02, ...]
```

Remarque : Pour utiliser ce mot-clé, vous devez définir ActionScript 2.0 et Flash Player 6 ou version ultérieure dans l'onglet Flash de la boîte de dialogue Paramètres de publication de votre fichier FLA. Ce mot-clé n'est supporté que lorsqu'il est utilisé dans des fichiers de scripts externes et non dans des scripts écrits dans le panneau Actions.

Description

Mot-clé : définit une classe qui doit fournir des implémentations pour toutes les méthodes définies dans l'interface (ou les interfaces) implémentée(s). Pour plus d'informations, consultez *Interfaces comme types de données*, page 177.

Exemple

Pour plus d'informations, consultez `interface`.

Consultez également

`class`, `extends`, `interface`

import

Disponibilité

Flash Player 6.

Usage

```
import nomDeLaClasse
import nomPaquet.*
```

Remarque : Pour utiliser ce mot-clé, vous devez définir ActionScript 2.0 et Flash Player 6 ou version ultérieure 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.

Paramètres

nomDeLaClasse Le nom pleinement qualifié d'une classe définie dans un fichier de classe externe.

nomPaquet Un répertoire dans lequel vous avez stocké des fichiers de classe liés.

Description

Mot-clé : vous permet d'accéder à des classes sans définir leurs noms pleinement qualifiés. Par exemple, si vous souhaitez utiliser la classe macr.util.users.UserClass.as dans un script, vous devez y faire référence à l'aide de son nom pleinement qualifié ou en l'important : si vous l'importez, vous pouvez y faire référence à l'aide du nom de classe simple :

```
// avant importation
var monUtilisateur:UserClass = new macr.util.users.UserClass();
// après importation
import macr.util.users.ClasseUtilisateur;
var monUtilisateur:ClasseUtilisateur = new ClasseUtilisateur();
```

S'il existe plusieurs fichiers de classe dans le répertoire auquel vous souhaitez accéder, vous pouvez les importer via une seule instruction :

```
import macr.util.users.*;
```

Vous devez émettre l'instruction `import` avant de tenter d'accéder à la classe importée dans définir pleinement son nom.

Si vous importez une classe mais que vous ne l'utilisez pas ultérieurement dans votre script, la classe n'est pas exportée comme partie du fichier SWF. Cela signifie que vous importez des paquets volumineux sans vous soucier de la taille de votre fichier SWF : le code d'octet associé à une classe n'est inséré dans un fichier SWF que si cette classe est réellement utilisée.

L'instruction `import` s'applique uniquement au script courant (image ou objet) dans lequel elle est appelée. Supposons par exemple que vous importiez toutes les classes du paquet macr.util sur l'image 1 d'un document Flash. Sur cette image, vous pouvez référencer les classes par leur simple nom dans ce paquet.

```
// sur l'image 1 d'un fichier FLA :
import macr.util.*;

var myFoo:foo = new foo();
```

Sur un autre script d'image, vous devrez cependant référencer des classes dans ce paquet par leurs noms pleinement qualifiés (`var myFoo:foo = new macr.util.foo();`) ou ajouter une instruction `import` à l'autre image qui importe également les classes dans ce paquet.

Pour plus d'informations sur l'importation, consultez *Importation de classes*, page 180 et *Utilisation de paquets*, page 179.

#include

Disponibilité

Flash Player 4.

Usage

```
#include "[chemin] nomDeFichier.as"
```

Remarque : Ne placez pas de point virgule (;) à la fin de la ligne contenant l'instruction `#include`.

Paramètres

`[chemin] nomDeFichier.as` Le nom de fichier et le chemin facultatif pour le script à ajouter au panneau Actions ; *.as* est l'extension de fichier recommandée.

Renvoie

Rien.

Description

Directive du compilateur : inclut le contenu du fichier spécifié, comme si les commandes du fichier faisait partie du script d'appel lui-même. La directive `#include` est invoquée lors de la compilation. De plus, si vous modifiez un fichier externe, vous devez l'enregistrer 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 du fichier inclus est également vérifiée.

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

Vous pouvez ne spécifier aucun chemin, indiquer un chemin relatif ou un chemin absolu pour le fichier à inclure.

- Si vous ne spécifiez aucun chemin, le fichier AS doit se situer dans le même répertoire que le fichier FLA ou le script contenant l'instruction `#include`.
- Pour définir un chemin pour le fichier AS par rapport au fichier FLA ou au script, utilisez un point (.) pour indiquer le répertoire en cours, deux points (. .) pour indiquer un répertoire parent et des barres obliques (/). Consultez les exemples suivants.
- Pour spécifier un chemin absolu pour le fichier AS, utilisez le format supporté par votre plate-forme (Macintosh ou Windows). Consultez les exemples suivants. Cependant, cet utilisation n'est pas recommandée, car elle nécessite que la structure du répertoire soit la même sur toutes les machines utilisées pour compiler le script.

Exemple

Les exemples suivants montrent diverses façons de spécifier un chemin pour un fichier à inclure dans votre script.

```
// Notez que les instructions #include ne se terminent pas par un point virgule
  (;)
// le fichier AS est dans le même répertoire que le fichier ou le script FLA
#include "init_script.as"

// le fichier AS se trouve dans un sous-répertoire du répertoire
//  contenant le fichier ou le script FLA
// Le sous-répertoire est nommé "FLA_includes"
#include "FLA_includes/init_script.as"

// Le fichier AS se trouve dans un répertoire au même niveau que le fichier ou
  le script FLA
// Le répertoire est nommé "ALL_includes"
#include "../ALL_includes/init_script.as"

// Le fichier AS est spécifié par un chemin absolu dans Windows
// Notez l'utilisation de barres obliques et non de barres obliques inverses
#include "C:/Flash_scripts/init_script.as"

// Le fichier AS est spécifié par un chemin absolu dans Macintosh
#include "Mac HD:Flash_scripts:init_script.as"
```

Consultez également

`import`

Infinity

Disponibilité

Flash Player 5.

Usage

```
Infinity
```

Description

Constante : spécifie la valeur IEEE-754 représentant l'infini positif. La valeur de cette constante est la même que `Number.POSITIVE_INFINITY`.

-Infinity

Disponibilité

Flash Player 5.

Usage

```
-Infinity
```

Description

Constante : spécifie la valeur IEEE-754 représentant l'infini négatif. La valeur de cette constante est la même que `Number.NEGATIVE_INFINITY`.

#initclip

Disponibilité

Flash Player 6.

Usage

```
#initclip ordre
```

Paramètres

ordre Un entier qui spécifie l'ordre d'exécution des blocs de code `#initclip`. Ce paramètre est facultatif.

Description

Directive du compilateur : indique le début d'un bloc d'actions d'initialisation. Lorsque plusieurs clips sont initialisés en même temps, vous pouvez utiliser le paramètre *order* pour spécifier l'initialisation ayant lieu en premier. Les actions d'initialisation sont exécutées lors de la définition d'un symbole de clip. Si le clip est un symbole exporté, les actions d'initialisation sont exécutées avant les actions de l'image 1 du fichier SWF. Sinon, elles sont exécutées immédiatement avant les actions s'appliquant à l'image qui contient la première occurrence du symbole de clip associé.

Les actions d'initialisation sont exécutées une seule fois lors de la lecture d'un fichier SWF ; utilisez-les en tant qu'initialisations ponctuelles, par exemple, pour la définition et l'enregistrement de classes.

Consultez également

`#endinitclip`

instanceof

Disponibilité

Flash Player 6.

Usage

objet instanceof *classe*

Paramètres

objet Un objet ActionScript.

classe Une référence à une fonction du constructeur ActionScript, telle que String ou Date.

Renvoie

Si *objet* est une occurrence de *classe*, instanceof renvoie true : sinon, instanceof renvoie false. De même, _global instanceof Object renvoie false.

Description

Opérateur : détermine si un objet appartient à une classe spécifiée. Teste si *objet* est une occurrence de *classe*.

L'opérateur instanceof ne convertit pas les types primitifs en objets enveloppes. Par exemple, le code suivant renvoie true

```
new String("Bonjour") instanceof String;
```

Le code suivant renvoie false

```
"Bonjour" instanceof String;
```

Consultez également

typeof

int

Disponibilité

Flash Player 4. Cette fonction est déconseillée dans Flash 5 ; utilisez plutôt `Math.round()`.

Usage

```
int(valeur)
```

Paramètres

valeur Un nombre devant être arrondi à un entier.

Renvoie

Rien.

Description

Fonction : convertit un nombre décimal à la valeur de l'entier le plus proche.

Consultez également

`Math.floor`

interface

Disponibilité

Flash Player 6.

Usage

```
interface nomDinterface {}
interface nomDinterface [extends nomDinterface [, nomDinterface ...] {}
```

Remarque : Pour utiliser ce mot-clé, vous devez définir ActionScript 2.0 et Flash Player 6 ou version ultérieure dans l'onglet Flash de la boîte de dialogue Paramètres de publication de votre fichier FLA. Ce mot-clé n'est supporté que lorsqu'il est utilisé dans des fichiers de scripts externes et non dans des scripts écrits dans le panneau Actions.

Description

Mot-clé : définit une interface. Une interface est similaire à une classe, mais présente les différences notables ci-dessous :

- Les interfaces contiennent seulement des déclarations de méthodes et non leurs implémentations. Autrement dit, chaque classe qui implémente une interface doit fournir une implémentation pour chaque méthode déclarée dans l'interface.
- Seuls les membres publics sont autorisés dans une définition d'interface. De plus, les membres d'occurrence et de classe ne sont pas autorisés.
- Les instructions `get` et `set` ne sont pas autorisées dans les définitions d'interface.

Pour plus d'informations, consultez *Création et utilisation d'interfaces*, page 175.

Exemple

L'exemple suivant illustre plusieurs méthodes de définition et d'implémentation d'interfaces.

```
(dans les fichiers .as du paquet de premier niveau Ia, B, C, Ib, D, Ic, E)

// nom de fichier Ia.as
interface Ia
{
  function k():Nombre;         // déclaration de méthode seulement
  function n(x:Nombre):Nombre; // sans implémentation
}
// nomDeFichier B.as
class B implements Ia
{
  function k():Nombre {return 25;}
  function n(x:Nombre):Nombre {return x+5;}
}
// script externe ou panneau Actions
var mvar:B = new B();
trace(mvar.k());   // 25
trace(mvar.n(7)); // 12

// nomDeFichier c.as
class C implements Ia
{
  function k():Nombre {return 25;}
} // erreur: la classe doit implémenter toutes les méthodes d'interface

// nomDeFichier Ib.as
```

```
interface Ib
{
  function o():Void;
}
class D implements Ia, Ib
{
  function k():Nombre {return 15;}
  function n(x:Nombre):Nombre {return x*x;}
  function o():Void {trace("o");}
}

// script externe ou panneau Actions
mvar = new D();
trace(D.k()); // 15
trace(D.n(7));  // 49
trace(D.o()); // "o"

interface Ic extends Ia
{
  function p():Void;
}
class E implements Ib, Ic
{
  function k():Nombre {return 25;}
  function n(x:Nombre):Nombre {return x+5;}
  function o():Void {trace("o");}
  function p():Void {trace("p");}
}
```

Consultez également

class, extends, implements

isFinite

Disponibilité

Flash Player 5.

Usage

```
isFinite(expression)
```

Paramètres

expression Une valeur booléenne, une variable ou une autre expression à évaluer.

Renvoie

Une valeur booléenne.

Description

Fonction : évalue *expression* et renvoie `true` s'il s'agit d'un nombre fini et `false` s'il s'agit d'infini ou d'infini négatif. La présence d'infini, ou d'infini négatif, indique une condition d'erreur mathématique (une division par 0, par exemple).

Exemple

Les exemples suivants sont des exemples de valeurs renvoyées pour `isFinite` :

```
isFinite(56)
// renvoie true

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

isNaN()

Disponibilité

Flash Player 5.

Usage

```
isNaN(expression)
```

Paramètres

expression Une valeur booléenne, une variable ou une autre expression à évaluer.

Renvoie

Une valeur booléenne.

Description

Fonction : évalue le paramètre et renvoie `true` si la valeur n'est pas un nombre (`NaN`), indiquant la présence d'erreurs mathématiques.

Exemple

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

```
isNaN("Tree")
// renvoie true

isNan(56)
// renvoie false

isNaN(Number.POSITIVE_INFINITY)
// renvoie false
```

Consultez également

`NaN`, `Number.NaN`

Classe Key

Disponibilité

Flash Player 6.

Description

La classe Key est une classe de premier niveau dont les méthodes et les propriétés sont utilisables sans instructeur. Utilisez les méthodes de la classe Key pour construire une interface qui peut être contrôlée par un utilisateur possédant un clavier standard. Les propriétés de la classe Key sont des constantes représentant les touches les plus souvent utilisées pour contrôler les jeux. Pour consulter la liste complète des valeurs des codes key, consultez l'Annexe C, *Touches du clavier et valeurs de code correspondantes*, page 901.

Méthodes de la classe Key

Méthode	Description
`Key.addListener()`	Enregistre un objet pour la réception de notification lorsque les méthodes `onKeyDown` et `onKeyUp` sont invoquées.
`Key.getAscii()`	Renvoie la valeur ASCII de la dernière touche enfoncée.
`Key.getCode()`	Renvoie le code virtuel de la dernière touche enfoncée.
`Key.isDown()`	Renvoie `true` si la touche spécifiée dans le paramètre est enfoncée.
`Key.isToggled()`	Renvoie `true` si les touches Verr Num ou Verr Maj sont activées.
`Key.removeListener()`	Supprime un objet précédemment enregistré avec `Key.addListener()`.

Propriétés de la classe Key

Toutes les propriétés de la classe Key sont des constantes.

Propriété	Description
`Key.BACKSPACE`	Constante associée à la valeur de code de touche pour la touche Rappel arrière (8).
`Key.CAPSLOCK`	Constante associée à la valeur de code de touche pour la touche Verrouillage des majuscules (20).
`Key.CONTROL`	Constante associée à la valeur de code de touche pour la touche Ctrl (17).
`Key.DELETEKEY`	Constante associée à la valeur de code de touche pour la touche Suppression (46).
`Key.DOWN`	Constante associée à la valeur de code de touche pour la touche Flèche vers le bas (40).
`Key.END`	Constante associée à la valeur de code de touche pour la touche Fin (35).
`Key.ENTER`	Constante associée à la valeur de code de touche pour la touche Entrée (13).
`Key.ESCAPE`	Constante associée à la valeur de code de touche pour la touche Echap (27).
`Key.HOME`	Constante associée à la valeur de code de touche pour la touche Origine (36).
`Key.INSERT`	Constante associée à la valeur de code de touche pour la touche Insertion (45).
`Key.LEFT`	Constante associée à la valeur de code de touche pour la touche Flèche vers la gauche (37).
`Key.PGDN`	Constante associée à la valeur de code de touche pour la touche Page suivante (34).
`Key.PGUP`	Constante associée à la valeur de code de touche pour la touche Page précédente (33).
`Key.RIGHT`	Constante associée à la valeur de code de touche pour la touche Flèche vers la droite (39).

Propriété	Description
Key.SHIFT	Constante associée à la valeur de code de touche pour la touche Maj (16).
Key.SPACE	Constante associée à la valeur de code de touche pour la touche Barre d'espacement (32).
Key.TAB	Constante associée à la valeur de code de touche pour la touche Tab (9).
Key.UP	Constante associée à la valeur de code de touche pour la touche Flèche vers le haut (38).

Ecouteurs de la classe Key

Méthode	Description
Key.onKeyDown	Notifié lorsqu'une touche est enfoncée.
Key.onKeyUp	Notifié lorsqu'une touche est relâchée.

Key.addListener()

Disponibilité

Flash Player 6.

Usage

```
Key.addListener (nouvelEcouteur)
```

Paramètres

`nouvelEcouteur` Un objet avec les méthodes `onKeyDown` et `onKeyUp`.

Renvoie

Rien.

Description

Méthode : enregistre un objet pour la réception de notifications `onKeyDown` et `onKeyUp`. Lorsqu'une touche est enfoncée ou relâchée, quel que soit le focus de saisie, la méthode `onKeyDown` ou `onKeyUp` de tous les objets écouteurs enregistrés avec `addListener()` est invoquée. Plusieurs objets peuvent attendre des notifications de clavier. Si l'écouteur *`nouvelEcouteur`* est déjà enregistré, aucun changement n'a lieu.

Exemple

L'exemple suivant crée un nouvel objet d'écoute et définit une fonction pour `onKeyDown` et `onKeyUp`. La dernière ligne utilise la méthode `addListener()` pour enregistrer l'écouteur avec l'objet Key, afin qu'il puisse recevoir une notification des événements de touche enfoncée et de touche relâchée.

```
monEcouteur = new Object();
monEcouteur.onKeyDown = function () {
  trace ("Vous avez appuyé sur une touche.");
}
monEcouteur.onKeyUp = function () {
  trace ("Vous avez relâché une touche.");
}
Key.addListener(monEcouteur);
```

L'exemple suivant attribue le raccourci clavier Ctrl+7 à un bouton avec un nom d'occurrence de monBouton, et rend les informations sur le raccourci accessibles aux lecteurs d'écran (consultez `_accProps`). Dans cet exemple, lorsque vous appuyez sur Ctrl+7, la fonction `myOnPress` affiche le texte « bonjour » dans le panneau de sortie ; dans votre fichier, vous devez créer une fonction plus explicite.

```
function myOnPress() {
  trace( "bonjour" );
}

function myOnKeyDown() {
  if (Key.isDown(Key.CONTROL) && Key.getCode() == 55) // 55 est le code de
  touche de 7
  {
    Selection.setFocus( monBouton );
    monBouton.onPress();
  }
}
```

```
var monEcouteur = new Object();
monEcouteur.onKeyDown = myOnKeyDown;
Key.addListener( monEcouteur );

monBouton.onPress = myOnPress;
monBouton._accProps.shortcut = "Ctrl+F"
Accessibility.updateProperties()
```

Consultez également

`Key.getCode()`, `Key.isDown()`, `Key.onKeyDown`, `Key.onKeyUp`, `Key.removeListener()`

Key.BACKSPACE

Disponibilité

Flash Player 5.

Usage

```
Key.BACKSPACE
```

Description

Propriété : associée à la valeur de code de touche pour la touche Retour arrière (8).

Key.CAPSLOCK

Disponibilité

Flash Player 5.

Usage

```
Key.CAPSLOCK
```

Description

Propriété : associée à la valeur de code de touche pour la touche Verr Maj (20).

Key.CONTROL

Disponibilité

Flash Player 5.

Usage

```
Key.CONTROL
```

Description

Propriété : associée à la valeur de code de touche pour la touche Ctrl (17).

Key.DELETEKEY

Disponibilité

Flash Player 5.

Usage

```
Key.DELETEKEY
```

Description

Propriété : associée à la valeur de code de touche pour la touche Suppr (46).

Key.DOWN

Disponibilité

Flash Player 5.

Usage

```
Key.DOWN
```

Description

Propriété : associée à la valeur de code de touche pour la touche Flèche vers le bas (40).

Key.END

Disponibilité

Flash Player 5.

Usage

```
Key.END
```

Description

Propriété : associée à la valeur de code de touche pour la touche Fin (35).

Key.ENTER

Disponibilité

Flash Player 5.

Usage

```
Key.ENTER
```

Description

Propriété : associée à la valeur de code de touche pour la touche Entrée (13).

Key.ESCAPE

Disponibilité

Flash Player 5.

Usage

```
Key.ESCAPE
```

Description

Propriété : associée à la valeur de code de touche pour la touche Echap (27).

Key.getAscii()

Disponibilité

Flash Player 5.

Usage

```
Key.getAscii();
```

Paramètres

Aucun.

Renvoie

Un entier représentant la valeur ASCII de la dernière touche enfoncée.

Description

Méthode : renvoie le code ASCII de la dernière touche enfoncée ou relâchée. Les valeurs ASCII renvoyées sont les valeurs du clavier anglais. Par exemple, si vous appuyez sur Maj+2, `Key.getAscii()` renvoie @ sur un clavier japonais, de la même façon que sur un clavier anglais.

Key.getCode()

Disponibilité

Flash Player 5.

Usage

```
Key.getCode();
```

Paramètres

Aucun.

Renvoie

Un entier représentant le code de touche de la dernière touche enfoncée.

Description

Méthode : renvoie la valeur de code de la dernière touche enfoncée. Pour faire correspondre la valeur du code touche renvoyée avec la touche sur un clavier standard, consultez l'Annexe C, *Touches du clavier et valeurs de code correspondantes*, page 901.

Key.HOME

Disponibilité

Flash Player 5.

Usage

```
Key.HOME
```

Description

Propriété : associée à la valeur de code de touche pour la touche Origine (36).

Key.INSERT

Disponibilité

Flash Player 5.

Usage

```
Key.INSERT
```

Description

Propriété : associée à la valeur de code de touche pour la touche Insertion (45).

Key.isDown()

Disponibilité

Flash Player 5.

Usage

`Key.isDown(`*`codeDeTouche`*`)`

Paramètres

codeDeTouche La valeur de code affectée à une touche spécifique ou une propriété de classe Key associée à une touche spécifique. Pour faire correspondre la valeur du code touche renvoyée avec la touche sur un clavier standard, consultez l'Annexe C, *Touches du clavier et valeurs de code correspondantes*, page 901.

Renvoie

Une valeur booléenne.

Description

Méthode : renvoie `true` si la touche spécifiée dans *`codeDeTouche`* est enfoncée, `false` si elle ne l'est pas. Sur Macintosh, les valeurs de codes des touches Verr Maj et Verr Num sont identiques.

Exemple

Le script suivant permet à l'utilisateur de contrôler l'emplacement d'un clip.

```
onClipEvent (enterFrame) {
  if(Key.isDown(Key.RIGHT)) {
    this._x=_x+10;
  } else if (Key.isDown(Key.DOWN)) {
    this._y=_y+10;
  }
}
```

Key.isToggled()

Disponibilité

Flash Player 5.

Usage

```
Key.isToggled(codeDeTouche)
```

Paramètres

codeDeTouche Le code de touche Verr Maj (20) ou Verr Num (144).

Renvoie

Une valeur booléenne.

Description

Méthode : renvoie `true` si la touche Verr Maj ou Verr Num est activée (activée), `false` si elle ne l'est pas. Sur Macintosh, les valeurs de codes des touches Verr Maj et Verr Num sont identiques.

Key.LEFT

Disponibilité

Flash Player 5.

Usage

```
Key.LEFT
```

Description

Propriété : associée à la valeur de code de touche pour la touche Flèche vers la gauche (37).

Key.onKeyDown

Disponibilité

Flash Player 6.

Usage

unEcouteur.onKeyDown

Description

Ecouteur : notifié lorsqu'une touche est enfoncée. Pour utiliser `onKeyDown`, vous devez créer un objet d'écoute. Vous pouvez alors définir une fonction pour `onKeyDown` et utiliser `addListener()` pour enregistrer l'écouteur avec l'objet Key, comme dans l'exemple suivant :

```
unEcouteur = new Object();
unEcouteur.onKeyDown = function () { ... };
Key.addListener(unEcouteur);
```

Les écouteurs permettent à différents morceaux de code de coopérer étant donné que plusieurs écouteurs peuvent recevoir une notification concernant un seul événement.

Consultez également

`Key.addListener()`

Key.onKeyUp

Disponibilité

Flash Player 6.

Usage

unEcouteur.onKeyUp

Description

Ecouteur : notifié lorsqu'une touche est relâchée. Pour utiliser `onKeyUp`, vous devez créer un objet d'écoute. Vous pouvez alors définir une fonction pour `onKeyUp` et utiliser `addListener()` pour enregistrer l'écouteur avec l'objet Key, comme dans l'exemple suivant :

```
unEcouteur = new Object();
unEcouteur.onKeyUp = function () { ... };
Key.addListener(unEcouteur);
```

Les écouteurs permettent à différents morceaux de code de coopérer étant donné que plusieurs écouteurs peuvent recevoir une notification concernant un seul événement.

Consultez également

`Key.addListener()`

Key.PGDN

Disponibilité

Flash Player 5.

Usage

```
Key.PGDN
```

Description

Propriété : associée à la valeur de code de touche pour la touche Page suivante (34).

Key.PGUP

Disponibilité

Flash Player 5.

Usage

```
Key.PGUP
```

Description

Propriété : associée à la valeur de code de touche pour la touche Page vers le haut (33).

Key.removeListener()

Disponibilité

Flash Player 6.

Usage

```
Key.removeListener (écouteur)
```

Paramètres

écouteur Un objet.

Renvoie

Si l'*écouteur* a été correctement retiré de la méthode renvoie `true`. Si l'*écouteur* n'a pas été correctement retiré, par exemple si l'*écouteur* n'apparaissait pas dans la liste des écouteurs de l'objet Key, la méthode renvoie `false`.

Description

Méthode : retire un objet précédemment enregistré avec `Key.addListener()`.

Key.RIGHT

Disponibilité

Flash Player 5.

Usage

```
Key.RIGHT
```

Description

Propriété : associée à la valeur de code de touche pour la touche Flèche vers la droite (39).

Key.SHIFT

Disponibilité

Flash Player 5.

Usage

```
Key.SHIFT
```

Description

Propriété : associée à la valeur de code de touche pour la touche Maj (16).

Key.SPACE

Disponibilité

Flash Player 5.

Usage

```
Key.SPACE
```

Description

Propriété : associée à la valeur de code de touche pour la touche Barre d'espacement (32).

Key.TAB

Disponibilité

Flash Player 5.

Usage

```
Key.TAB
```

Description

Propriété : associée à la valeur de code de touche pour la touche Tab (9).

Key.UP

Disponibilité

Flash Player 5.

Usage

```
Key.UP
```

Description

Propriété : associée à la valeur de code de touche pour la touche Flèche vers le haut (38).

le (inférieur ou égal à – spécifique aux chaînes)

Disponibilité

Flash Player 4. Cet opérateur est déconseillé dans Flash 5 et remplacé par l'opérateur <= (inférieur ou égal à).

Usage

```
expression1 le expression2
```

Paramètres

expression1, expression2 Nombres, chaînes ou variables.

Renvoie

Rien.

Description

Opérateur (comparaison) : compare *expression1* avec *expression2* et renvoie `true` si *expression1* est inférieure ou égale à *expression2* ; sinon, renvoie `false`.

Consultez également

<= (inférieur ou égal à)

length

Disponibilité

Flash Player 4. Cette fonction, de même que toutes les fonctions de chaînes, est déconseillée dans Flash 5. Macromedia recommande l'utilisation des méthodes de la classe String et de la propriété `String.length` pour effectuer les mêmes opérations.

Usage

```
length(expression)
length(variable)
```

Paramètres

`expression` Une chaîne.

`variable` Le nom d'une variable.

Renvoie

La longueur de la chaîne spécifiée ou du nom de la variable.

Description

Fonction de chaîne : renvoie la longueur de la chaîne ou variable spécifiée.

Exemple

L'exemple suivant renvoie la valeur de la chaîne `"Bonjour"`.

```
length("Bonjour");
```

Le résultat est 5.

Consultez également

`" " (délimiteur de chaîne)`, Classe String, `String.length`

_level

Disponibilité

Flash Player 4.

Usage

```
_levelN
```

Description

Identifiant : une référence au scénario racine de _level*N*. Vous devez utiliser `loadMovieNum()` pour charger les 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 automatiquement chargé dans `_level0`. Le fichier SWF dans `_level0` définit la cadence, la couleur d'arrière-plan et la taille des images pour tous les autres fichiers SWF chargés. Les fichiers SWF sont alors empilés à des niveaux plus élevés au-dessus du fichier SWF dans `_level0`.

Vous devez affecter un niveau à chaque fichier SWF que vous chargez dans Flash Player à l'aide de l'action `loadMovieNum()`. Vous pouvez affecter des niveaux dans n'importe quel ordre. Si vous affectez un niveau qui contient déjà un fichier SWF (y compris `_level0`), le fichier SWF qui se trouve à ce niveau est purgé et remplacé par le nouveau fichier SWF.

Exemple

L'exemple suivant arrête la tête de lecture du scénario principal du fichier SWF de `_level9`.

```
_level9.stop();
```

L'exemple suivant envoie la tête de lecture du scénario principal du fichier SWF de `_level4` vers l'image 5. Le fichier SWF de `_level4` doit avoir été chargé avec une action `loadMovieNum()`.

```
_level4.gotoAndStop(5);
```

Consultez également

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

loadMovie()

Disponibilité

Flash Player 3.

Usage

```
loadMovie("url",cible [, méthode])
```

Paramètres

url L'URL absolue ou relative du fichier SWF ou JPEG à charger. Un chemin relatif doit être relatif au fichier SWF au niveau 0. Des URL absolues doivent inclure la référence au protocole, comme http:// ou file:///.

cible Un chemin vers la cible d'un clip. Le clip cible sera remplacé par l'image ou le fichier SWF chargé.

méthode Un paramètre facultatif spécifiant une méthode HTTP d'envoi des variables. Le paramètre doit être la chaîne `GET` ou `POST`. Omettez ce paramètre si aucune variable ne doit être envoyée. La méthode `GET` ajoute les variables à la fin de l'URL et est utilisée pour des petits nombres de variables. La méthode `POST` envoie les variables dans un en-tête HTTP distinct et est utilisée pour les longues chaînes de variables.

Renvoie

Rien.

Description

Fonction : charge un fichier SWF ou JPEG dans Flash Player en cours de lecture du fichier SWF d'origine.

Conseil : Si vous souhaitez suivre la progression du téléchargement, utilisez `MovieClipLoader.loadClip()` au lieu de cette fonction.

La fonction `loadMovie()` vous permet d'afficher plusieurs fichiers SWF simultanément et de basculer entre les fichiers SWF sans charger un autre document HTML. Sans la fonction `loadMovie()`, Flash Player affiche un seul fichier SWF avant de quitter.

Si vous souhaitez charger un fichier SWF ou JPEG dans un niveau spécifique, utilisez `loadMovieNum()` au lieu de `loadMovie()`.

Si un fichier SWF est chargé dans un clip cible, vous pouvez utiliser le chemin cible de ce clip pour cibler le fichier chargé. Un fichier SWF ou une image chargée dans une cible hérite de ses propriétés de position, rotation et échelle. Le coin supérieur gauche de l'image ou du fichier SWF chargé s'aligne avec le point d'alignement du clip ciblé. Ou bien, si la cible est le scénario `_root`, le coin supérieur gauche de l'image ou du fichier SWF s'aligne avec le coin supérieur gauche de la scène.

Utilisez `unloadMovie()` pour supprimer les fichiers SWF chargés avec `loadMovie()`.

Exemple

L'instruction `loadMovie()` suivante est associée à un bouton de navigation intitulé Produits. Un clip invisible se trouve sur la scène et porte le nom d'occurrence `zoneCible`. La fonction `loadMovie()` utilise ce clip comme paramètre cible pour charger les produits du fichier SWF à la position correcte sur la Scène.

```
on(release) {
  loadMovie("produits.swf",_root.zoneCible);
}
```

L'exemple suivant charge une image JPEG à partir du même répertoire que le fichier SWF qui appelle la fonction `loadMovie()` :

```
loadMovie("image45.jpeg", "notreClip");
```

Consultez également

`_level`, `loadMovieNum()`, `MovieClipLoader.loadClip()`, `unloadMovie()`

loadMovieNum()

Disponibilité

Flash Player 4. Les fichiers Flash 4 ouverts dans Flash 5 ou ultérieur sont convertis de manière à utiliser la syntaxe correcte.

Usage

```
loadMovieNum("url",niveau[, variables])
```

Paramètres

`url` Une URL absolue ou relative du fichier SWF ou JPEG à charger. Un chemin relatif doit être relatif au fichier SWF au niveau 0. Pour une utilisation dans Flash Player autonome ou pour un test en mode test d'animation dans l'application auteur Flash, tous les fichiers SWF doivent être stockés dans le même répertoire et les noms de fichier ne peuvent pas inclure des spécifications de répertoire ou de lecteur de disque.

`niveau` Un entier spécifiant le niveau Flash Player auquel le fichier SWF sera chargé.

`variables` Un paramètre facultatif spécifiant une méthode HTTP d'envoi des variables. Le paramètre doit être la chaîne `GET` ou `POST`. Omettez ce paramètre si aucune variable ne doit être envoyée. La méthode `GET` ajoute les variables à la fin de l'URL et est utilisée pour des petits nombres de variables. La méthode `POST` envoie les variables dans un en-tête HTTP distinct et est utilisée pour les longues chaînes de variables.

Renvoie

Rien.

Description

Fonction : charge un fichier SWF ou JPEG dans un niveau de Flash Player en cours de lecture du fichier SWF d'origine.

Conseil : Si vous souhaitez suivre la progression du téléchargement, utilisez `MovieClipLoader.loadClip()` au lieu de cette fonction.

Normalement, Flash Player affiche un seul fichier SWF avant de quitter. L'action `loadMovie()` vous permet d'afficher plusieurs fichiers SWF simultanément et de basculer entre les fichiers SWF sans charger un autre document HTML.

Si vous souhaitez spécifier une cible au lieu d'un niveau, utilisez `loadMovie()` au lieu de `loadMovieNum()`.

Flash Player a un ordre d'empilement des niveaux qui commence au niveau 0. Ces niveaux sont comme des calques : ils sont transparents, à l'exception des objets qui se trouvent sur chaque niveau. Lorsque vous utilisez `loadMovieNum()`, vous devez spécifier un niveau Flash Player dans lequel charger le fichier SWF. Lorsqu'un fichier SWF est chargé dans un niveau, vous pouvez utiliser la syntaxe `_levelN` pour cibler le fichier SWF, où *N* est le numéro de niveau.

Lorsque vous chargez un fichier SWF, vous pouvez spécifier un quelconque numéro et charger des fichiers SWF dans un niveau dans lequel un fichier SWF a déjà été chargé. Si tel est le cas, le nouveau fichier SWF remplacera le fichier SWF existant. Si vous chargez un fichier SWF dans le niveau 0, tous les niveaux de Flash Player sont vidés et le niveau 0 est remplacé par le nouveau fichier. Le fichier SWF dans le niveau 0 définit la cadence, la couleur d'arrière-plan et la taille des images pour tous les autres fichiers SWF chargés.

L'action `loadMovieNum()` vous permet également de charger des fichiers JPEG dans un fichier SWF en cours de lecture. Le coin supérieur gauche de l'image s'aligne, aussi bien pour les images que pour les fichiers SWF, avec le coin supérieur gauche de la scène lors du chargement du fichier. De plus, dans les deux cas, le fichier chargé hérite de la rotation et de l'échelle, le contenu original étant écrasé.

Utilisez `unloadMovieNum()` pour supprimer des fichiers SWF ou des images chargés avec `loadMovieNum()`.

Exemple

Cet exemple charge l'image JPEG image45.jpg dans le niveau 2 de Flash Player.

```
loadMovieNum("http://www.blag.com/image45.jpg", 2);
```

Consultez également

`loadMovie()`, `unloadMovieNum()`, `_level`

loadVariables()

Disponibilité

Flash Player 4 ; comportement modifié dans Flash Player 7.

Usage

```
loadVariables ("url" , cible [, variables])
```

Paramètres

url Une URL absolue ou relative où se trouvent les variables. Si le fichier SWF résultant de cet appel est ouvert dans un navigateur web, l'*url* doit être du même domaine que le fichier SWF ; pour plus de détails, consultez la description ci-dessous.

cible Le chemin cible d'un clip qui reçoit les variables chargées.

variables Un paramètre facultatif spécifiant une méthode HTTP d'envoi des variables. Le paramètre doit être la chaîne `GET` ou `POST`. Omettez ce paramètre si aucune variable ne doit être envoyée. La méthode `GET` ajoute les variables à la fin de l'URL et est utilisée pour des petits nombres de variables. La méthode `POST` envoie les variables dans un en-tête HTTP distinct et est utilisée pour les longues chaînes de variables.

Renvoie

Rien.

Description

Fonction : lit des données depuis un fichier externe, tel qu'un fichier texte ou du texte généré par un script CGI, ASP, PHP ou Perl, et définit les valeurs des variables dans un clip cible. Cette action peut également être utilisée pour affecter de nouvelles valeurs aux variables du fichier SWF actif.

Le texte de l'URL spécifiée doit être au format standard MIME *application/x-www-form-urlencoded* (un format standard utilisé par les scripts CGI). Tout nombre de variables peut être spécifié. Par exemple, la séquence suivante définit plusieurs variables :

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

Dans des fichiers SWF exécutés dans une version du lecteur antérieure à Flash Player 7, l'*url* doit être du même super-domaine que le fichier SWF résultant de cet appel. Par exemple, un fichier SWF à l'adresse www.Domaine.com peut charger des variables d'un fichier SWF à l'adresse store.Domaine.com car les deux fichiers sont du même super-domaine de Domaine.com.

Dans des fichiers SWF d'une quelconque version exécutée dans Flash Player 7 ou ultérieur, l'*url* doit être exactement du même domaine (consultez *Fonctions de sécurité de Flash Player*, page 199). Par exemple, un fichier SWF à l'adresse www.Domaine.com peut uniquement charger des variables de fichiers SWF également à l'adresse www.Domaine.com. Si vous souhaitez charger des variables d'un autre domaine, vous pouvez placer un *fichier de régulation inter-domaines* sur le serveur hébergeant le fichier SWF auquel vous accédez actuellement. Pour plus d'informations, consultez *A propos de l'autorisation de chargement de données inter-domaines*, page 201.

Si vous souhaitez charger des variables dans un niveau spécifique, utilisez `loadvariablesNum()` au lieu de `loadVariables()`.

Exemple

Cet exemple charge des informations depuis un fichier texte vers des champs de texte dans le clip `varCible` du scénario principal. Les noms de variables des champs de texte doivent correspondre aux noms de variables du fichier données.txt.

```
on(release) {
  loadVariables("donnees.txt", "_root.varCible");
}
```

Consultez également

`loadvariablesNum()`, `loadMovie()`, `loadMovieNum()`, `getURL()`, `MovieClip.loadMovie`, `MovieClip.loadVariables()`

loadvariablesNum()

Disponibilité

Flash Player 4. Les fichiers Flash 4 ouverts dans Flash 5 ou ultérieur sont convertis de manière à utiliser la syntaxe correcte. Comportement modifié dans Flash Player 7.

Usage

```
loadVariablesNum("url",niveau[, variables])
```

Paramètres

`url` Une URL absolue ou relative où se trouvent les variables. Si le fichier SWF résultant de cet appel est ouvert dans un navigateur web, l'*`url`* doit être du même domaine que le fichier SWF ; pour plus de détails, consultez la description ci-dessous.

`niveau` Un entier spécifiant le niveau Flash Player de réception des variables.

`variables` Un paramètre facultatif spécifiant une méthode HTTP d'envoi des variables. Le paramètre doit être la chaîne `GET` ou `POST`. Omettez ce paramètre si aucune variable ne doit être envoyée. La méthode `GET` ajoute les variables à la fin de l'URL et est utilisée pour des petits nombres de variables. La méthode `POST` envoie les variables dans un en-tête HTTP distinct et est utilisée pour les longues chaînes de variables.

Renvoie

Rien.

Description

Fonction : lit des données depuis un fichier externe, tel qu'un fichier texte ou un texte généré par un script CGI, ASP, PHP ou Perl, et définit les valeurs des variables dans un niveau Flash Player. Vous pouvez également utiliser cette fonction pour mettre à jour les variables du fichier SWF actif à l'aide de 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). Tout nombre de variables peut être spécifié. Par exemple, la séquence suivante définit plusieurs variables :

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

Dans des fichiers SWF exécutés dans une version du lecteur antérieure à Flash Player 7, l'*url* doit être du même super-domaine que le fichier SWF résultant de cet appel. Par exemple, un fichier SWF à l'adresse www.Domaine.com peut charger des variables d'un fichier SWF à l'adresse store.Domaine.com car les deux fichiers sont du même super-domaine de Domaine.com.

Dans des fichiers SWF d'une quelconque version exécutée dans Flash Player 7 ou ultérieur, l'*url* doit être exactement du même domaine (consultez *Fonctions de sécurité de Flash Player*, page 199). Par exemple, un fichier SWF à l'adresse www.Domaine.com peut uniquement charger des variables de fichiers SWF également à l'adresse www.Domaine.com. Si vous souhaitez charger des variables d'un autre domaine, vous pouvez placer un *fichier de régulation inter-domaines* sur le serveur hébergeant le fichier SWF auquel vous accédez actuellement. Pour plus d'informations, consultez *A propos de l'autorisation de chargement de données inter-domaines*, page 201.

Si vous souhaitez charger des variables dans un clip spécifique, utilisez `loadVariables()` au lieu de `loadVariablesNum()`.

Exemple

Cet exemple charge des informations depuis un fichier texte vers des champs de texte dans le scénario principal du fichier SWF (niveau 0) de Flash Player. Les noms de variables des champs de texte doivent correspondre aux noms de variables du fichier données.txt.

```
on(release) {
  loadVariablesNum("donnees.txt", 0);
}
```

Consultez également

`getURL()`, `loadMovie()`, `loadMovieNum()`, `loadVariables()`, `MovieClip.loadMovie`, `MovieClip.loadVariables()`

Classe LoadVars

Disponibilité

Flash Player 6.

Description

La classe LoadVars constitue une alternative à la fonction `loadVariables()` pour transférer des variables entre une application Flash et un serveur.

Vous pouvez utiliser cette classe pour obtenir la vérification du chargement réussi des données, les indications de progression et les données de flux pendant le téléchargement. La classe LoadVars fonctionne de manière semblable à Classe XML : elle utilise les méthodes `load()`, `send()` et `sendAndLoad()` pour communiquer avec un serveur. La principale différence entre la classe LoadVars et la classe XML est que LoadVars transfère des paires nom et valeur ActionScript, au lieu de l'arborescence DOM XML stockée dans l'objet XML.

La classe LoadVars est soumise aux mêmes restrictions de sécurité que la classe XML.

Méthodes de la classe LoadVars

Méthode	Description
`LoadVars.addRequestHeader()`	Ajoute ou modifie les en-têtes HTTP pour les opérations `POST`.
`LoadVars.getBytesLoaded()`	Renvoie le nombre d'octets téléchargés par `LoadVars.load()` ou `LoadVars.sendAndLoad()`.
`LoadVars.getBytesTotal()`	Renvoie le nombre total d'octets téléchargés par une opération `load` ou une méthode `sendAndLoad`.
`LoadVars.load()`	Télécharge des variables à partir d'une URL spécifiée.
`LoadVars.send()`	Publie des variables d'un objet LoadVars à une URL.
`LoadVars.sendAndLoad()`	Publie les variables d'un objet LoadVars à une URL et télécharge la réponse du serveur vers un objet cible.
`LoadVars.toString()`	Renvoie une chaîne encodée URL qui contient toutes les variables énumérables de l'objet LoadVars.

Propriétés de la classe LoadVars

Propriété	Description
`LoadVars.contentType`	Indique le type MIME des données.
`LoadVars.loaded`	Une valeur booléenne indiquant si une opération `load` ou `sendAndLoad` est terminée.

Gestionnaires d'événement de la classe LoadVars

Gestionnaire d'événement	Description
`LoadVars.onData`	Invoqué lorsque les données ont été complètement téléchargées du serveur ou lorsqu'une erreur se produit au cours de ce téléchargement.
`LoadVars.onLoad`	Invoqué lorsqu'une opération `load` ou `sendAndLoad` est terminée.

Constructeur de la classe LoadVars

Disponibilité

Flash Player 6.

Usage

```
new LoadVars()
```

Paramètres

Aucun.

Renvoie

Rien.

Description

Constructeur : crée un objet LoadVars Vous pouvez ensuite utiliser les méthodes de cet objet LoadVars pour envoyer et charger des données.

Exemple

L'exemple suivant crée un objet LoadVars nommé `mes_lv` :

```
var mes_lv = new LoadVars();
```

LoadVars.addRequestHeader()

Disponibilité

Flash Player 6.

Usage

```
mon_lv.addRequestHeader(nomDentête, valeurDentête)
mon_lv.addRequestHeader(["nomDentête_1", "valeurDentête_1" ... "nomDentête_n",
  "valeurDentête_n"])
```

Paramètres

nomDentête Un nom d'en-tête de requête HTTP.

valeurDentête La valeur associée à *nomDentête*.

Renvoie

Rien.

Description

Méthode : ajoute ou modifie les en-têtes de requête HTTP (telles que `Content-type` ou `SOAPAction`) envoyés avec les actions `POST`. Dans la première utilisation, vous transmettez deux chaînes à la méthode : *nomDentête* et *valeurDentête*. Dans la seconde utilisation, vous transmettez un tableau de chaînes, alternant noms et valeurs d'en-têtes.

Si des appels multiples sont définis pour un seul et même nom d'en-tête, chaque valeur successive remplace la valeur définie dans le précédent appel.

Les en-têtes HTTP standard suivantes *ne peuvent pas* être ajoutées ou modifiées dans 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`.

Exemple

Cet exemple ajoute un en-tête HTTP personnalisé nommé `SOAPAction` avec une valeur `Foo` à l'objet `mon_lv`.

```
mon_lv.addRequestHeader("SOAPAction", "'Foo'");
```

L'exemple suivant crée un tableau nommé `en-têtes` contenant deux en-têtes secondaires et leurs valeurs associées. Le tableau est transmis en tant qu'argument à `addRequestHeader()`.

```
var en-têtes = ["Content-type", "texte/normal", "X-ClientAppVersion", "2.0"];
mon_lv.addRequestHeader(headers);
```

Voir aussi

`XML.addRequestHeader()`

LoadVars.contentType

Disponibilité

Flash Player 6.

Usage

mes_lv.contentType

Description

Propriété : le type MIME envoyé au serveur lorsque vous appelez LoadVars.send()ou LoadVars.sendAndLoad(). La valeur par défaut est application/x-www-urlform-encoded.

Consultez également

LoadVars.send(), LoadVars.sendAndLoad()

LoadVars.getBytesLoaded()

Disponibilité

Flash Player 6.

Usage

mes_lv.getBytesLoaded()

Paramètres

Aucun.

Renvoie

Un entier.

Description

Méthode : renvoie le nombre d'octets téléchargés par LoadVars.load() ou LoadVars.sendAndLoad(). Cette méthode renvoie undefined si aucune opération load n'est en cours ou si une opération load n'a pas encore été initiée.

LoadVars.getBytesTotal()

Disponibilité

Flash Player 6.

Usage

```
mes_lv.getBytesTotal()
```

Paramètres

Aucun.

Renvoie

Un entier.

Description

Méthode : renvoie le nombre total d'octets téléchargés par `LoadVars.load()` ou `LoadVars.sendAndLoad()`. Cette méthode renvoie `undefined` si aucune opération load n'est en cours ou si une opération load n'a pas encore été initiée. Cette méthode renvoie également `undefined` si le nombre total d'octets ne peut pas être déterminé (par exemple, si le téléchargement a été initié mais que le serveur n'a pas transmis de longueur de contenu HTTP).

LoadVars.load()

Disponibilité

Flash Player 6 ; comportement modifié dans Flash Player 7.

Usage

```
mes_lv.load(url)
```

Paramètres

url L'URL où se trouvent les variables à télécharger. Si le fichier SWF résultant de cet appel est ouvert dans un navigateur web, l'*url* doit être du même domaine que le fichier SWF ; pour plus de détails, consultez la description ci-dessous.

Renvoie

Une chaîne.

Description

Méthode : télécharge des variables depuis l'URL spécifiée, analyse les données et place les variables résultantes dans *mes_lv*. Les propriétés de *mes_lv* avec les mêmes noms que les variables téléchargées sont supprimées. Les propriétés de *mes_lv* avec des noms différents des variables téléchargées ne sont pas supprimées. Il s'agit d'une action asynchrone.

Les données téléchargées doivent être au format de contenu MIME *application/x-www-form-urlencoded*. Il s'agit du même format que celui utilisé par `loadVariables()`.

Dans des fichiers SWF exécutés dans une version du lecteur antérieure à Flash Player 7, l'*url* doit être du même super-domaine que le fichier SWF résultant de cet appel. Par exemple, un fichier SWF à l'adresse www.Domaine.com peut charger des variables d'un fichier SWF à l'adresse store.Domaine.com car les deux fichiers sont du même super-domaine de Domaine.com.

Dans des fichiers SWF d'une quelconque version exécutée dans Flash Player 7 ou ultérieur, l'*url* doit être exactement du même domaine (consultez *Fonctions de sécurité de Flash Player*, page 199). Par exemple, un fichier SWF à l'adresse www.Domaine.com peut uniquement charger des variables de fichiers SWF également à l'adresse www.Domaine.com. Si vous souhaitez charger des variables d'un autre domaine, vous pouvez placer un *fichier de régulation inter-domaines* sur le serveur hébergeant le fichier SWF auquel vous accédez actuellement. Pour plus d'informations, consultez *A propos de l'autorisation de chargement de données inter-domaines*, page 201.

De même, dans des fichiers publiés pour Flash Player 7, la sensibilité à la casse (consultez *Hauteur de casse*, page 32) est prise en charge pour les variables externes chargées à l'aide de `LoadVars.load()`.

Cette méthode est similaire à `XML.load()`.

LoadVars.loaded

Disponibilité

Flash Player 6.

Usage

```
mes_lv.loaded
```

Description

Propriété : undefined par défaut. Lorsqu'une opération `LoadVars.load()` ou `LoadVars.sendAndLoad()` est initiée, la propriété `loaded` est définie sur la valeur `false` ; lorsque l'opération est terminée, la propriété `loaded` est définie sur la valeur `true`. Si une opération de chargement n'est pas encore terminée ou a échoué avec une erreur, la propriété `loaded` reste définie sur la valeur `false`.

Cette propriété est similaire à la propriété `XML.loaded`.

LoadVars.onData

Disponibilité

Flash Player 6.

Usage

```
mon_lv.onData = function(src) {
  // vos instructions
}
```

Paramètres

src Les données brutes (non analysées) à partir de l'appel d'une méthode `LoadVars.load()` ou `LoadVars.sendAndLoad()`.

Renvoie

Rien.

Description

Gestionnaire d'événement : invoqué lorsque les données ont été complètement téléchargées du serveur ou lorsqu'une erreur se produit au cours de ce téléchargement. Ce gestionnaire est invoqué avant l'analyse des données ; ainsi, il peut être utilisé pour appeler une routine d'analyse personnalisée au lieu d'une routine intégrée dans Flash Player. La valeur du paramètre *src* transmise à la fonction affectée à `LoadVars.onData` peut être soit `undefined`, soit une chaîne contenant les paires nom-valeur encodées URL téléchargées depuis le serveur. Si la valeur renvoyée est `undefined`, une erreur a eu lieu pendant le téléchargement à partir du serveur.

L'implémentation par défaut de `LoadVars.onData` invoque `LoadVars.onLoad`. Vous pouvez écraser cette implémentation par défaut en affectant une fonction personnalisée à `LoadVars.onData`, mais `LoadVars.onLoad` ne sera plus appelé à moins que vous ne l'appeliez dans votre implémentation de `LoadVars.onData`.

LoadVars.onLoad

Disponibilité

Flash Player 6.

Usage

```
mon_lv.onLoad = function(success) {
  // vos instructions
}
```

Paramètres

succès Le paramètre indique si l'opération de chargement s'est déroulée avec succès (`true`) ou non (`false`).

Renvoie

Une valeur booléenne.

Description

Gestionnaire d'événement : invoqué lorsqu'une opération `LoadVars.load()` ou `LoadVars.sendAndLoad()` est terminée. Si l'opération est réussi, *mes_lv* est renseigné par des variables téléchargées par l'opération ; ces variables sont disponibles lorsque ce gestionnaire est invoqué.

Par défaut, ce gestionnaire n'est pas défini.

Cette méthode est similaire à `XML.onLoad`.

LoadVars.send()

Disponibilité

Flash Player 6.

Usage

```
mes_lv.send(url[,cible, méthode] )
```

Paramètres

url L'URL à laquelle télécharger les variables.

cible La fenêtre du navigateur dans laquelle les réponses seront affichées.

méthode La méthode `GET` ou `POST` du protocole HTTP.

Renvoie

Une chaîne.

Description

Méthode : envoie les variables de l'objet *mes_lv* à l'URL spécifiée. Toutes les variables énumérables dans *mes_lv* sont concaténées dans une chaîne au format *application/x-www-form-urlencoded* par défaut et la chaîne est envoyée à l'URL à l'aide de la méthode HTTP `POST`. Il s'agit du même format que celui utilisé par l'action `loadVariables()`. Le type de contenu MIME envoyé dans les en-têtes de requête HTTP est la valeur `mes_lv.contentType` ou la valeur par défaut *application/x-www-form-urlencoded*. La méthode `POST` est utilisée si `GET` n'est pas spécifiée.

Si le paramètre *cible* n'est pas spécifié, la réponse du serveur est affichée dans la fenêtre nommée cible du navigateur. Si le paramètre *cible* est omis, la réponse du serveur n'est pas prise en compte.

Cette méthode est similaire à `XML.send`.

LoadVars.sendAndLoad()

Disponibilité

Flash Player 6 ; comportement modifié dans Flash Player 7.

Usage

```
mes_lv.sendAndLoad(url, objetCible[,méthode])
```

Paramètres

url L'URL à laquelle télécharger les variables. Si le fichier SWF résultant de cet appel est ouvert dans un navigateur web, l'*url* doit être du même domaine que le fichier SWF ; pour plus de détails, consultez la description ci-dessous.

objetCible L'objet LoadVars qui reçoit les variables chargées.

méthode La méthode `GET` ou `POST` du protocole HTTP.

Renvoie

Une chaîne.

Description

Méthode : envoie les variables de l'objet *mes_lv* à l'URL spécifiée. La réponse du serveur est téléchargée, analysée en tant que données de variables, et les variables résultantes sont placées dans l'objet *objetCible*.

Les variables sont envoyées de la même façon que `LoadVars.send()`. Les variables sont téléchargées dans *objetCible* de la même façon que `LoadVars.load()`.

Dans des fichiers SWF exécutés dans une version du lecteur antérieure à Flash Player 7, l'*url* doit être du même super-domaine que le fichier SWF résultant de cet appel. Par exemple, un fichier SWF à l'adresse www.Domaine.com peut charger des variables d'un fichier SWF à l'adresse store.Domaine.com car les deux fichiers sont du même super-domaine de Domaine.com.

Dans des fichiers SWF d'une quelconque version exécutée dans Flash Player 7 ou ultérieur, l'*url* doit être exactement du même domaine (consultez *Fonctions de sécurité de Flash Player*, page 199). Par exemple, un fichier SWF à l'adresse www.Domaine.com peut uniquement charger des variables de fichiers SWF également à l'adresse www.Domaine.com. Si vous souhaitez charger des variables d'un autre domaine, vous pouvez placer un *fichier de régulation inter-domaines* sur le serveur hébergeant le fichier SWF auquel vous accédez actuellement. Pour plus d'informations, consultez *A propos de l'autorisation de chargement de données inter-domaines*, page 201.

Cette méthode est similaire à `XML.sendAndLoad`.

LoadVars.toString()

Disponibilité

Flash Player 6.

Usage

```
mes_lv.toString()
```

Paramètres

Aucun.

Renvoie

Une chaîne.

Description

Méthode : renvoie une chaîne contenant toutes les variables énumérables dans `mes_lv`, dans le contenu MIME encodant *application/x-www-form-urlencoded*.

Exemple

```
var mesVars = new LoadVars();
mesVars.nom = "Gary";
mesVars.age = 26;
trace (mesVars.toString());
// donnerait le résultat
//nom=Gary&age=26
```

Classe LocalConnection

Disponibilité

Flash Player 6.

Description

La classe LocalConnection permet de développer des fichiers SWF pouvant échanger des instructions sans l'utilisation de `fscommand()` ou JavaScript. Les objets LocalConnection ne peuvent communiquer qu'entre des fichiers SWF exécutées sur le même ordinateur client, mais ils peuvent être exécutés dans deux applications différentes, par exemple, un fichier SWF exécuté dans un navigateur et un autre exécuté dans une projection. Vous pouvez utiliser les objets LocalConnection pour envoyer et recevoir des données au sein d'un seul et même fichier SWF, mais il ne s'agit pas là d'une implémentation standard ; tous les exemples contenus dans cette section illustrent la communication entre les différents fichiers SWF.

Les principales méthodes utilisées pour envoyer et recevoir des données sont `LocalConnection.send()` et `LocalConnection.connect()`. De la façon la plus simple, votre code implémente les commandes suivantes ; notez que les commandes `LocalConnection.send()` et `LocalConnection.connect()` spécifient le même nom de connexion, `nom_lc` :

```
// Code dans l'animation de réception
Réception_lc = new LocalConnection();
Réception_lc.methodToExecute = function(param1, param2)
{
  // Code à exécuter
}
Réception_lc.connect("nom_lc");
// Code dans l'animation d'envoi
Envoi_lc = new LocalConnection();
Envoi_lc.send("nom_lc", "méthodeAExécuter", donnée1, donnée2)
```

La façon la plus simple d'utiliser un objet LocalConnection est de n'autoriser la communication qu'entre les objets LocalConnection situés dans le même domaine, puisque vous n'aurez pas à vous occuper des questions de sécurité. Cependant, si vous devez autoriser la communication entre les domaines, vous disposez de plusieurs façons d'implémenter les mesures de sécurité. Pour plus d'informations, consultez la discussion concernant le paramètre *nomDeConnexion* dans `LocalConnection.send()` et les entrées `LocalConnection.allowDomain` et `LocalConnection.domain()`.

Méthodes de la classe LocalConnection

Méthode	Description
`LocalConnection.close()`	Ferme (déconnecte) l'objet LocalConnection.
`LocalConnection.connect()`	Prépare l'objet LocalConnection à recevoir les commandes d'une commande `LocalConnection.send()`.
`LocalConnection.domain()`	Renvoie une chaîne représentant le super-domaine de l'emplacement du fichier SWF actuel.
`LocalConnection.send()`	Invoque une méthode sur un objet LocalConnection spécifié.

Gestionnaires d'événement de la classe LocalConnection

Gestionnaire d'événement	Description
`LocalConnection.allowDomain`	Invoque si l'objet actuel LocalConnection spécifié (de réception) reçoit une requête pour invoquer une méthode à partir d'un objet d'envoi LocalConnection.
`LocalConnection.allowInsecureDomain`	Invoqué si l'objet LocalConnection (de réception) actuel, qui se trouve dans un fichier SWF hébergé dans un domaine utilisant un protocole sécurisé (HTTPS), reçoit une requête pour invoquer une méthode à partir de l'objet LocalConnection d'envoi qui se trouve dans un fichier SWF hébergé sur un protocole non sécurisé.
`LocalConnection.onStatus`	Invoqué après qu'un objet d'envoi LocalConnection ait essayé d'envoyer une commande à un objet LocalConnection de réception.

Constructeur de la classe LocalConnection

Disponibilité

Flash Player 6.

Usage

```
new LocalConnection()
```

Paramètres

Aucun.

Renvoie

Rien.

Description

Constructeur : crée un objet LocalConnection

Exemple

L'exemple suivant montre comment un fichier SWF d'envoi et de réception crée des objets LocalConnnection. Notez que les deux fichiers SWF peuvent utiliser le même nom ou des noms différents pour leurs objets LocalConnection respectifs. Dans cet exemple, ils utilisent le même nom, `ma_lc`.

```
// Code dans le fichier SWF de réception
ma_lc = new LocalConnection();
ma_lc.uneMéthode = function()
  // Vos instructions
}
ma_lc.connect("nomDeConnexion");

// Code dans le fichier SWF d'envoi
ma_lc = new LocalConnection();
ma_lc.send("nomDeConnexion", "uneMéthode");
```

Consultez également

`LocalConnection.connect()`, `LocalConnection.send()`

LocalConnection.allowDomain

Disponibilité

Flash Player 6 ; comportement modifié dans Flash Player 7.

Usage

```
Réception_lc.allowDomain = function([domaineDenvoi]) {
  // Vos instructions renvoient true ou false
}
```

Paramètres

domaineDenvoi Un paramètre facultatif spécifiant le domaine du fichier SWF contenant l'objet d'envoi LocalConnection.

Renvoie

Rien.

Description

Gestionnaire d'événement : invoqué dès que *Réception_lc* reçoit une requête pour invoquer une méthode à partir d'un objet d'envoi LocalConnection. Flash attend le code que vous implémentez dans ce gestionnaire pour renvoyer une valeur booléenne `true` ou `false`. Si le gestionnaire ne revoie pas `true`, la requête de l'objet d'envoi est ignorée et la méthode n'est pas invoquée.

Utilisez cette commande pour autoriser explicitement les objets LocalConnection des domaines spécifiés, ou de tout domaine, à exécuter les méthodes de réception de l'objet LocalConnection. Si vous ne déclarez pas le paramètre *domaineDenvoi*, vous aurez probablement à accepter les commandes de tous les domaines ; le code de votre gestionnaire sera tout simplement `return true`. Si vous déclarez *domaineDenvoi*, vous aurez probablement à comparer la valeur de *domaineDenvoi* avec les domaines dont vous voulez accepter les commandes. Les exemples suivants illustrent ces deux implémentations.

Dans des fichiers exécutés dans Flash Player 6, le paramètre *domaineDenvoi* contient le super-domaine de l'appelant. Dans des fichiers exécutés dans Flash Player 7 ou ultérieur, le paramètre *domaineDenvoi* contient le domaine exact de l'appelant. Dans ce dernier cas, pour autoriser l'accès par des fichiers SWF hébergés à l'adresse www.domaine.com ou store.domaine.com, vous devez autoriser explicitement l'accès à partir des deux domaines.

```
// Pour Flash Player 6
Réception_lc.allowDomain = function(domaineDenvoi) {
  return(domaineDenvoi=="domaine.com");
}
// Commandes correspondantes pour autoriser l'accès à partir de fichiers SWF
// exécutés dans Flash Player 7 ou ultérieur
Réception_lc.allowDomain = function(domaineDenvoi) {
  return(domaineDenvoi=="www.domaine.com");
    domaineDenvoi=="store.domaine.com");
}
```

De même, pour des fichiers exécutés dans Flash Player 7 ou ultérieur, vous ne pouvez pas utiliser cette méthode pour autoriser des fichiers SWF hébergés à l'aide d'un protocole sécurisé (HTTPS) de permettre l'accès à partir de fichiers SWF hébergés par des protocoles non sécurisés ; vous devez plutôt utiliser le gestionnaire d'événement `LocalConnection.allowInsecureDomain`.

Exemple

L'exemple suivant montre comment un objet LocalConnection dans un fichier SWF de réception peut autoriser les fichiers SWF de tout domaine à invoquer ses méthodes. Comparez ceci à l'exemple de `LocalConnection.connect()`, dans lequel seuls les fichiers SWF du même domaine peuvent invoquer la méthode `Trace` dans le fichier SWF de réception. Pour une discussion concernant l'utilisation du trait de soulignement (_) dans le nom de connexion, consultez `LocalConnection.send()`.

```
var uneConnexionLocale = new LocalConnection();
uneConnexionLocale.Trace = function(uneChaîne)
{
  unChampDeTexte = unChampDeTexte + uneChaîne + newline;
}

uneConnexionLocale.allowDomain = function() {
  // Tout domaine peut invoquer des méthodes sur cet objet LocalConnection
  return true;
}

uneConnexionLocale.connect("_trace");
```

Dans l'exemple suivant, le fichier SWF de réception n'accepte que les commandes des fichiers SWF situés dans ceDomaine.fr ou ceDomaineLa.fr.

```
var uneConnexionLocale = new LocalConnection();
uneConnexionLocale.Trace = function(uneChaîne)
{
  unChampDeTexte = unChampDeTexte + uneChaîne + newline;
}

uneConnexionLocale.allowDomain = function(domaineDenvoi)
{
  return(domaineDenvoi=="ceDomaine.fr" || domaineDenvoi=="ceDomaineLa.fr");
}

uneConnexionLocale.connect("_trace");
```

Consultez également

`LocalConnection.connect()`, `LocalConnection.domain()`, `LocalConnection.send()`

LocalConnection.allowInsecureDomain

Disponibilité

Flash Player 7.

Usage

```
Réception_lc.allowInsecureDomain = function([domaineDenvoi]) {
  // Vos instructions renvoient true ou false
}
```

Paramètres

domaineDenvoi Un paramètre facultatif spécifiant le domaine du fichier SWF contenant l'objet d'envoi LocalConnection.

Renvoie

Rien.

Description

Gestionnaire d'événement : invoqué si *Réception_lc*, qui se trouve dans un fichier SWF hébergé dans un domaine utilisant un protocole sécurisé (HTTPS), reçoit une requête pour invoquer une méthode à partir de l'objet LocalConnection d'envoi qui se trouve dans un fichier SWF hébergé sur un protocole non sécurisé. Flash attend le code que vous implémentez dans ce gestionnaire pour renvoyer une valeur booléenne `true` ou `false`. Si le gestionnaire ne revoie pas `true`, la requête de l'objet d'envoi est ignorée et la méthode n'est pas invoquée.

Par défaut, les fichiers SWF hébergés utilisant le protocole HTTPS ne sont accessibles que par d'autres fichiers SWF hébergés utilisant le protocole HTTPS. Cette implémentation garantit l'intégrité fournie par le protocole HTTPS.

L'utilisation de cette méthode pour remplacer le comportement par défaut n'est pas recommandée car elle compromet la sécurité HTTPS. Vous pouvez devoir cependant l'utiliser, par exemple, si vous devez autoriser l'accès à des fichiers HTTPS publiés pour Flash Player 7 ou ultérieur à partir de fichiers HTTP publiés pour Flash Player 6.

Un fichier SWF publié pour Flash Player 6 peut utiliser le gestionnaire d'événement `LocalConnection.allowDomain` pour autoriser un accès HTTP à HTTPS. Cependant, étant donné que la sécurité est implémentée différemment dans Flash Player 7, vous devez utiliser la méthode `LocalConnection.allowInsecureDomain()` pour autoriser un tel accès dans des fichiers SWF publiés pour Flash Player 7 ou ultérieur.

Consultez également

`LocalConnection.allowDomain`, `LocalConnection.connect()`

LocalConnection.close()

Disponibilité

Flash Player 6.

Usage

Réception_lc.close

Paramètres

Aucun.

Renvoie

Rien.

Description

Méthode : ferme (déconnecte) un objet LocalConnection. Emettez cette commande lorsque vous ne voulez plus que l'objet accepte les commandes, par exemple, lorsque vous souhaitez émettre une commande LocalConnection.connect() à l'aide du même paramètre *nomDeConnexion* dans un autre fichier SWF.

Consultez également

LocalConnection.connect()

LocalConnection.connect()

Disponibilité

Flash Player 6.

Usage

Réception_lc.connect(*nomDeConnexion*)

Paramètres

nomDeConnexion Une chaîne qui correspond au nom de connexion spécifié dans la commande LocalConnection.send() qui veut communiquer avec *Réception_lc*.

Renvoie

La valeur booléenne est true si aucun autre traitement en cours sur la même machine client n'a déjà émis cette commande à l'aide de la même valeur pour le paramètre *nomDeConnexion* ; dans le cas contraire, la valeur est false.

Description

Méthode : prépare un objet LocalConnection pour recevoir les commandes à partir d'une commande LocalConnection.send() (nommée « objet d'envoi LocalConnection »). L'objet utilisé avec cette commande est nommé « objet de réception LocalConnection ». Les objets de réception et d'envoi doivent être exécutés sur la même machine client.

Assurez-vous de définir les méthodes liées à *Réception_lc* avant d'appeler cette méthode, comme indiqué dans les exemples de cette section.

Par défaut, Flash Player résout *nomDeConnexion* en une valeur "*super-domaine*:nomDeConnexion", où *super-domaine* est le super-domaine du fichier SWF contenant la commande LocalConnection.connect(). Par exemple, si le fichier SWF contenant l'objet de réception LocalConnection est situé dans www.Domaine.com, *nomDeConnexion* indique "unDomain.com:nomDeConnexion". (Si un fichier SWF est situé sur l'ordinateur client, la valeur affectée à super-domaine est "localhost".)

Par défaut également, Flash Player permet à l'objet de réception LocalConnection d'accepter uniquement les commandes des objets d'envoi LocalConnection dont le nom de connexion indique également une valeur "*super-domaine*:nomDeConnexion". Ainsi, Flash facilite la communication entre les fichiers SWF situés dans le même domaine.

Si vous implémentez une communication seulement entre les fichiers SWF du même domaine, spécifiez une chaîne pour *nomDeConnexion* qui ne commence pas par un trait de soulignement (_) et qui n'indique aucun nom de domaine (par exemple "monDomaine:nomDeConnexion"). Utilisez la même chaîne dans la commande LocalConnection.connect(*nomDeConnexion*).

Si vous implémentez une communication entre des fichiers SWF situés dans différents domaines, consultez la section relative à *nomDeConnexion* dans LocalConnection.send() et les entrées LocalConnection.allowDomain et LocalConnection.domain().

Exemple

L'exemple suivant illustre comment un fichier SWF situé dans un domaine particulier peut invoquer une méthode nommée `Trace` dans un fichier SWF de réception du même domaine. La réception d'un fichier SWF fonctionne comme une fenêtre trace pour le fichier SWF d'envoi ; elle contient deux méthodes que les autres fichiers SWF peuvent appeler `Trace` et `Clear`. Les boutons enfoncés dans les fichiers SWF d'envoi appellent ces méthodes avec les paramètres spécifiés.

```
// Fichier SWF de réception
var uneConnexionLocale = new LocalConnection();
uneConnexionLocale.Trace = function(uneChaîne)
{
  unChampDeTexte = unChampDeTexte + uneChaîne + newline;
}
uneConnexionLocale.allowDomain = function() {
{
  unChampDeTexte = "";
}
uneConnexionLocale.connect("trace");
stop();
```

Le fichier SWF 1 contient le code suivant lié à un bouton étiqueté AppuyezIci. Lorsque vous appuyez sur le bouton, vous voyez apparaître la phrase « Le bouton a été enfoncé » dans le fichier SWF de réception.

```
on (press)
{
  var cl = new LocalConnection();
  cl.send("trace", "Trace", "Le bouton a été enfoncé.");
  delete cl;
}
```

Le fichier SWF 2 contient un champ de texte de saisie avec un nom de variable `monTexte` et le code suivant associé à un bouton nommé Copier. Lorsque vous tapez du texte puis enfoncez le bouton, vous voyez le texte tapé apparaître dans le fichier SWF de réception.

```
on (press)
{
  _parent.lc.send("trace", "Trace", _parent.monTexte);
  _parent.monTexte = "";
}
```

Le fichier SWF 3 contient le code suivant lié à un bouton étiqueté Clear. Lorsque vous enfoncez le bouton, le contenu de la fenêtre trace dans le fichier SWF de réception est effacé.

```
on (press)
{
  var cl = new LocalConnection();
  cl.send("trace", "Clear");
  delete cl;
}
```

Consultez également

`LocalConnection.send()`

LocalConnection.domain()

Disponibilité

Flash Player 6 ; comportement modifié dans Flash Player 7.

Usage

```
ma_lc.domain()
```

Paramètres

Aucun.

Renvoie

Une chaîne représentant le domaine de l'emplacement du fichier SWF actuel ; pour plus de détails, consultez la description ci-dessous.

Description

Méthode : renvoie une chaîne représentant le domaine de l'emplacement du fichier SWF actuel.

Dans des fichiers SWF publiés pour Flash Player 6, la chaîne renvoyée est le super-domaine du fichier SWF actuel. Par exemple, si le fichier SWF est situé sur www.macromedia.com, cette commande renvoie `macromedia.com`.

Dans des fichiers SWF publiés pour Flash Player 7 ou ultérieur, la chaîne renvoyée est le domaine exact du fichier SWF actuel. Par exemple, si le fichier SWF est situé sur www.macromedia.com, cette commande renvoie `"www.macromedia.com"`.

Si le fichier SWF actuel est un fichier local résidant sur l'ordinateur client, cette commande renvoie `"localhost"`.

La façon la plus courante d'utiliser cette commande est d'inclure le nom de domaine de l'objet d'envoi LocalConnection comme paramètre dans la méthode que vous envisagez d'invoquer dans l'objet de réception LocalConnection, ou conjointement avec `LocalConnection.allowDomain` pour accepter les commandes d'un domaine spécifique. Si vous activez la communication seulement entre des objets LocalConnection situés dans un même domaine, vous n'aurez probablement pas besoin d'utiliser cette commande.

Exemple

Dans l'exemple suivant, un fichier SWF de réception accepte uniquement les commandes des fichiers SWF situés dans le même domaine ou sur macromedia.com.

```
ma_lc = new LocalConnection();
ma_lc.allowDomain = function(domaineDenvoi)
{
  return (domaineDenvoi==this.domain() || domaineDenvoi=="macromedia.com");
}
```

Dans l'exemple suivant, un fichier SWF d'envoi situé dans votreDomaine.com invoque une méthode dans un fichier SWF de réception situé sur monDomaine.com. Le fichier SWF d'envoi inclut son nom de domaine en tant que paramètre dans la méthode qu'il invoque, de sorte que le fichier SWF de réception puisse renvoyer une valeur à un objet LocalConnection dans le domaine correct. Le fichier SWF d'envoi spécifie également qu'il n'accepte que les commandes des fichiers SWF de monDomaine.com.

Les numéros de ligne sont inclus pour référence. La séquence des événements est la suivante :

- Le fichier de réception se prépare à recevoir les commandes sur une connexion nommée `"somme"` (ligne 11). Flash Player correspond au nom de cette connexion à `"monDomaine.com:somme"` (consultez `LocalConnection.connect()`).
- Le fichier SWF d'envoi se prépare à recevoir une réponse sur l'objet LocalConnection nommé `"résultat"` (ligne 58). Il indique également qu'il n'accepte que les commandes des fichiers SWF de monDomaine.com (lignes 51 à 53).
- Le fichier SWF d'envoi invoque la méthode `uneSomme` d'une connexion nommée `"monDomaine.com:somme"` (ligne 59) et transmet les paramètres suivants : son domaine (`lc.domain()`), le nom de la connexion qui doit recevoir la réponse (`"résultat"`) et les valeurs à utiliser par `uneSomme` (123 et 456).
- La méthode `uneSomme` (ligne 6) est invoquée avec les valeurs suivantes : sender = `"monDomaine.com:résultat"`, `replyMethod` = `"unRésultat"`, `n1` = 123 et `n2` = 456. Elle exécute donc la ligne de code suivante :

```
this.send("monDomaine.com:résultat", "unRésultat", (123 + 456));
```

- La méthode `unRésultat` (ligne 54) affiche la valeur renvoyée par `uneSomme` (579).

```
// Le fichier SWF de réception sur http://www.monDomaine.com/répertoire/
  animation.swf
// contient le code suivant

1    var uneConnexionLocale = new LocalConnection();
2    uneConnexionLocale.allowDomain = function()
3    {
     // Autoriser les connexions de tous les domaines
4      return true;
5    }
6    uneConnexionLocale.uneSomme = function(sender, replyMethod, n1, n2)
7    {
8      this.send(sender, replyMethod, (n1 + n2));
9    }
10
11   uneConnexionLocale.connect("somme");

// Le fichier SWF d'envoi sur http://www.votreDomaine.com/répertoire/
  animation.swf
// contient le code suivant

50   var cl = new LocalConnection();
51   cl.allowDomain = function(unDomaine) {
       // N'autoriser que les connexions à partir de monDomaine.com
52     return (unDomaine == "monDomaine.com");
53   }
54   cl.unRésultat = function(unParam) {
55     trace("La somme est" + unParam);
56   }
57
58   cl.connect("résultat");
59   cl.send("monDomaine.com:somme", "uneSomme", cl.domain() + ':' +
  "résultat",
         "unRésultat", 123, 456);
```

Consultez également

`LocalConnection.allowDomain`

LocalConnection.onStatus

Disponibilité

Flash Player 6.

Usage

```
Envoi_lc.onStatus = function(objetInfo) {
  // vos instructions
}
```

Paramètres

objetInfo Un paramètre défini selon le message d'état. Pour plus de détails concernant ce paramètre, consultez la description ci-dessous.

Renvoie

Rien.

Description

Gestionnaire d'événement : invoqué après qu'un objet d'envoi LocalConnection a essayé d'envoyer une commande à un objet de réception LocalConnection. Si vous souhaitez répondre à ce gestionnaire d'événement, vous devez créer une fonction afin de traiter l'objet LocalConnection.

Si l'objet d'information renvoyé par ce gestionnaire d'événement contient une valeur `level "Etat"`, Flash parvient à envoyer la commande à un objet de réception LocalConnection. Cela ne signifie pas que Flash parvient à invoquer la méthode spécifiée de l'objet de réception LocalConnection, mais simplement qu'il a envoyé la commande. Par exemple, la méthode n'est pas invoquée si l'objet de réception LocalConnection ne permet 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 bien été invoquée est de faire en sorte que l'objet de réception envoie une réponse à l'objet d'envoi.

Si l'objet d'information renvoyé par ce gestionnaire contient une valeur `level "Error"`, cela signifie que Flash n'a pas pu envoyer la commande à un objet de réception LocalConnection, sans doute parce qu'il n'existe pas d'objet de réception LocalConnection connecté dont le nom spécifié corresponde au nom spécifié dans la commande *Envoi_lc*`.send()` qui a invoqué ce gestionnaire.

Outre 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 lui répondre, Flash traite une fonction affectée à `System.onStatus` s'il en existe une.

Dans la plupart des cas, vous n'aurez à implémenter ce gestionnaire que pour répondre aux conditions d'erreur, comme indiqué dans l'exemple suivant.

Exemple

L'exemple suivant affiche les informations concernant une connexion ayant échoué dans le panneau de sortie :

```
Envoi_lc = new LocalConnection();
Envoi_lc.onStatus = function(objetInfo)
{
  if (objetInfo.level == "Erreur")
  {
    trace("La connexion a échoué.");
  }
}
Envoi_lc.send("Réception_lc", "nomDeMéthode");
```

Consultez également

LocalConnection.send(), System.onStatus

LocalConnection.send()

Disponibilité

Flash Player 6.

Usage

Envoi_lc.send (*nomDeConnexion*, *méthode* [, *p1,...,pN*])

Paramètres

nomDeConnexion Une chaîne qui correspond au nom de connexion spécifié dans la commande LocalConnection.connect() qui veut communiquer avec *Envoi_lc*.

méthode Une chaîne spécifiant le nom d'une méthode à invoquer dans l'objet de réception LocalConnection. Les noms de méthode suivants provoquent l'échec de la commande : send, connect, close, domain, onStatus et allowDomain.

p1,...pN paramètres facultatifs devant être transmis à la méthode spécifiée.

Renvoie

Une valeur booléenne true si Flash peut effectuer la requête : sinon, la valeur est false.

Remarque : Une valeur renvoyée true ne signifie pas nécessairement que Flash a réussi à se connecter à un objet de réception LocalConnection, mais simplement que la syntaxe de la commande est correcte. Pour déterminer si la connexion a réussi, consultez LocalConnection.onStatus.

Description

Méthode : invoque la méthode nommée *méthode* sur une connexion ouverte avec la commande LocalConnection.connect(*nomDeConnexion*) (appelée « objet de réception LocalConnection »). L'objet utilisé avec cette commande est nommé « objet d'envoi LocalConnection ». Les fichiers SWF contenant les objets d'envoi et de réception doivent être exécutés sur le même ordinateur client.

Il existe une limite en terme de quantité de données transmissibles en tant que paramètres à cette commande. Si la commande renvoie false mais que votre syntaxe est correcte, essayez de séparer les requêtes LocalConnection.send() en plusieurs commandes.

Comme discuté dans l'entrée LocalConnection.connect(), Flash ajoute le super-domaine actuel à *nomDeConnexion* par défaut. Si vous implémentez une communication entre différents domaines, vous devez définir *nomDeConnexion* dans les objets d'envoi et de réception LocalConnection, de telle sorte que Flash n'ajoute pas le super-domaine actuel à *nomDeConnexion*. Il existe deux façons de faire :

- Utilisez un trait de soulignement (_) au début de la chaîne *nomDeConnexion* des objets d'envoi et de réception LocalConnection. Dans le fichier SWF contenant l'objet de réception, utilisez LocalConnection.allowDomain pour indiquer que les connexions à partir de tous les domaines sont acceptées. Cette implémentation vous permet de stocker vos fichiers SWF d'envoi et de réception dans tous les domaines.
- Incluez le super-domaine dans le *nomDeConnexion* de l'objet d'envoi LocalConnection, par exemple monDomaine.com:monNomDeConnexion. Dans l'objet de réception, utilisez LocalConnection.allowDomain pour indiquer que les connexions à partir du super-domaine spécifié seront acceptées (dans ce cas, monDomaine.com) ou que les connexions à partir de tous les domaines seront acceptées.

Remarque : Il n'est pas possible de spécifier un super-domaine dans la chaîne *nom de connexion* de l'objet de réception LocalConnection. Vous ne pouvez le faire que dans l'objet d'envoi LocalConnection.

Exemple

Pour un exemple de communication entre des objets LocalConnection situés dans le même domaine, consultez `LocalConnection.connect()`. Pour un exemple de communication entre des objets LocalConnection situés dans tous les domaines, consultez `LocalConnection.allowDomain`. Pour un exemple de communication entre des objets LocalConnection situés dans des domaines spécifiés, consultez `LocalConnection.allowDomain` et `LocalConnection.domain()`.

Consultez également

`LocalConnection.allowDomain`, `LocalConnection.connect()`, `LocalConnection.domain()`, `LocalConnection.onStatus`

lt (inférieur à – spécifique aux chaînes)

Disponibilité

Flash Player 4. Cet opérateur est déconseillé dans Flash 5 et remplacé par le nouvel opérateur <(inférieur à).

Usage

expression1 `lt` *expression2*

Paramètres

expression1, *expression2* Nombres, chaînes ou variables.

Description

Opérateur (comparaison) : compare *expression1* avec *expression2* et renvoie `true` si *expression1* est inférieure à *expression2* ; sinon, renvoie `false`.

Consultez également

`<(inférieur à)`

Classe Math

Disponibilité

Flash Player 5. Dans Flash Player 4, les méthodes et les propriétés de la classe Math sont émulées à l'aide d'approximations et peuvent ne pas être aussi précises que les fonctions mathématiques non émulées prises en charge par Flash Player 5.

Description

La classe Math est une classe de premier niveau dont les méthodes et les propriétés sont utilisables sans constructeur.

Utilisez les méthodes et les propriétés de cette classe pour accéder aux constantes et aux fonctions mathématiques et pour les manipuler. Toutes les propriétés et méthodes de la classe Math sont statiques et doivent être appelées en utilisant la syntaxe `Math.méthode(paramètre)` ou `Math.constante`. Dans ActionScript, les constantes sont définies avec le maximum de précision des nombres à virgule flottante double précision IEEE-754.

Plusieurs méthodes de la classe Math utilisent le radian d'un angle comme paramètre. Vous pouvez utiliser l'équation ci-dessous pour calculer les valeurs en radians, ou simplement transmettre l'équation (en entrant une valeur pour les degrés) pour le paramètre de radian.

Pour calculer une valeur en radians, utilisez cette formule :

```
radian = Math.PI/180 * degree
```

L'exemple suivant illustre la transmission de l'équation comme paramètre pour calculer le sinus d'un angle de 45 degrés :

`Math.SIN(Math.PI/180 * 45)` est identique à `Math.SIN(.7854)`

La classe Math est totalement prise en charge par Flash Player 5. Dans Flash Player 4, les méthodes de la classe Math fonctionnent, mais sont émulées à l'aide d'approximations et peuvent ne pas être aussi précises que les fonctions mathématiques non émulées prises en charge par Flash Player 5.

Méthodes de la classe Math

Méthode	Description
`Math.abs()`	Calcule une valeur absolue.
`Math.acos`	Calcule le cosinus d'un arc.
`Math.asin()`	Calcule le sinus d'un arc.
`Math.atan()`	Calcule la tangente d'un arc.
`Math.atan2()`	Calcule un angle depuis l'axe des x jusqu'au point.
`Math.ceil()`	Arrondit un nombre à l'entier supérieur le plus proche.
`Math.cos`	Calcule un cosinus.
`Math.exp`	Calcule une valeur exponentielle.
`Math.floor`	Arrondit un nombre à l'entier inférieur le plus proche.
`Math.log()`	Calcule un logarithme naturel.

Méthode	Description
`Math.max()`	Renvoie le plus grand des deux entiers.
`Math.min()`	Renvoie le plus petit des deux entiers.
`Math.pow()`	Calcule *x* élevé à la puissance *y*.
`Math.random()`	Renvoie un nombre pseudo-aléatoire entre 0,0 et 1,0.
`Math.round()`	Arrondit à l'entier le plus proche.
`Math.sin()`	Calcule un sinus.
`Math.sqrt()`	Calcule une racine carrée.
`Math.tan()`	Calcule une tangente.

Propriétés de la classe Math

Toutes les propriétés de la classe Math sont des constantes.

Propriété	Description
`Math.E`	Constante d'Euler et base des logarithmes naturels (approximativement 2,718).
`Math.LN2`	Le logarithme naturel de 2 (approximativement 0,693).
`Math.LOG2E`	Le logarithme de base 2 de *e* (approximativement 1,442).
`Math.LN2`	Le logarithme naturel de 10 (approximativement 2,302).
`Math.LOG10E`	Le logarithme de base 10 de *e* (approximativement 0,434).
`Math.PI`	Le rapport de la circonférence d'un cercle à son diamètre (approximativement 3,14159).
`Math.SQRT1_2`	La réciproque de la racine carrée de 1/2 (approximativement 0,707).
`Math.SQRT2`	La racine carrée de 2 (approximativement 1,414).

Math.abs()

Disponibilité

Flash Player 5. Dans Flash Player 4, les méthodes et les propriétés de la classe Math sont émulées à l'aide d'approximations et peuvent ne pas être aussi précises que les fonctions mathématiques non émulées prises en charge par Flash Player 5.

Usage

```
Math.abs(x)
```

Paramètres

x Un nombre.

Renvoie

Nombre.

Description

Méthode : calcule et renvoie une valeur absolue pour le nombre spécifié par le paramètre *x*.

Math.acos

Disponibilité

Flash Player 5. Dans Flash Player 4, les méthodes et les propriétés de la classe Math sont émulées à l'aide d'approximations et peuvent ne pas être aussi précises que les fonctions mathématiques non émulées prises en charge par Flash Player 5.

Usage

```
Math.acos(x)
```

Paramètres

x Un nombre entre -1,0 et 1,0.

Renvoie

Rien.

Description

Méthode : calcule et renvoie le cosinus de l'arc du nombre spécifié par le paramètre *x*, en radians.

Math.asin()

Disponibilité

Flash Player 5. Dans Flash Player 4, les méthodes et les propriétés de la classe Math sont émulées à l'aide d'approximations et peuvent ne pas être aussi précises que les fonctions mathématiques non émulées prises en charge par Flash Player 5.

Usage

```
Math.asin(x);
```

Paramètres

x Un nombre entre -1,0 et 1,0.

Renvoie

Nombre.

Description

Méthode : calcule et renvoie le sinus de l'arc du nombre spécifié par le paramètre *x*, en radians.

Math.atan()

Disponibilité

Flash Player 5. Dans Flash Player 4, les méthodes et les propriétés de la classe Math sont émulées à l'aide d'approximations et peuvent ne pas être aussi précises que les fonctions mathématiques non émulées prises en charge par Flash Player 5.

Usage

```
Math.atan(x)
```

Paramètres

x Un nombre représentant la tangente d'un angle.

Renvoie

Nombre.

Description

Méthode : calcule et renvoie la valeur, en radians, de l'angle dont la tangente est spécifiée dans le paramètre *x*. La valeur renvoyée se situe entre pi négatif divisé par 2 et pi positif divisé par 2.

Consultez également

`Math.atan2()`, `Math.tan()`

Math.atan2()

Disponibilité

Flash Player 5. Dans Flash Player 4, les méthodes et les propriétés de la classe Math sont émulées à l'aide d'approximations et peuvent ne pas être aussi précises que les fonctions mathématiques non émulées prises en charge par Flash Player 5.

Usage

```
Math.atan2(y, x)
```

Paramètres

y Un nombre spécifiant la coordonnée *y* du point.

x Un nombre spécifiant la coordonnée *x* du point.

Renvoie

Nombre.

Description

Méthode : calcule et renvoie l'angle du point *y*/*x* en radians, mesuré à partir de l'axe des *x* d'un cercle dans le sens inverse des aiguilles d'une montre (0,0 représentant le centre du cercle). La valeur renvoyée se situe entre pi positif et pi négatif.

Consultez également

`Math.atan()`, `Math.tan()`

Math.ceil()

Disponibilité

Flash Player 5. Dans Flash Player 4, les méthodes et les propriétés de la classe Math sont émulées à l'aide d'approximations et peuvent ne pas être aussi précises que les fonctions mathématiques non émulées prises en charge par Flash Player 5.

Usage

```
Math.ceil(x)
```

Paramètres

x Un nombre ou expression.

Renvoie

Nombre.

Description

Méthode : renvoie le plafond du nombre ou de l'expression spécifiés. Le plafond d'un nombre est l'entier le plus proche supérieur ou égal à ce nombre.

Math.cos

Disponibilité

Flash Player 5. Dans Flash Player 4, les méthodes et les propriétés de la classe Math sont émulées à l'aide d'approximations et peuvent ne pas être aussi précises que les fonctions mathématiques non émulées prises en charge par Flash Player 5.

Usage

```
Math.cos(x)
```

Paramètres

x Un angle mesuré en radians.

Renvoie

Nombre.

Description

Méthode : renvoie le cosinus (une valeur entre -1.0 et 1.0) de l'angle spécifié par le paramètre *x*. L'angle *x* doit être spécifié en radians. Utilisez les informations surlignées dans l'entrée Classe Math afin de calculer un radian.

Math.E

Disponibilité

Flash Player 5. Dans Flash Player 4, les méthodes et les propriétés de la classe Math sont émulées à l'aide d'approximations et peuvent ne pas être aussi précises que les fonctions mathématiques non émulées prises en charge par Flash Player 5.

Usage

```
Math.E
```

Paramètres

Aucun.

Renvoie

Rien.

Description

Constante : une constante mathématique pour la base des logarithmes naturels, exprimée sous la forme *e*. La valeur approximative de *e* est 2.71828.

Math.exp

Disponibilité

Flash Player 5. Dans Flash Player 4, les méthodes et les propriétés de la classe Math sont émulées à l'aide d'approximations et peuvent ne pas être aussi précises que les fonctions mathématiques non émulées prises en charge par Flash Player 5.

Usage

```
Math.exp(x)
```

Paramètres

x L'exposant : un nombre ou une expression.

Renvoie

Nombre.

Description

Méthode : renvoie la valeur de la base du logarithme naturel (*e*), à la puissance de l'exposant spécifié dans le paramètre *x*. La constante `Math.E` peut fournir la valeur de *e*.

Math.floor

Disponibilité

Flash Player 5. Dans Flash Player 4, les méthodes et les propriétés de la classe Math sont émulées à l'aide d'approximations et peuvent ne pas être aussi précises que les fonctions mathématiques non émulées prises en charge par Flash Player 5.

Usage

```
Math.floor(x)
```

Paramètres

x Un nombre ou expression.

Renvoie

Nombre.

Description

Méthode : renvoie le plancher du nombre ou de l'expression spécifiés dans le paramètre *x*. Le plancher est l'entier le plus proche inférieur ou égal au nombre ou à l'expression spécifiés.

Exemple

Le code suivant renvoie une valeur de 12 :

```
Math.floor(12.5);
```

Math.log()

Disponibilité

Flash Player 5. Dans Flash Player 4, les méthodes et les propriétés de la classe Math sont émulées à l'aide d'approximations et peuvent ne pas être aussi précises que les fonctions mathématiques non émulées prises en charge par Flash Player 5.

Usage

```
Math.log(x)
```

Paramètres

x Un nombre ou une expression avec une valeur supérieure à 0.

Renvoie

Nombre.

Description

Méthode : renvoie le logarithme du paramètre *x*.

Math.LN2

Disponibilité

Flash Player 5. Dans Flash Player 4, les méthodes et les propriétés de la classe Math sont émulées à l'aide d'approximations et peuvent ne pas être aussi précises que les fonctions mathématiques non émulées prises en charge par Flash Player 5.

Usage

```
Math.LN2
```

Paramètres

Aucun.

Renvoie

Rien.

Description

Constante : une constante mathématique pour le logarithme naturel de 2, exprimée sous la forme $\log_e 2$, avec une valeur approximative de 0.6931471805599452862.

Math.LN10

Disponibilité

Flash Player 5. Dans Flash Player 4, les méthodes et les propriétés de la classe Math sont émulées à l'aide d'approximations et peuvent ne pas être aussi précises que les fonctions mathématiques non émulées prises en charge par Flash Player 5.

Usage

```
Math.LN10
```

Paramètres

Aucun.

Renvoie

Rien.

Description

Constante : une constante mathématique pour le logarithme naturel de 10, exprimée sous la forme $\log_e 10$, avec une valeur approximative de 2.3025850929940459011.

Math.LOG2E

Disponibilité

Flash Player 5. Dans Flash Player 4, les méthodes et les propriétés de la classe Math sont émulées à l'aide d'approximations et peuvent ne pas être aussi précises que les fonctions mathématiques non émulées prises en charge par Flash Player 5.

Usage

```
Math.LOG2E
```

Paramètres

Aucun.

Renvoie

Rien.

Description

Constante : une constante mathématique pour le logarithme de base 2 de la constante *e* (`Math.E`), exprimée sous la forme $\log 2_e$, avec une valeur approximative de 1.442695040888963387.

Math.LOG10E

Disponibilité

Flash Player 5. Dans Flash Player 4, les méthodes et les propriétés de la classe Math sont émulées à l'aide d'approximations et peuvent ne pas être aussi précises que les fonctions mathématiques non émulées prises en charge par Flash Player 5.

Usage

```
Math.LOG10E
```

Paramètres

Aucun.

Renvoie

Rien.

Description

Constante : une constante mathématique pour le logarithme de base 10 de la constante *e* (`Math.E`), exprimée sous la forme $\log_{10}e$, avec une valeur approximative de 0.4342944819032518167.

Math.max()

Disponibilité

Flash Player 5. Dans Flash Player 4, les méthodes et les propriétés de la classe Math sont émulées à l'aide d'approximations et peuvent ne pas être aussi précises que les fonctions mathématiques non émulées prises en charge par Flash Player 5.

Usage

```
Math.max(x , y)
```

Paramètres

x Un nombre ou expression.

y Un nombre ou une expression.

Renvoie

Nombre.

Description

Méthode : évalue *x* et *y* et renvoie la valeur la plus grande.

Math.min()

Disponibilité

Flash Player 5. Dans Flash Player 4, les méthodes et les propriétés de la classe Math sont émulées à l'aide d'approximations et peuvent ne pas être aussi précises que les fonctions mathématiques non émulées prises en charge par Flash Player 5.

Usage

```
Math.min(x , y)
```

Paramètres

x Un nombre ou expression.

y Un nombre ou une expression.

Renvoie

Nombre.

Description

Méthode : évalue *x* et *y* et renvoie la valeur la plus petite.

Math.PI

Disponibilité

Flash Player 5. Dans Flash Player 4, les méthodes et les propriétés de la classe Math sont émulées à l'aide d'approximations et peuvent ne pas être aussi précises que les fonctions mathématiques non émulées prises en charge par Flash Player 5.

Usage

```
Math.PI
```

Paramètres

Aucun.

Renvoie

Rien.

Description

Constante : une constante mathématique pour le rapport de la circonférence d'un cercle à son diamètre, exprimée sous la forme pi, avec une valeur de 3.14159265358979.

Math.pow()

Disponibilité

Flash Player 5. Dans Flash Player 4, les méthodes et les propriétés de la classe Math sont émulées à l'aide d'approximations et peuvent ne pas être aussi précises que les fonctions mathématiques non émulées prises en charge par Flash Player 5.

Usage

```
Math.pow(x , y)
```

Paramètres

x Un nombre à élever à une puissance.

y Un nombre spécifiant la puissance à laquelle le paramètre *x* est élevé.

Renvoie

Nombre.

Description

Méthode : calcule et renvoie *x* à la puissance de *y*: *xy*.

Math.random()

Disponibilité

Flash Player 5. Dans Flash Player 4, les méthodes et les propriétés de la classe Math sont émulées à l'aide d'approximations et peuvent ne pas être aussi précises que les fonctions mathématiques non émulées prises en charge par Flash Player 5.

Usage

```
Math.random()
```

Paramètres

Aucun.

Renvoie

Nombre.

Description

Méthode : renvoie n, où 0 <= n < 1.

Consultez également

random

Math.round()

Disponibilité

Flash Player 5. Dans Flash Player 4, les méthodes et les propriétés de la classe Math sont émulées à l'aide d'approximations et peuvent ne pas être aussi précises que les fonctions mathématiques non émulées prises en charge par Flash Player 5.

Usage

```
Math.round(x)
```

Paramètres

x Un nombre.

Renvoie

Nombre.

Description

Méthode : arrondit la valeur du paramètre *x* à l'entier inférieur ou supérieur le plus proche et renvoie la valeur.

Math.sin()

Disponibilité

Flash Player 5. Dans Flash Player 4, les méthodes et les propriétés de la classe Math sont émulées à l'aide d'approximations et peuvent ne pas être aussi précises que les fonctions mathématiques non émulées prises en charge par Flash Player 5.

Usage

```
Math.sin(x)
```

Paramètres

x Un angle mesuré en radians.

Renvoie

Nombre : le sinus de l'angle spécifié (entre -1,0 et 1,0).

Description

Méthode : calcule et renvoie le sinus de l'angle spécifié, en radians. Utilisez les informations surlignées dans l'entrée Classe Math afin de calculer un radian.

Math.sqrt()

Disponibilité

Flash Player 5. Dans Flash Player 4, les méthodes et les propriétés de la classe Math sont émulées à l'aide d'approximations et peuvent ne pas être aussi précises que les fonctions mathématiques non émulées prises en charge par Flash Player 5.

Usage

```
Math.sqrt(x)
```

Paramètres

x Un nombre ou une expression supérieur ou égal à 0.

Renvoie

Nombre.

Description

Méthode : calcule et renvoie la racine carrée du nombre spécifié.

Math.SQRT1_2

Disponibilité

Flash Player 5. Dans Flash Player 4, les méthodes et les propriétés de la classe Math sont émulées à l'aide d'approximations et peuvent ne pas être aussi précises que les fonctions mathématiques non émulées prises en charge par Flash Player 5.

Usage

```
Math.SQRT1_2
```

Paramètres

Aucun.

Renvoie

Rien.

Description

Constante : une constante mathématique pour la racine carrée d'un demi.

Math.SQRT2

Disponibilité

Flash Player 5. Dans Flash Player 4, les méthodes et les propriétés de la classe Math sont émulées à l'aide d'approximations et peuvent ne pas être aussi précises que les fonctions mathématiques non émulées prises en charge par Flash Player 5.

Usage

```
Math.SQRT2
```

Paramètres

Aucun.

Renvoie

Rien.

Description

Constante : une constante mathématique pour la racine carrée de 2, avec une valeur approximative de 1.414213562373.

Math.tan()

Disponibilité

Flash Player 5. Dans Flash Player 4, les méthodes et les propriétés de la classe Math sont émulées à l'aide d'approximations et peuvent ne pas être aussi précises que les fonctions mathématiques non émulées prises en charge par Flash Player 5.

Usage

```
Math.tan(x)
```

Paramètres

x Un angle mesuré en radians.

Renvoie

Nombre.

Description

Méthode : calcule et renvoie la tangente de l'angle spécifié. Pour calculer un radian, utilisez les informations indiquées dans l'introduction de Classe Math.

Consultez également

```
Math.atan()
```

maxscroll

Disponibilité

Flash Player 4. Cette fonction est déconseillée et remplacée par la propriété `TextField.maxscroll`.

Usage

```
nomDeVariable.maxscroll
```

Description

Propriété (lecture seule) : une propriété déconseillée qui indique le numéro de ligne de la première ligne de texte visible dans un champ de texte lorsque la dernière ligne du champ est également visible. La propriété `maxscroll` fonctionne avec la propriété `scroll` pour contrôler l'affichage des informations dans un champ de texte. Cette propriété peut être récupérée mais pas modifiée.

Consultez également

`TextField.maxscroll`, `TextField.scroll`

mbchr

Disponibilité

Flash Player 4. Cette fonction est déconseillée et remplacée par la méthode `String.fromCharCode()`.

Usage

```
mbchr(nombre)
```

Paramètres

nombre Le nombre à convertir en caractère multioctet.

Renvoie

Une chaîne.

Description

Fonction de chaîne : convertit un code ASCII en caractère multioctet.

Consultez également

`String.fromCharCode()`

mblength

Disponibilité

Flash Player 4. Cette fonction est déconseillée et remplacée par Classe String.

Usage

```
mblength(chaîne)
```

Paramètres

chaîne Une chaîne.

Renvoie

Nombre.

Description

Fonction de chaîne : renvoie la longueur de la chaîne de caractères multioctet.

mbord

Disponibilité

Flash Player 4. Cette fonction est déconseillée dans Flash 5 ; utilisez plutôt `String.charCodeAt`.

Usage

```
mbord(caractère)
```

Paramètres

caractère Le caractère à convertir en nombre multioctet.

Renvoie

Nombre.

Description

Fonction de chaîne : convertit le caractère spécifié en nombre multioctet.

Consultez également

`String.fromCharCode()`

mbsubstring

Disponibilité

Flash Player 4. Cette fonction est déconseillée dans Flash 5 ; utilisez plutôt `String.substr`.

Usage

```
mbsubstring(valeur, index, nombre)
```

Paramètres

valeur La chaîne multioctet de laquelle extraire une nouvelle chaîne multioctet.

index Le numéro du premier caractère à extraire.

nombre Le nombre de caractères à inclure dans la chaîne extraite, en excluant le caractère d'index.

Renvoie

Une chaîne.

Description

Fonction de chaîne : extrait une nouvelle chaîne de caractères multioctet d'une chaîne de caractères multioctet.

Consultez également

`String.substr`

Classe Microphone

Disponibilité

Flash Player 6.

Description

La classe Microphone permet de capturer du son à partir d'un microphone connecté à l'ordinateur sur lequel Flash Player est exécuté.

La classe Microphone est essentiellement destinée à être utilisée avec Flash Communication Server, mais vous pouvez l'utiliser de façon restreinte sans le serveur, par exemple, pour transmettre du son à partir du microphone vers les haut-parleurs de votre système local.

Pour créer ou référencer un objet Microphone, utilisez la méthode `Microphone.get()`.

Méthodes de la classe Microphone

Méthode	Description
`Microphone.get()`	Renvoie un objet Microphone par défaut ou spécifié, ou `null` si le microphone n'est pas disponible.
`Microphone.setGain()`	Spécifie le niveau d'amplification que le microphone doit appliquer au signal.
`Microphone.setRate()`	Spécifie le taux d'échantillonnage auquel le microphone doit capturer le son, en kHz.
`Microphone.setSilenceLevel()`	Spécifie le niveau sonore requis pour activer le microphone.
`Microphone.setUseEchoSuppression()`	Spécifie l'utilisation ou non de la fonction de suppression de l'écho du codec audio.

Propriétés de la classe Microphone

Propriété (lecture seule)	Description
`Microphone.activityLevel`	Quantité de son détectée par le microphone.
`Microphone.gain`	Niveau d'amplification que le microphone applique au signal avant de le transmettre.
`Microphone.index`	Index du microphone actuel.
`Microphone.muted`	Valeur booléenne indiquant si l'utilisateur dispose de l'accès autorisé ou refusé au microphone.
`Microphone.name`	Nom du périphérique de capture du son, tel que renvoyé par le périphérique de capture du son.
`Microphone.names`	Propriété de classe : tableau de chaînes contenant les noms de l'ensemble des périphériques de capture du son disponibles, y compris les cartes son et les microphones.
`Microphone.rate`	Taux d'échantillonnage du son, en kHz.
`Microphone.silenceLevel()`	Niveau sonore requis pour activer le microphone.
`Microphone.silenceTimeout()`	Nombre de millisecondes entre le moment où le microphone cesse de détecter du son et le moment où `Microphone.onActivity(false)` est appelé.
`Microphone.useEchoSuppression()`	Valeur booléenne spécifiant si la fonction de suppression de l'écho est utilisée.

Gestionnaires d'événement de la classe Microphone

Gestionnaire d'événement	Description
`Microphone.onActivity`	Invoqué lorsque le microphone commence ou arrête de détecter du son.
`Microphone.onStatus`	Invoqué lorsque l'utilisateur autorise ou refuse l'accès au microphone.

Constructeur de la classe Microphone

Pour plus d'informations, consultez `Microphone.get()`.

Microphone.activityLevel

Disponibilité

Flash Player 6.

Usage

`MicrophoneActif``.activityLevel`

Description

Propriété (lecture seule) : valeur numérique spécifiant le niveau sonore détecté par le microphone. Cette valeur est comprise entre 0 (aucun son n'est détecté) et 100 (les sons très puissants sont détectés). La valeur de cette propriété peut vous aider à déterminer la valeur à transmettre à la méthode `Microphone.setSilenceLevel()`.

Si le microphone est disponible mais qu'il n'est pas encore utilisé car `Microphone.get()` n'a pas encore été appelé, cette propriété est définie sur -1.

Exemple

L'exemple suivant définit la variable `niveau` sur le niveau d'activité du microphone actuel, `monMic.activityLevel`.

```
var niveau = monMic.activityLevel;
```

Consultez également

`Microphone.setGain()`

Microphone.gain

Disponibilité

Flash Player 6.

Usage

MicrophoneActif.gain

Description

Propriété (lecture seule) : le niveau d'amplification que le microphone applique au signal. Les valeurs valides sont comprises entre 0 et 100. La valeur par défaut est 50.

Exemple

L'exemple suivant est associé au curseur d'une barre de réglage. Lorsque le clip est chargé, Flash recherche la valeur de `monMic.gain` et utilise une valeur par défaut si elle n'est pas définie. La position `_x` sert ensuite à définir le gain du microphone selon les préférences de l'utilisateur.

```
onClipEvent(load) {
  if (_root.monMic.gain == undefined) {
    _root.monMic.setGain = 75;
  }

  this._x = _root.monMic.gain;
  _root.txt_micgain = this._x;

  left = this._x;
  right = left+50;
  top = this._y;
  bottom = top;
}

on(press) {
  startDrag(this, false, gauche, haut, droit, bas);
  this._xscale = 100;
  this._yscale = 100;
}

on (release, releaseOutside) {
  stopDrag();
  g = (this._x-50)*2;
  _root.monMic.setGain(g);
  _root.txt_micgain = g;
  this._xscale = 100;
  this._yscale = 100;
}
```

Consultez également

`Microphone.setGain()`

Microphone.get()

Disponibilité

Flash Player 6.

Usage

```
Microphone.get([index])
```

Remarque : La syntaxe correcte est `Microphone.get()`. Pour affecter l'objet Microphone à une variable, utilisez une syntaxe de type *`active_mic`*`=Microphone.get()`.

Paramètres

`index` Un entier de base rézo facultatif qui spécifie le microphone à récupérer, comme l'indique le tableau contenu par `Microphone.names`. Pour récupérer le microphone par défaut (ce qui est recommandé pour la plupart des applications), omettez ce paramètre.

Renvoie

- Si *`index`* n'est pas spécifié, la méthode renvoie une référence au microphone par défaut ou, s'il n'est pas disponible, au premier microphone disponible. Si aucun microphone n'est disponible ou installé, la méthode renvoie `null`.
- Si *`index`* n'est pas spécifié, la méthode renvoie une référence au microphone demandé ou `null` s'il n'est pas disponible.

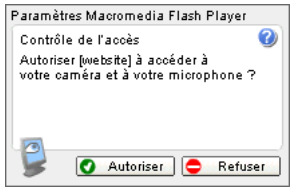
Description

Méthode : renvoie une référence à l'objet Microphone afin de capturer du son. Pour réellement commencer à capturer du son, vous devez associer l'objet Microphone à un objet MovieClip (consultez `MovieClip.attachAudio()`).

Contrairement aux objets créés à l'aide du constructeur `new`, les appels multiples à `Microphone.get()` font toujours référence au même microphone. Si le script contient les lignes `mic1=Microphone.get()` et `mic2=Microphone.get()`, `mic1` et `mic2` font toujours référence au même microphone (par défaut).

En général, il n'est pas nécessaire de transmettre la valeur à *`index`* ; il suffit d'utiliser la méthode `Microphone.get()` pour renvoyer une référence au microphone par défaut. L'utilisateur peut indiquer le microphone à utiliser par défaut dans le panneau Microphone des paramètres Flash Player (décrit plus loin dans cette section). Si vous transmettez une valeur à *`index`*, vous risquez de faire référence à un microphone différent de celui que l'utilisateur préfère. La valeur *`index`* peut servir dans quelques cas assez rares, par exemple, si votre application capture du son à partir de deux microphones à la fois.

Lorsqu'un fichier SWF tente d'accéder au microphone renvoyé par la méthode `Microphone.get()` (par exemple, lorsque vous utilisez `MovieClip.attachAudio()`), Flash Player affiche le panneau de contrôle de l'accès permettant à l'utilisateur d'autoriser ou de refuser l'accès au microphone. (Vérifiez que la scène mesure au moins 215 x 138 pixels ; c'est la taille minimale requise par Flash pour afficher la boîte de dialogue.)



Lorsque l'utilisateur répond à cette boîte de dialogue, le gestionnaire d'événement `Microphone.onStatus` renvoie un objet information indiquant la réponse. Pour savoir si l'utilisateur a autorisé ou refusé l'accès à la camera sans traiter le gestionnaire d'événement, utilisez `Microphone.muted`.

Pour spécifier des paramètres de contrôle de l'accès permanents pour un domaine particulier, l'utilisateur peut également cliquer avec le bouton droit de la souris (Windows) ou en appuyant sur la touche Contrôle (Macintosh) pendant la lecture d'un fichier SWF, puis choisir Paramètres, ouvrir le panneau de contrôle de l'accès et sélectionner Mémoriser.

Si `Microphone.get()` renvoie `null`, le microphone est utilisé par une autre application ou aucun microphone n'est installé sur le système. Pour savoir si des microphones sont installés, utilisez `Microphones.names.length`. Pour afficher le panneau Microphone des paramètres Flash Player, qui permet à l'utilisateur de choisir le microphone référencé par `Microphone.get()`, utilisez `System.showSettings(2)`.

Exemple

L'exemple suivant permet à l'utilisateur de spécifier le microphone par défaut, de capturer du son et de le restitue localement. Pour éviter le retour, vous pouvez tester ce code en portant un casque.

```
System.showSettings(2);
monMic = Microphone.get();
_root.attachAudio(monMic);
```

Consultez également

`Microphone.index`, `Microphone.muted`, `Microphone.names`, `Microphone.onStatus`, `MovieClip.attachAudio()`

Microphone.index

Disponibilité

Flash Player 6.

Usage

`MicrophoneActif``.index`

Description

Propriété (lecture seule) : un entier basé sur zéro spécifiant l'index du microphone, tel que dans le tableau renvoyé par `Microphone.names`.

Consultez également

`Microphone.get()`, `Microphone.names`

Microphone.muted

Disponibilité

Flash Player 6.

Usage

MicrophoneActif.muted

Description

Propriété (lecture seule) : valeur booléenne spécifiant si l'utilisateur a refusé l'accès au microphone (`true`) ou l'a autorisé (`false`). Lorsque cette valeur change, `Microphone.onStatus` est invoqué. Pour plus d'informations, consultez `Microphone.get()`.

Exemple

Dans l'exemple suivant, lorsque l'utilisateur clique sur le bouton, Flash publie et lit un flux continu si le microphone n'est pas désactivé.

```
on (press)
{
  // Si l'utilisateur désactive le microphone, afficher une notification.
  // Sinon, publier et lire un flux continu à partir du microphone.
  if(monMic.muted) {
    _root.debugWindow+="Microphone désactivé." + newline;
  } else {

    // Publier les données du microphone en appelant
    // la fonction racine pubLive().
    _root.pubLive();

    // Jouer ce qui est publié en appelant
    // la fonction racine playLive().
    _root.playLive();
  }
}
```

Consultez également

`Microphone.get()`, `Microphone.onStatus`

Microphone.name

Disponibilité

Flash Player 6.

Usage

MicrophoneActif.name

Description

Propriété (lecture seule) : une chaîne spécifiant le nom du périphérique de son actuel, tel que renvoyé par le périphérique de capture du son.

Exemple

L'exemple suivant affiche le nom du microphone par défaut dans le panneau de sortie.

```
monMic = Microphone.get();
trace("Le nom du microphone est : " + monMic.name);
```

Consultez également

Microphone.get(), Microphone.names

Microphone.names

Disponibilité

Flash Player 6.

Usage

```
Microphone.names
```

Remarque : La syntaxe correcte est `Microphone.names`. Pour affecter la valeur de renvoi à une variable, utilisez une syntaxe de type *`mic_array`*`=Microphone.names`. Pour connaître le nom du microphone actuel, utilisez *`MicrophoneActif`*`.name`.

Description

Propriété de classe (lecture seule) : récupère un tableau de chaînes contenant les noms de tous les périphériques de capture de son disponibles sans afficher le panneau des paramètres de contrôle de l'accès de Flash Player. Ce tableau se comporte comme tous les tableaux ActionScript : il fournit implicitement l'index basé sur zéro de chaque périphérique de capture du son et indique le nombre total de périphériques de capture du son installés sur le système (à l'aide de `Microphone.names.length`). Pour plus d'informations, consultez l'entrée Classe Array.

La fonction `Microphone.names` implique une analyse détaillée du matériel ; la création du tableau peut donc prendre quelques secondes. Dans la plupart des cas, il suffit d'utiliser le microphone par défaut.

Exemple

Le code suivant renvoie des informations sur le tableau des périphériques audio.

```
allMicNames_array = Microphone.names;
_root.debugWindow += "Microphone.names a trouvé le(s) périphérique(s)
  suivant(s) :" + newline;
for(i=0; i < allMicNames_array.length; i++){
  debugWindow += "[" + i + "]: " + allMicNames[i] + newline;
}
```

Vous pouvez, par exemple, afficher les informations suivantes :

```
Microphone.names a trouvé le(s) périphérique(s) suivant(s) :
[0]: Crystal SoundFusion(tm)
[1]: Périphérique audio USB
```

Consultez également

Classe Array, `Microphone.name`

Microphone.onActivity

Disponibilité

Flash Player 6.

Usage

```
MicrophoneActif.onActivity = function(activité) {
  // vos instructions
}
```

Paramètres

activité Une valeur booléenne définie sur `true` lorsque le microphone commence à détecter du son et sur `false` lorsqu'il s'arrête.

Renvoie

Rien.

Description

Gestionnaire d'événement : invoqué lorsque le microphone commence ou arrête de détecter du son. Si vous souhaitez répondre à ce gestionnaire d'événement, vous devez créer une fonction afin de traiter sa valeur *activité*.

Pour déterminer le niveau sonore requis pour invoquer `Microphone.onActivity(true)` et la durée du silence devant s'écouler avant d'invoquer `Microphone.onActivity(false)`, utilisez `Microphone.setSilenceLevel()`.

Exemple

L'exemple suivant affiche `true` ou `false` dans le panneau de sortie lorsque le microphone commence ou arrête de détecter du son.

```
m = Microphone.get();
_root.attachAudio(m);
m.onActivity = function(mode)
{
  trace(mode);
};
```

Consultez également

`Microphone.onActivity`, `Microphone.setSilenceLevel()`

Microphone.onStatus

Disponibilité

Flash Player 6.

Usage

```
activeMicrophone.onStatus = function(objetInfo) {
  // vos instructions
}
```

Paramètres

objetInfo Un paramètre défini selon le message d'état.

Renvoie

Rien.

Description

Gestionnaire d'événement : invoqué lorsque l'utilisateur autorise ou refuse l'accès au microphone. Si vous voulez répondre à ce gestionnaire d'événement, vous devez créer une fonction pour traiter l'objet d'information généré par le microphone.

Lorsqu'un fichier SWF tente d'accéder au microphone, Flash Player affiche le panneau de contrôle de l'accès permettant à l'utilisateur d'autoriser ou de refuser l'accès au microphone.

- Si l'utilisateur autorise l'accès, la propriété `Microphone.muted` est définie sur la valeur `false` ; ce gestionnaire d'événement est invoqué avec un objet information dont la propriété `code` est `"Microphone.Unmuted"` et dont la propriété `niveau` est `"Etat"`.
- Si l'utilisateur refuse l'accès, la propriété `Microphone.muted` est définie sur la valeur `true` ; ce gestionnaire d'événement est invoqué avec un objet information dont la propriété `code` est `"Microphone.Muted"` et dont la propriété `niveau` est `"Etat"`.

Pour savoir si l'utilisateur a autorisé ou refusé l'accès au microphone sans traiter le gestionnaire d'événement, utilisez `Microphone.muted`.

Remarque : Si l'utilisateur choisit d'autoriser ou de refuser définitivement l'accès à tous les fichiers SWF à partir d'un domaine spécifique, cette méthode n'est pas invoquée pour les fichiers SWF de ce domaine, à moins que l'utilisateur ne modifie ultérieurement les paramètres de contrôle de l'accès. Pour plus d'informations, consultez `Microphone.get()`.

Exemple

Consultez l'exemple relatif à `Camera.onStatus`.

Consultez également

`Microphone.get()`, `Microphone.muted`

Microphone.rate

Disponibilité

Flash Player 6.

Usage

MicrophoneActif.rate

Description

Propriété (lecture seule) : le taux d'échantillonnage auquel le microphone capture le son, en kHz. La valeur par défaut est 8 kHz si le périphérique de capture du son supporte cette valeur. Sinon, la valeur par défaut correspond au premier niveau de capture au-delà de 8 kHz supporté par votre périphérique de capture du son. Il s'agit généralement de 11 kHz.

Pour définir cette valeur, utilisez `Microphone.setRate()`.

Exemple

L'exemple suivant enregistre le taux actuel dans la variable `original`.

```
original = monMic.rate;
```

Consultez également

`Microphone.setRate()`

Microphone.setGain()

Disponibilité

Flash Player 6.

Usage

MicrophoneActif.setGain(*gain*)

Paramètres

gain Un entier spécifiant le niveau d'amplification que le microphone doit appliquer au signal. Les valeurs valides sont comprises entre 0 et 100. La valeur par défaut est 50, mais l'utilisateur peut la modifier dans le panneau Microphone des paramètres Flash Player.

Renvoie

Rien.

Description

Méthode : définit le gain du microphone, c'est-à-dire le coefficient de multiplication que le microphone doit appliquer au signal avant de le transmettre. La valeur 0 multiplie le signal par zéro ; cela signifie que le microphone ne transmet aucun son.

Ce paramètre s'apparente au bouton de réglage du volume d'une chaîne hi-fi : 0 correspond à aucun volume et 50 au volume normal ; une valeur inférieure à 50 correspond à un volume plus faible que le volume normal et une valeur supérieure à 50 correspond à un volume plus élevé.

Exemple

L'exemple suivant veille à ce que le paramètre de gain du microphone soit inférieur ou égal à 55.

```
var monMic = Microphone.get();
if (monMic.gain > 55){
  monMic.setGain(55);
}
```

Consultez également

Microphone.gain, Microphone.setUseEchoSuppression()

Microphone.setRate()

Disponibilité

Flash Player 6.

Usage

MicrophoneActif.setRate(*kHz*)

Paramètres

kHz Le taux d'échantillonnage auquel le microphone capture le son, en kHz. Les valeurs valides sont 5, 8, 11, 22 et 44. La valeur par défaut est 8 kHz si le périphérique de capture du son supporte cette valeur. Sinon, la valeur par défaut correspond au premier niveau de capture au-delà de 8 kHz supporté par votre périphérique de capture du son. Il s'agit généralement de 11 kHz.

Renvoie

Rien.

Description

Méthode : définit le taux d'échantillonnage, en kHz, auquel le microphone capture le son.

Exemple

L'exemple suivant définit le taux d'échantillonnage conformément au choix de l'utilisateur (affecté à la variable `tauxUtilisateur`) s'il correspond à l'une des valeurs suivantes : 5, 8, 11, 22 ou 44. Si la valeur choisie par l'utilisateur ne correspond pas, elle est arrondie à la valeur supportée par le périphérique de capture du son la plus proche.

```
monMic.setRate(tauxUtilisateur);
```

Consultez également

`Microphone.rate`

Microphone.setSilenceLevel()

Disponibilité

Flash Player 6.

Usage

MicrophoneActif.setSilenceLevel(*niveau* [, *délai*])

Paramètres

niveau Un entier spécifiant le niveau sonore requis pour activer le microphone et invoquer Microphone.onActivity(true). Les valeurs valides sont comprises entre 0 et 100. La valeur par défaut est 10.

délai Un paramètre entier facultatif spécifiant le nombre de millisecondes d'inactivité devant s'écouler pour que Flash considère que le son s'est arrêté et invoque Microphone.onActivity(false). La valeur par défaut est 2 000 (2 secondes).

Renvoie

Rien.

Description

Méthode : définit le niveau sonore minimum considéré comme du son et (éventuellement) la durée de silence nécessaire pour considérer que le son est vraiment arrêté.

- Pour que le microphone ne détecte aucun son, affectez la valeur 100 au paramètre *niveau* ; Microphone.onActivity n'est jamais invoqué.
- Pour connaître le niveau sonore que le microphone détecte actuellement, utilisez Microphone.activityLevel.

La détection de l'activité consiste à surveiller les niveaux sonores pour détecter lorsqu'une personne est en train de parler. Elle permet d'économiser de la bande passante en évitant d'envoyer le flux audio lorsque personne ne parle. Ces informations peuvent également servir de retour visuel pour indiquer aux utilisateurs qu'ils (ou les autres utilisateurs) sont silencieux.

Les valeurs de silence sont la réciproque directe des valeurs d'activité. Le silence complet correspond à une valeur d'activité de 0. Un bruit fort et constant (aussi fort qu'il peut être enregistré, d'après le gain actuel) correspond à une valeur d'activité de 100. Une fois le gain correctement réglé, la valeur d'activité est inférieure à la valeur de silence lorsque vous ne parlez pas ; lorsque vous parlez, la valeur d'activité est supérieure à la valeur de silence.

Cette méthode a le même objectif que Camera.setMotionLevel() ; les deux méthodes permettent de déterminer à quel moment le gestionnaire d'événement onActivity doit être invoqué. Cependant, ces méthodes ont un impact très différent sur la publication en flux continus :

- Camera.setMotionLevel() est conçu pour détecter les mouvements et n'affecte pas l'utilisation de la bande passante. Même lorsqu'un flux continu vidéo ne détecte pas de mouvement, la vidéo est envoyée.
- Microphone.setSilenceLevel() est conçu pour optimiser la bande passante. Lorsqu'un flux continu audio est considéré comme silencieux, aucune donnée audio n'est envoyée. Un message unique est envoyé, indiquant le début du silence.

Exemple

L'exemple suivant permet à l'utilisateur de déterminer le niveau du silence. Le code suivant est affecté au bouton :

```
on (press)
{
  this.makeSilenceLevel(this.silenceLevel);
}
```

La fonction `makeSilenceLevel()` appelée par le bouton continue :

```
function makeSilenceLevel(s)
{
  this.obj.setSilenceLevel(s);
  this.SyncMode();
  this.silenceLevel= s;
}
```

Pour plus d'informations, consultez l'exemple relatif à `Camera.setMotionLevel()`.

Consultez également

`Microphone.activityLevel`, `Microphone.onActivity`, `Microphone.setGain()`, `Microphone.silenceLevel()`, `Microphone.silenceTimeout()`

Microphone.setUseEchoSuppression()

Disponibilité

Flash Player 6.

Usage

MicrophoneActif.setUseEchoSuppression(*supprimer*)

Paramètres

supprimer Une valeur booléenne indiquant si la fonction de suppression de l'écho doit être utilisée (`true`) ou non (`false`).

Renvoie

Rien.

Description

Méthode : spécifie l'utilisation ou non de la fonction de suppression de l'écho du codec audio. La valeur par défaut est `false` à moins que l'utilisateur ne sélectionne ReduceEcho dans le panneau Microphone des paramètres Flash Player.

La suppression de l'écho permet de réduire les effets de bouclage audio qui se produisent lorsque le son émis par le haut-parleur est capté par le microphone sur le même ordinateur. (A ne pas confondre avec la fonction d'annulation de l'écho, qui supprime totalement le retour.)

Il est généralement recommandé d'utiliser la suppression de l'écho lorsque le son capturé par le micro est diffusé par des haut-parleurs (et non par un casque) sur le même ordinateur. Si le fichier SWF permet aux utilisateurs de spécifier le périphérique de sortie audio, vous pouvez appeler `Microphone.setUseEchoSuppression(true)` s'ils utilisent conjointement les haut-parleurs et le microphone.

Les utilisateurs peuvent également régler ces paramètres dans le panneau Microphone des paramètres Flash Player.

Exemple

L'exemple suivant active la suppression de l'écho.

```
monmic.setUseEchoSuppression(true);
```

Consultez également

`Microphone.setGain()`, `Microphone.useEchoSuppression()`

Microphone.silenceLevel()

Disponibilité

Flash Player 6.

Usage

MicrophoneActif.silenceLevel

Description

Propriété (lecture seule) : un entier spécifiant le niveau sonore requis pour activer le microphone et invoquer `Microphone.onActivity(true)`. La valeur par défaut est 10.

Exemple

Consultez l'exemple relatif à `Microphone.silenceTimeout()`.

Consultez également

`Microphone.gain`, `Microphone.setSilenceLevel()`

Microphone.silenceTimeout()

Disponibilité

Flash Player 6.

Usage

MicrophoneActif.silenceTimeout

Description

Propriété (lecture seule) : valeur numérique représentant le nombre de millisecondes entre le moment où le microphone cesse de détecter du son et le moment où `Microphone.onActivity(false)` est invoqué. La valeur par défaut est 2 000 (2 secondes).

Pour définir cette valeur, utilisez `Microphone.setSilenceLevel()`.

Exemple

L'exemple suivant multiplie par deux la valeur actuelle du délai d'attente.

```
monMic.setSilenceLevel(monMic.silenceLevel, monMic.silenceTimeOut * 2);
```

Consultez également

`Microphone.setSilenceLevel()`

Microphone.useEchoSuppression()

Disponibilité

Flash Player 6.

Usage

MicrophoneActif.useEchoSuppression

Description

Propriété (lecture seule) : valeur booléenne définie sur la valeur `true` si la suppression de l'écho est activée et sur la valeur `false` dans le cas contraire. La valeur par défaut est `false` à moins que l'utilisateur ne sélectionne ReduceEcho dans le panneau Microphone des paramètres Flash Player.

Exemple

L'exemple suivant vérifie l'état de la suppression de l'écho et l'active si elle est désactivée.

```
_root.monMic.onActivity = function(active) {
  if (active == true) {
    if (_root.monMic.useEchoSuppression == false) {
_root.monMic.setUseEchoSuppression(true);
    }
  }
}
```

Consultez également

Microphone.setUseEchoSuppression()

MMExecute()

Disponibilité

Flash Player 7.

Usage

```
MMExecute("Commande API Flash JavaScript;")
```

Paramètres

Commande API Flash JavaScript Toute commande que vous pouvez utiliser dans un fichier Flash JavaScript (JSFL).

Renvoie

Le résultat, le cas échéant, est envoyé par l'instruction JavaScript.

Description

Fonction : vous permet d'émettre des commandes de l'API Flash JavaScript à partir d'ActionScript.

L'API Flash JavaScript (JSAPI) propose plusieurs objets, méthodes et propriétés pour dupliquer ou émuler des commandes qu'un utilisateur peut saisir dans l'environnement de programmation. Grâce à JSAPI, vous pouvez écrire des scripts qui étendent Flash de diverses manières : ajout de commandes aux menus, manipulation d'objets sur la scène, répétition de séquences de commandes, etc.

Un utilisateur exécute généralement un script JSAPI en sélectionnant Commandes > Exécuter la commande. Vous pouvez cependant utiliser cette fonction dans un script ActionScript pour appeler une commande JSAPI directement. Si vous utilisez `MMExecute()` dans un script sur l'image 1 de votre fichier, la commande est exécutée lorsque le fichier SWF est chargé.

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

Exemple

La commande suivante renvoie un tableau d'objets dans la bibliothèque :

```
var libe:Array = MMExecute("fl.getDocumentDOM().library.items;");
trace(libe.length + " articles de la bibliothèque");
```

Classe Mouse

Disponibilité

Flash Player 5.

Description

La classe Mouse est une classe de haut niveau : vous pouvez accéder à ses propriétés et à ses méthodes sans utiliser de constructeur. Utilisez les méthodes de la classe Mouse pour masquer et afficher le pointeur de la souris (curseur) dans le fichier SWF. Le pointeur est visible par défaut, mais vous pouvez le masquer et implémenter un pointeur personnalisé que vous créez avec un clip (consultez *Création d'un pointeur de souris personnalisé*, page 100).

Méthodes de la classe Mouse

Méthode	Description
Mouse.addListener()	Enregistre un objet pour la réception de notifications onMouseDown, onMouseMove et onMouseUp.
Mouse.hide()	Masque le pointeur de la souris dans le fichier SWF.
Mouse.removeListener()	Supprime un objet enregistré avec addListener().
Mouse.show()	Affiche le pointeur de la souris dans le fichier SWF.

Ecouteurs de la classe Mouse

Méthode	Description
Mouse.onMouseDown	Notifié lorsque le bouton de la souris est enfoncé.
Mouse.onMouseMove	Notifié lorsque la souris est déplacée.
Mouse.onMouseUp	Notifié lorsque le bouton de la souris est relâché.
Mouse.onMouseWheel	Notifié lorsque l'utilisateur manipule la molette de la souris.

Mouse.addListener()

Disponibilité

Flash Player 6.

Usage

```
Mouse.addListener (nouvelEcouteur)
```

Paramètres

nouvelEcouteur Un objet.

Renvoie

Rien.

Description

Méthode : enregistre un objet pour la réception de notifications d'écouteurs onMouseDown, onMouseMove et onMouseUp.

Le paramètre *nouvelEcouteur* doit contenir un objet avec des méthodes définies pour les écouteurs onMouseDown, onMouseMove et onMouseUp.

Lorsque le bouton de la souris est enfoncé, que la souris est déplacée ou que le bouton de la souris est relâché, quel que soit le focus de saisie, la méthode onMouseDown, onMouseMove ou onMouseUp de tous les objets écouteurs enregistrés avec cette méthode est invoquée. Plusieurs objets peuvent attendre des notifications de souris. Si l'écouteur *nouvelEcouteur* est déjà enregistré, aucun changement n'a lieu.

Consultez également

Mouse.onMouseDown, Mouse.onMouseMove, Mouse.onMouseUp

Mouse.hide()

Disponibilité

Flash Player 5.

Usage

```
Mouse.hide()
```

Paramètres

Aucun.

Renvoie

Une valeur booléenne : `true` si le pointeur est visible et `false` dans le cas contraire.

Description

Méthode : masque le pointeur dans un fichier SWF. Le pointeur est visible par défaut.

Exemple

Le code suivant, associé à un clip du scénario principal, masque le pointeur standard et définit les positions *x* et *y* de l'occurrence de clip `PointeurPerso_mc` aux positions *x* et *y* de la souris dans le scénario principal.

```
onClipEvent (enterFrame) {
  Mouse.hide();
  PointeurPerso_mc._x = _root._xmouse;
  PointeurPerso_mc._y = _root._ymouse;
}
```

Consultez également

`Mouse.show()`, `MovieClip._xmouse`, `MovieClip._ymouse`

Mouse.onMouseDown

Disponibilité

Flash Player 6.

Usage

unEcouteur.onMouseDown

Paramètres

Aucun.

Renvoie

Rien.

Description

Ecouteur : notifié lorsque le bouton de la souris est enfoncé. Pour utiliser l'écouteur onMouseDown, vous devez créer un objet d'écoute. Vous pouvez alors définir une fonction pour onMouseDown et utiliser addListener() pour enregistrer l'écouteur avec l'objet Mouse, comme dans l'exemple suivant :

```
unEcouteur = new Object();
unEcouteur.onMouseDown = function () { ... };
Mouse.addListener(unEcouteur);
```

Les écouteurs permettent à différents morceaux de code de coopérer étant donné que plusieurs écouteurs peuvent recevoir une notification concernant un seul événement.

Consultez également

Mouse.addListener()

Mouse.onMouseMove

Disponibilité

Flash Player 6.

Usage

unEcouteur.onMouseMove

Paramètres

Aucun.

Renvoie

Rien.

Description

Ecouteur : notifié lorsque la souris se déplace. Pour utiliser l'écouteur `onMouseMove`, vous devez créer un objet d'écoute. Vous pouvez alors définir une fonction pour `onMouseMove` et utiliser `addListener()` pour enregistrer l'écouteur avec l'objet Mouse, comme dans l'exemple suivant :

```
unEcouteur = new Object();
unEcouteur.onMouseMove = function () { ... };
Mouse.addListener(unEcouteur);
```

Les écouteurs permettent à différents morceaux de code de coopérer étant donné que plusieurs écouteurs peuvent recevoir une notification concernant un seul événement.

Consultez également

`Mouse.addListener()`

Mouse.onMouseUp

Disponibilité

Flash Player 6.

Usage

unEcouteur.onMouseUp

Paramètres

Aucun.

Renvoie

Rien.

Description

Ecouteur : notifié lorsque le bouton de la souris est relâché. Pour utiliser l'écouteur `onMouseUp`, vous devez créer un objet d'écoute. Vous pouvez alors définir une fonction pour `onMouseUp` et utiliser `addListener()` pour enregistrer l'écouteur avec l'objet Mouse, comme dans l'exemple suivant :

```
unEcouteur = new Object();
unEcouteur.onMouseUp = function () { ... };
Mouse.addListener(unEcouteur);
```

Les écouteurs permettent à différents morceaux de code de coopérer étant donné que plusieurs écouteurs peuvent recevoir une notification concernant un seul événement.

Consultez également

`Mouse.addListener()`

Mouse.onMouseWheel

Disponibilité

Flash Player 7 (Windows uniquement).

Usage

```
unEcouteur.onMouseWheel = function ( [ delta , scrollTarget ]) {
  // vos instructions
}
```

Paramètres

delta Un nombre facultatif indiquant le nombre de lignes qui défilent pour chaque cran de la molette de la souris que l'utilisateur fait tourner. Une valeur *delta* positive indique que le défilement se fait vers le haut ; une valeur négative indique que le défilement se fait vers le bas. Cette valeur se situe généralement entre 1 et 3. Un défilement plus rapide peut produire des valeurs plus grandes.

Si vous ne souhaitez pas spécifier de valeur pour *delta* mais que vous souhaitez en spécifier une pour *scrollTarget*, définissez `null` pour *delta*.

scrollTarget La première occurrence de clip sous le pointeur une fois la molette de la souris tournée.

Renvoie

Rien.

Description

Ecouteur : notifié lorsque l'utilisateur manipule la molette de la souris. Pour utiliser l'écouteur `onMouseWheel`, vous devez créer un objet d'écoute. Vous pouvez ensuite définir une fonction pour `onMouseWheel` et utiliser `addListener()` pour enregistrer l'écouteur avec l'objet Mouse.

Remarque : Les écouteurs d'événement molette de souris sont disponibles uniquement dans les versions Windows de Flash Player.

Exemple

L'exemple suivant montre comment créer un objet écouteur qui réagit aux événements de mollette de souris. Dans cet exemple, la coordonnée *x* d'un objet clip appelé `clip_mc` (non illustré) change à chaque fois que l'utilisateur fait tourner la mollette de la souris.

```
EcouteurSouris = new Object();
EcouteurSouris.onMouseWheel = function(delta) {
  clip_mc._x += delta;
}
Mouse.addListener(EcouteurSouris);
```

Consultez également

`Mouse.addListener()`, `TextField.mouseWheelEnabled`

Mouse.removeListener()

Disponibilité

Flash Player 6.

Usage

```
Mouse.removeListener (écouteur)
```

Paramètres

`écouteur` Un objet.

Renvoie

Si l'objet *écouteur* a été correctement retiré, la méthode renvoie `true`. Si *écouteur* n'a pas été correctement retiré, par exemple si *écouteur* n'apparaissait pas dans la liste des écouteurs de l'objet Mouse, la méthode renvoie `false`.

Description

Méthode : supprime un objet précédemment enregistré avec `addListener()`.

Mouse.show()

Disponibilité

Flash Player 5.

Usage

```
Mouse.show()
```

Paramètres

Aucun.

Renvoie

Rien.

Description

Méthode : affiche le pointeur de la souris dans un fichier SWF. Le pointeur est visible par défaut.

Consultez également

`Mouse.show()`, `MovieClip._xmouse`, `MovieClip._ymouse`

Classe MovieClip

Disponibilité

Flash Player 3.

Description

Les méthodes de la classe MovieClip fournissent les mêmes fonctionnalités que les actions ciblant les clips. Il existe également des méthodes supplémentaires qui n'ont pas d'action équivalente dans la boîte à outils Actions du panneau Actions.

Il n'est pas nécessaire d'utiliser une méthode de constructeur pour appeler les méthodes de la classe MovieClip ; faites référence aux occurrences de clips par leurs noms, avec la syntaxe suivante :

```
mon_mc.play();
mon_mc.gotoAndPlay(3);
```

Méthodes de la classe MovieClip

Méthode	Description
MovieClip.attachAudio()	Capture et joue du son local à partir d'un microphone.
MovieClip.attachMovie()	Associe un fichier SWF dans la bibliothèque.
MovieClip.createEmptyMovieClip()	Crée un clip vide.
MovieClip.createTextField()	Crée un champ de texte vide.
MovieClip.duplicateMovieClip	Duplique le clip spécifié.
MovieClip.getBounds	Renvoie les coordonnées *x* et *y* minimum et maximum d'un fichier SWF dans un espace de coordonnées spécifié.
MovieClip.getBytesLoaded()	Renvoie le nombre d'octets chargés pour le clip spécifié.
MovieClip.getBytesTotal()	Renvoie la taille du clip, en octets.
MovieClip.getDepth()	Renvoie la profondeur d'un clip.
MovieClip.getInstanceAtDepth()	Indique si une profondeur particulière est déjà occupée par un clip.
MovieClip.getNextHighestDepth()	Indique une valeur de profondeur que vous pouvez transmettre à d'autres méthodes pour vous assurer que Flash effectue un rendu du clip devant tous les objets du clip actuel.
MovieClip.getSWFVersion()	Renvoie un entier indiquant la version deFlash Player pour laquelle le clip a été publié.
MovieClip.getTextSnapshot()	Renvoie un objet TextSnapshot contenant le texte dans les champs de texte statiques du clip spécifié.
MovieClip.getURL()	Récupère un document d'une URL.
MovieClip.globalToLocal()	Convertit l'objet de point des coordonnées de scène en coordonnées locales du clip spécifié.
MovieClip.gotoAndPlay()	Envoie la tête de lecture vers une image spécifique du clip et lit le fichier SWF.
MovieClip.gotoAndStop	Envoie la tête de lecture vers une image spécifique du clip et arrête le fichier SWF.
MovieClip.hitTest()	Renvoie `true` si le cadre de délimitation du clip spécifié croise le cadre de délimitation du clip cible.
MovieClip.loadMovie	Charge le fichier SWF spécifié dans le clip.
MovieClip.loadVariables()	Charge les variables depuis une URL ou un autre emplacement dans le clip.

Méthode	Description
MovieClip.localToGlobal()	Convertit un objet de point des coordonnées locales du clip en coordonnées globales de scène.
MovieClip.nextFrame()	Envoie la tête de lecture à l'image suivante du clip.
MovieClip.play()	Exécute le clip spécifié.
MovieClip.prevFrame()	Envoie la tête de lecture à l'image précédente du clip.
MovieClip.removeMovieClip()	Supprime le clip du scénario s'il a été créé avec duplicateMovieClip(), MovieClip.duplicateMovieClip ou MovieClip.attachMovie().
MovieClip.setMask()	Spécifie un clip comme masque d'un autre clip.
MovieClip.startDrag()	Spécifie un clip comme déplaçable et commence à déplacer le clip.
MovieClip.stop()	Arrête la lecture du fichier SWF actuel.
MovieClip.stopDrag()	Arrête le déplacement d'un clip en cours de déplacement.
MovieClip.swapDepths()	Echange le niveau de profondeur de deux fichiers SWF.
MovieClip.unloadMovie()	Supprime un fichier SWF chargé avec loadMovie().

Méthodes de dessin de la classe MovieClip

Méthode	Description
MovieClip.beginFill()	Commence à tracer un remplissage sur la scène.
MovieClip.beginGradientFill()	Commence à tracer un remplissage en dégradé sur la scène.
MovieClip.clear()	Supprime toutes les commandes de dessin associées à une occurrence de clip.
MovieClip.curveTo()	Dessine une ligne avec le style de trait le plus récent.
MovieClip.endFill()	Termine le remplissage spécifié par beginFill() ou beginGradientFill().
MovieClip.lineStyle()	Définit le trait des lignes créées avec les méthodes lineTo() et curveTo().
MovieClip.lineTo	Dessine une ligne avec le style de trait actuel.
MovieClip.moveTo	Déplace la position de dessin actuelle aux coordonnées spécifiées.

Propriétés de la classe MovieClip

Propriété	Description
MovieClip._alpha	La valeur de transparence d'une occurrence de clip.
MovieClip._currentframe	Le numéro de l'image dans laquelle la tête de lecture est actuellement située.
MovieClip._droptarget	Le chemin absolu (avec la syntaxe à barre oblique) de l'occurrence de clip sur laquelle un clip déplaçable a été déposé.
MovieClip.enabled	Indique si un clip de bouton est activé.
MovieClip.focusEnabled	Active un clip pour qu'il puisse recevoir le focus.
MovieClip._focusrect	Indique si un clip avec focus est encadré d'un rectangle jaune.
MovieClip._framesloaded	Le nombre d'images chargées à partir d'un fichier SWF en flux continu.
MovieClip._height	La hauteur d'une occurrence de clip, en pixels.
MovieClip.hitArea	Désigne un autre clip qui servira de zone active pour un clip de bouton.
MovieClip._highquality	Définit la qualité du rendu d'un fichier SWF.
MovieClip.menu	Associe un objet ContextMenu à un clip.
MovieClip._name	Le nom d'occurrence d'une occurrence de clip.
MovieClip._parent	Une référence au clip contenant le clip.
MovieClip._rotation	Le degré de rotation d'une occurrence de clip.
MovieClip._soundbuftime	Le nombre de secondes avant d'un son ne démarre en lecture continue.
MovieClip.tabChildren	Indique si les enfants d'un clip sont inclus dans l'ordre de tabulation automatique.
mon_mc.tabEnabled	Indique si un clip est inclus dans l'ordre de tabulation.
MovieClip.tabIndex	Indique l'ordre de tabulation d'un objet.
MovieClip._target	Le chemin cible d'une occurrence de clip.
MovieClip._totalframes	Le nombre total d'images d'une occurrence de clip.
MovieClip.trackAsMenu	Indique si d'autres boutons peuvent recevoir des événements de relâchement du bouton de la souris.
MovieClip._url	L'URL du fichier SWF d'où le clip a été téléchargé.
MovieClip.useHandCursor	Détermine si le curseur de main est affiché lorsque la souris passe au-dessus d'un clip bouton.
MovieClip._visible	Une valeur booléenne déterminant si l'occurrence d'un clip est masquée ou visible.
MovieClip._width	La largeur d'une occurrence de clip, en pixels.

Propriété	Description
MovieClip._x	La coordonnée *x* d'une occurrence de clip.
MovieClip._xmouse	La coordonnée *x* du pointeur de la souris dans une occurrence de clip.
MovieClip._xscale	La valeur spécifiant le pourcentage de redimensionnement horizontal d'un clip.
MovieClip._y	La coordonnée *y* d'une occurrence de clip.
MovieClip._ymouse	La coordonnée *y* du pointeur de la souris dans une occurrence de clip.
MovieClip._yscale	La valeur spécifiant le pourcentage de redimensionnement vertical d'un clip.

Gestionnaires d'événement de la classe MovieClip

Gestionnaire d'événement	Description
MovieClip.onData	Invoqué lorsque toutes les données sont chargées dans un clip.
MovieClip.onDragOut	Invoqué lorsque le pointeur se trouve en dehors du bouton, le bouton de la souris est enfoncé à l'intérieur, puis le pointeur sort de la zone du bouton.
MovieClip.onDragOver	Invoqué lorsque le pointeur se trouve au-dessus du bouton, le bouton de la souris a été enfoncé puis que le pointeur sort du bouton et ramené au-dessus du bouton.
MovieClip.onEnterFrame	Invoqué de manière continue à la cadence du fichier SWF. Les actions associées à l'événement de clip enterFrame sont traitées avant les actions associées aux images affectées.
MovieClip.onKeyDown	Invoqué lorsqu'une touche est enfoncée. Utilisez les méthodes Key.getCode() et Key.getAscii() pour récupérer les informations concernant la dernière touche enfoncée.
MovieClip.onKeyUp	Invoqué lorsqu'une touche est relâchée.
MovieClip.onKillFocus	Invoqué lorsque le focus est retiré d'un bouton.
MovieClip.onLoad	Invoqué lorsque le clip est instancié et apparaît dans le scénario.
MovieClip.onMouseDown	Invoqué lorsque le bouton gauche de la souris est enfoncé.
MovieClip.onMouseMove	Invoqué à chaque fois que la souris est déplacée.
MovieClip.onMouseUp	Invoqué lorsque le bouton gauche de la souris est relâché.
MovieClip.onPress	Invoqué lorsque le bouton de la souris est enfoncé alors que le pointeur au-dessus d'un bouton.
MovieClip.onRelease	Invoqué lorsque le bouton de la souris est relâché alors que le pointeur au-dessus d'un bouton.

Gestionnaire d'événement	Description
MovieClip.onReleaseOutside	Invoqué lorsque la souris est relâchée alors que le pointeur se trouve au dehors du bouton après l'enfoncement du bouton pendant que le pointeur est à l'intérieur du bouton.
MovieClip.onRollOut	Invoqué lorsque le pointeur passe à l'extérieur d'un bouton.
MovieClip.onRollOver	Invoqué lorsque le pointeur de la souris passe au-dessus d'un bouton.
MovieClip.onSetFocus	Invoqué lorsqu'un bouton a le focus de saisie et qu'une touche est relâchée.
MovieClip.onUnload	Invoqué dans la première image après la suppression du clip du scénario. Les actions associées à l'événement de clip Unload sont traitées avant que des actions ne soient associées à l'image affectée.

MovieClip._alpha

Disponibilité

Flash Player 4.

Usage

mon_mc._alpha

Description

Propriété : la valeur de transparence alpha du clip spécifié par *mon_mc*. Les valeurs valides vont de 0 (transparence complète) à 100 (opacité complète). La valeur par défaut est 100. Les objets d'un clip avec _alpha défini sur 0 sont actifs, même s'ils sont invisibles. Par exemple, vous pouvez cliquer sur un bouton dans un clip dont la propriété _alpha est définie sur 0.

Exemple

Le code suivant définit la propriété _alpha d'un clip appelé star_mc à 30 % lorsque l'utilisateur clique sur le bouton :

```
on(release) {
  star_mc._alpha = 30;
}
```

Consultez également

Button._alpha, TextField._alpha

MovieClip.attachAudio()

Disponibilité

Flash Player 6 ; la possibilité de joindre du son à partir de fichiers Flash Video (FLV) a été ajoutée dans Flash Player 7.

Usage

```
mon_mc.attachAudio(source)
```

Paramètres

source L'objet contenant le son à jouer. Les valeurs valides sont un objet Microphone, un objet NetStream qu'un fichier FLV lit et `false` (arrête la lecture du son).

Renvoie

Rien.

Description

Méthode : spécifie la source audio à lire. Pour arrêter la lecture de la source audio, affectez `false` à *source*.

Exemple

Le code suivant associe un microphone à un clip.

```
mon_mic = Microphone.get();
this.attachAudio(mon_mic);
```

L'exemple suivant montre comment vous pouvez utiliser un objet Sound pour contrôler le son associé à un fichier FLV.

```
// Clip est le nom d'occurrence du clip
// qui contient l'objet vidéo "ma_vidéo".
_root.Clip.ma_vidéo.attachVideo(_root.myNetStream);
_root.Clip.attachAudio(_root.myNetStream);
var snd = new Sound("_root.Clip");
//Pour régler le son :
_root.snd.setVolume(100);
```

Consultez également

Classe Microphone, `NetStream.play()`, Classe Sound, `Video.attachVideo()`

MovieClip.attachMovie()

Disponibilité

Flash Player 5.

Usage

```
mon_mc.attachMovie(nomIdentifiant, nouveauNom, profondeur [, objetInit])
```

Paramètres

nomIdentifiant Le 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.

nouveauNom Un nom d'occurrence unique pour le clip en cours d'association avec le clip.

profondeur Un entier spécifiant le niveau de profondeur auquel associer le fichier SWF.

objetInit (supporté par Flash Player 6 et ultérieur) Un objet contenant les propriétés avec lesquelles remplir le clip nouvellement associé. Ce paramètre permet aux clips créés dynamiquement de recevoir des paramètres. Si *objetInit* n'est pas un objet, il est ignoré. Toutes les propriétés de *objetInit* sont copiées dans la nouvelle occurrence. Les propriétés spécifiées avec *objetInit* peuvent être utilisées par la fonction constructeur. Ce paramètre est facultatif.

Renvoie

Une référence à la nouvelle occurrence.

Description

Méthode : prend un symbole de la bibliothèque et l'associe au fichier SWF spécifié par *mon_mc* sur la scène. Utilisez `removeMovieClip()` ou `unloadMovie()` pour supprimer un fichier SWF associé à `attachMovie()`.

Exemple

L'exemple suivant associe le symbole dont l'identifiant de liaison est un « cercle » à l'occurrence de clip qui se trouve sur la scène du fichier SWF.

```
on (release) {
    truc.attachMovie( "cercle", "cercle1", 2 );
}
```

Consultez également

`MovieClip.removeMovieClip()`, `MovieClip.unloadMovie()`, `Object.registerClass()`, `removeMovieClip()`

MovieClip.beginFill()

Disponibilité

Flash Player 6.

Usage

```
mon_mc.beginFill([rvb[, alpha]])
```

Paramètre

rvb Une valeur chromatique hexadécimale (par exemple, le rouge correspond à 0xFF0000, le bleu correspond à 0x0000FF, etc.). Si cette valeur n'est pas fournie ou si elle est indéfinie, aucun remplissage n'est créé.

alpha Un entier compris entre 0–100 spécifiant la valeur alpha du remplissage. Si cette valeur n'est pas indiquée, 100 (uni) est utilisé. Si la valeur est inférieure à 0, Flash utilise 0. Si la valeur est supérieure à 100, Flash utilise 100.

Renvoie

Rien.

Description

Méthode : indique le début d'un nouveau trajet de dessin. Si un chemin ouvert existe, c'est-à-dire si la position de dessin actuelle n'est pas égale à la position précédente spécifiée dans une méthode `moveTo()`, et qu'un remplissage y est associé, ce chemin est fermé avec un trait, puis rempli. Cela est semblable à ce qui se produit à l'appel de `endFill()`. Si aucun remplissage n'est actuellement associé au chemin, `endFill()` doit être appelé pour appliquer le remplissage.

Consultez également

`MovieClip.beginGradientFill()`, `MovieClip.endFill()`

MovieClip.beginGradientFill()

Disponibilité

Flash Player 6.

Usage

```
mon_mc.beginGradientFill(typeDeRemplissage, couleurs, alphas, rapports,
matrice)
```

Paramètre

typeDeRemplissage La chaîne `"linear"` ou la chaîne "radial".

couleurs Un tableau des valeurs hexadécimales RVB à utiliser dans le dégradé (par exemple, le rouge correspond à 0xFF0000, le bleu correspond à 0x0000FF, etc.).

alphas Un tableau de valeurs alpha pour les couleurs correspondantes du tableau *couleurs*, les valeurs valides sont comprises entre 0–100. Si la valeur est inférieure à 0, Flash utilise 0. Si la valeur est supérieure à 100, Flash utilise 100.

rapports Un tableau de rapports de distribution des couleurs, les valeurs valides étant 0–255. Cette valeur définit le pourcentage de la largeur où la couleur est échantillonnée à 100 %.

matrice Une matrice de transformation qui est un objet avec l'un des deux ensembles de propriétés suivants.

- *a*, *b*, *c*, *d*, *e*, *f*, *g*, *h*, *i*, qui peuvent être utilisés pour décrire une matrice 3x3 au format suivant :

  ```
  a b c
  d e f
  g h i
  ```

  L'exemple suivant utilise une méthode `beginGradientFill()` avec un paramètre *matrice*, qui est un objet avec ces propriétés.

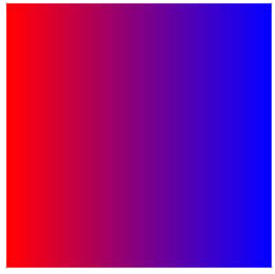
  ```
  _root.createEmptyMovieClip( "dégradé", 1 );
  with ( _root.dégradé )

    {

    couleurs = [ 0xFF0000, 0x0000FF ];
    alphas = [ 100, 100 ];
    rapports = [ 0, 0xFF ];
    matrice = { a:200, b:0, c:0, d:0, e:200, f:0, g:200, h:200, i:1 };
    beginGradientFill( "linear", couleurs, alphas, rapports, matrice );
    moveto(100,100);
    lineto(100,300);
    lineto(300,300);
    lineto(300,100);
    lineto(100,100);
    endFill();

    }
  ```

  Si une propriété *typeDeMatrice* n'existe pas, les autres paramètres sont tous requis (la fonction échoue s'ils ne sont pas tous présents). Cette matrice redimensionne, traduit, pivote et incline le gradient unitaire qui est défini à (-1,-1) et (1,1).

- *typeDeMatrice*, *x*, *y*, *l*, *h*, *r*.

  Les propriétés indiquent : *typeDeMatrice* est la chaîne `"box"`, *x* est la position horizontale relative au point d'alignement du clip parent pour le coin supérieur gauche du dégradé, *y* est la position verticale relative au point d'alignement du clip parent pour le coin supérieur gauche du dégradé, *l* est la largeur du gradient, *h* est la hauteur du dégradé et *r* est la rotation du dégradé, en radians.

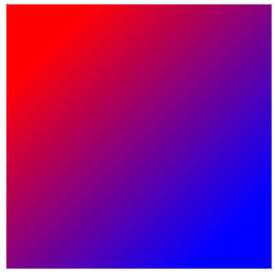
  L'exemple suivant utilise une méthode `beginGradientFill()` avec un paramètre *matrice*, qui est un objet avec ces propriétés.

```
_root.createEmptyMovieClip( "dégradé", 1 );
              with ( _root.dégradé )

              {

              couleurs = [ 0xFF0000, 0x0000FF ];
              alphas = [ 100, 100 ];
              rapports = [ 0, 0xFF ];
              matrice = { typeDeMatrice:"box", x:100, y:100, l:200, h:200,
  r:(45/180)*Math.PI };
              beginGradientFill( "linear", couleurs, alphas, rapports,
  matrice );
              moveto(100,100);
              lineto(100,300);
              lineto(300,300);
              lineto(300,100);
              lineto(100,100);
              endFill();
              }
```

Si une propriété *typeDeMatrice* existe, elle doit être égale à `"box"` et les autres paramètres sont tous obligatoires. La fonction échoue si une ou plusieurs de ces conditions ne sont pas remplies.



Renvoie

Rien.

Description

Méthode : indique le début d'un nouveau trajet de dessin. Si le premier paramètre est `undefined`, ou si aucun paramètre n'est transmis, le trajet n'a aucun remplissage. Si un chemin ouvert existe (si la position actuelle du dessin n'est pas égale à la position précédente spécifiée dans une méthode `moveTo()`) et qu'un remplissage y est associé, ce chemin est fermé avec un trait, puis rempli. Cela est semblable à ce qui se produit à l'appel de `endFill()`.

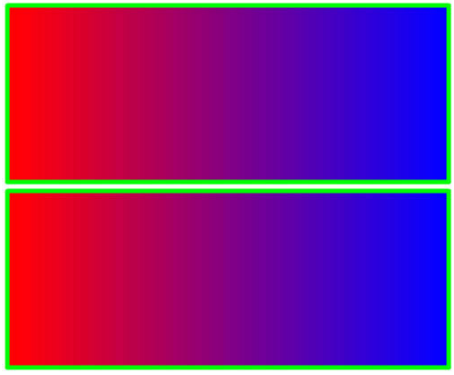
Cette méthode échoue si une ou plusieurs des conditions suivantes existent :

- Le nombre d'éléments dans les paramètres *couleurs*, *alphas* et *rapports* n'est pas le même.
- Le paramètre *typeDeRemplissage* n'est ni `"linear"` ni `"radial"`.
- Un ou plusieurs des champs de l'objet pour le paramètre *matrice* sont absents ou invalides.

Exemple

Le code suivant utilise les deux méthodes pour tracer deux rectangles empilés avec un remplissage dégradé rouge-bleu et un trait vert de 5 points.

```
_root.createEmptyMovieClip("goober",1);
with ( _root.goober )
{
  couleurs = [ 0xFF0000, 0x0000FF ];
  alphas = [ 100, 100 ];
  rapports = [ 0, 0xFF ];
  lineStyle( 5, 0x00ff00 );
  matrice = { a:500,b:0,c:0,d:0,e:200,f:0,g:350,h:200,i:1};
  beginGradientFill( "linear", couleurs, alphas, rapports, matrice );
  moveto(100,100);
  lineto(100,300);
  lineto(600,300);
  lineto(600,100);
  lineto(100,100);
  endFill();
  matrice = { typeDeMatrice:"box", x:100, y:310, l:500, h:200, r:(0/
  180)*Math.PI };
  beginGradientFill( "linear", couleurs, alphas, rapports, matrice );
  moveto(100,310);
  lineto(100,510);
  lineto(600,510);
  lineto(600,310);
  lineto(100,310);
  endFill();
}
```



Consultez également

MovieClip.beginFill(), MovieClip.endFill(), MovieClip.lineStyle(), MovieClip.lineTo, MovieClip.moveTo

MovieClip.clear()

Disponibilité

Flash Player 6.

Usage

mon_mc.clear()

Paramètres

Aucun.

Renvoie

Rien.

Description

Méthode : supprime tous les graphiques créés à l'exécution à l'aide des méthodes de dessin de la classe MovieClip, y compris les styles de traits spécifiés avec MovieClip.lineStyle(). Les formes et les lignes dessinées à main levée pendant la phase de programmation (avec les outils de dessin Flash) ne sont pas supprimées.

Consultez également

MovieClip.lineStyle()

MovieClip.createEmptyMovieClip()

Disponibilité

Flash Player 6.

Usage

mon_mc.createEmptyMovieClip(*nomDoccurrence*, *profondeur*)

Paramètres

nomDoccurrence Une chaîne identifiant le nom d'occurrence du nouveau clip.

profondeur Un entier spécifiant la profondeur du nouveau clip.

Renvoie

Une référence au nouveau clip.

Description

Méthode : crée un clip vide comme enfant d'un clip existant. Cette méthode a le même comportement que la méthode attachMovie(), mais vous n'avez pas besoin de fournir un nom 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 un ou plusieurs des paramètres sont absents.

Consultez également

MovieClip.attachMovie()

MovieClip.createTextField()

Disponibilité

Flash Player 6.

Usage

```
mon_mc.createTextField(nomDoccurrence, profondeur, x, y, largeur, hauteur)
```

Paramètres

nomDoccurence Une chaîne identifiant le nom d'occurrence du nouveau champ de texte.

profondeur Un entier positif spécifiant la profondeur du nouveau champ de texte.

x Un entier spécifiant la coordonnée *x* du nouveau champ de texte.

y Un entier spécifiant la coordonnée *y* du nouveau champ de texte.

largeur Un entier positif spécifiant la largeur du nouveau champ de texte.

hauteur Un entier positif spécifiant la hauteur du nouveau champ de texte.

Renvoie

Rien.

Description

Méthode : crée un champ de texte vide comme enfant du clip spécifié par *mon_mc*. Vous pouvez utiliser `createTextField()` pour créer des champs de texte lors de la lecture d'un fichier SWF. Le champ de texte est placé à (*x*, *y*) avec les dimensions *largeur* x *hauteur*. Les paramètres `x` et `y` sont relatifs au clip container et correspondent aux propriétés `_x` et `_y` du champ de texte. Les paramètres *largeur* et *hauteur* correspondent aux propriétés `_width` et `_height` du champ de texte.

Les propriétés par défaut d'un champ de texte sont les suivantes :

```
type = "dynamic"
border = false
background = false
password = false
multiline = false
html = false
embedFonts = false
variable = null
maxChars = null
```

Un champ de texte créé avec `createTextField()` reçoit l'objet TextFormat par défaut suivant :

```
font = "Times New Roman"
size = 12
textColor = 0x000000
bold = false
italic = false
underline = false
url = ""
target = ""
align = "left"
leftMargin = 0
rightMargin = 0
indent = 0
leading = 0
bullet = false
tabStops = [] (tableau vide)
```

Exemple

L'exemple suivant crée un champ de texte avec une largeur de 300, une hauteur de 100, une coordonnée *x* de 100, une coordonnée *y* de 100, aucune bordure, texte rouge et souligné.

```
_root.createTextField("monTexte",1,100,100,300,100);
monTexte.multiline = true;
monTexte.wordWrap = true;
monTexte.border = false;

monFormat = new TextFormat();
monFormat.color = 0xff0000;
monFormat.bullet = false;
monFormat.underline = true;

monTexte.text = "Ceci est mon premier test de texte d'objet de champ de texte";
monTexte.setTextFormat(monFormat);
```

Consultez également

Classe TextFormat

MovieClip._currentframe

Disponibilité

Flash Player 4.

Usage

```
mon_mc._currentframe
```

Description

Propriété (lecture seule) : renvoie le numéro de l'image où se trouve la tête de lecture dans le scénario spécifié par *mon_mc*.

Exemple

L'exemple suivant utilise la propriété `_currentframe` pour faire avancer la tête de lecture du clip `actionClip_mc` de cinq images par rapport à son emplacement actuel.

```
actionClip_mc.gotoAndStop(_currentframe + 5);
```

MovieClip.curveTo()

Disponibilité

Flash Player 6.

Usage

mon_mc.curveTo(*contrôleX*, *contrôleY*, *ancreX*, *ancreY*)

Paramètres

contrôleX Un entier spécifiant une position horizontale par rapport au point d'alignement du clip parent du point de contrôle.

contrôleY Un entier spécifiant une position verticale par rapport au point d'alignement du clip parent du point de contrôle.

ancreX Un entier spécifiant une position horizontale par rapport au point d'alignement du clip parent du point d'ancrage suivant.

ancreY Un entier spécifiant une position verticale par rapport au point d'alignement du clip parent du point d'ancrage suivant.

Renvoie

Rien.

Description

Méthode : dessine une courbe avec le style de trait courant depuis la position de dessin actuelle à (*ancreX*, *ancreY*), utilisant le point spécifié par (*contrôleX*, *contrôleY*). La position de dessin actuelle est alors définie sur (*ancreX*, *ancreY*). Si le clip dans lequel vous dessinez contient un contenu créé avec les outils de dessin Flash, les appels à `curveTo()` sont dessinés au-dessous de ce contenu. Si vous appelez `curveTo()` avant tout appel à `moveTo()`, la position de dessin actuelle est par défaut (0, 0). Cette méthode échoue lorsque des paramètres sont absents et la position de dessin courante n'est pas changée.

Exemple

L'exemple suivant dessine un cercle avec un trait bleu très fin et un remplissage uni rouge.

```
_root.createEmptyMovieClip( "cercle", 1 );
with ( _root.cercle )
{
  lineStyle( 0, 0x0000FF, 100 );
  beginFill( 0xFF0000 );
  moveTo( 500, 500 );
  curveTo( 600, 500, 600, 400 );
  curveTo( 600, 300, 500, 300 );
  curveTo( 400, 300, 400, 400 );
  curveTo( 400, 500, 500, 500 );
  endFill();
}
```

Consultez également

MovieClip.beginFill(), MovieClip.createEmptyMovieClip(), MovieClip.endFill(), MovieClip.lineStyle(), MovieClip.lineTo, MovieClip.moveTo

MovieClip._droptarget

Disponibilité

Flash Player 4.

Usage

mon_mc._droptarget

Description

Propriété (lecture seule) : renvoie le chemin absolu avec la syntaxe à barre oblique de l'occurrence de clip sur laquelle *mon_mc* a été déposé. La propriété _droptarget renvoie toujours un chemin commençant par une barre oblique (/). Pour comparer la propriété _droptarget d'une occurrence avec une référence, utilisez la fonction eval() pour convertir la valeur renvoyée de la syntaxe à barre oblique en une référence avec syntaxe à point.

Remarque : Vous devez réaliser cette conversion si vous utilisez ActionScript 2.0 qui ne prend pas en charge la syntaxe à barre oblique.

Exemple

L'exemple suivant évalue la propriété _droptarget de l'occurrence de clip tasDeTrucs et utilise eval() pour la convertir d'une syntaxe à barre oblique en une référence avec syntaxe à point. La référence tasDeTrucs est ensuite comparée avec la référence à l'occurrence de clip poubelle. Si les deux références sont équivalentes, la visibilité de tasDeTrucs est définie sur false. Si elles ne sont pas équivalentes, l'occurrence tasDeTrucs est redéfinie sur sa position d'origine.

```
if (eval(tasDeTrucs._droptarget) == _root.poubelle) {
  tasDeTrucs._visible = false;
} else {
  tasDeTrucs._x = x_pos;
  tasDeTrucs._y = y_pos;
}
```

Les variables x_pos et y_pos sont définies dans l'image 1 du fichier SWF à l'aide du script suivant :

```
x_pos = tasDeTrucs._x;
y_pos = tasDeTrucs._y;
```

Consultez également

startDrag()

MovieClip.duplicateMovieClip

Disponibilité

Flash Player 5.

Usage

```
mon_mc.duplicateMovieClip(nouveauNom, profondeur [,objetInit])
```

Paramètres

nouveauNom Un identifiant unique pour le clip dupliqué.

profondeur Un nombre unique indiquant la profondeur à laquelle le fichier SWF spécifié doit être placé.

objetInit (Supporté par Flash Player 6 et ultérieur.) Un objet contenant les propriétés avec lesquelles remplir le clip dupliqué. Ce paramètre permet aux clips créés dynamiquement de recevoir des paramètres. Si `objetInit` n'est pas un objet, il est ignoré. Toutes les propriétés de `objetInit` sont copiées dans la nouvelle occurrence. Les propriétés spécifiées avec `objetInit` peuvent être utilisées par la fonction constructeur. Ce paramètre est facultatif.

Renvoie

Une référence au clip dupliqué.

Description

Méthode : crée une occurrence du clip spécifié pendant la lecture du fichier SWF. Les clips dupliqués commencent toujours à partir de l'image 1, quelle que soit l'image sur laquelle se trouve l'animation originale lorsque la méthode `duplicateMovieClip()` est appelée. Les variables du clip parent ne sont pas copiées dans le clip dupliqué. Les clips qui ont été créés avec la méthode `duplicateMovieClip()` ne sont pas dupliqués si vous appelez la méthode `duplicateMovieClip()` pour leur parent. Si le clip parent est effacé, le clip dupliqué l'est également.

Consultez également

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

MovieClip.enabled

Disponibilité

Flash Player 6.

Usage

mon_mc.enabled

Description

Propriété : valeur booléenne indiquant si un clip de bouton est activé. La valeur par défaut de enabled est true. Si enabled est défini sur la valeur false, les méthodes de rappel et les gestionnaires d'événement d'action *on* du clip de bouton ne sont plus invoqués et les images Dessus, Bas et Haut sont désactivées. La propriété enabled n'affecte pas le scénario du clip de bouton ; si un clip est en cours de lecture, cette lecture se poursuit. Le clip continue à recevoir des événements de clip (par exemple, mouseDown, mouseUp, keyDown et keyUp).

La propriété enabled régit uniquement les propriétés de type bouton d'un clip de bouton. Vous pouvez changer la propriété enabled à tout moment, le clip de bouton modifié étant immédiatement activé ou désactivé. La propriété enabled peut être lue à partir d'un objet prototype. Si enabled est défini sur false, l'objet n'est pas inclus dans l'ordre de tabulation automatique.

MovieClip.endFill()

Disponibilité

Flash Player 6.

Usage

mon_mc.endFill()

Paramètres

Aucun.

Renvoie

Rien.

Description

Méthode : applique un remplissage aux lignes et aux courbes ajoutées depuis le dernier appel de la méthode 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 qu'un remplissage est défini, le trajet est fermé avec un trait, puis rempli.

MovieClip.focusEnabled

Disponibilité

Flash Player 6.

Usage

mon_mc.focusEnabled

Description

Propriété : si la valeur est `undefined` ou `false`, un clip ne peut pas recevoir le focus de saisie, à moins qu'il ne s'agisse d'un clip de bouton. Si la propriété `focusEnabled` a pour valeur `true`, un clip peut recevoir le focus de saisie, même s'il ne s'agit pas d'un clip de bouton.

MovieClip._focusrect

Disponibilité

Flash Player 6.

Usage

mon_mc._focusrect

Description

Propriété : une valeur booléenne spécifiant si un rectangle jaune apparaît autour du clip avec focus clavier. Cette propriété peut annuler la propriété `_focusrect` globale.

MovieClip._framesloaded

Disponibilité

Flash Player 4.

Usage

mon_mc._framesloaded

Description

Propriété (lecture seule) : le nombre d'images chargées depuis un fichier SWF lu en flux continu. Cette propriété est utile pour déterminer si le contenu d'une image spécifique, et de toutes les images avant elle, a été chargé et est disponible localement dans le navigateur. Cette propriété est très utile pour contrôler le processus de téléchargement des fichiers SWF volumineux. Par exemple, il peut s'avérer utile d'afficher un message aux utilisateurs leur indiquant que le fichier SWF est en cours de chargement jusqu'à ce qu'une image spécifique contenue dans le fichier SWF soit chargée.

Exemple

L'exemple suivant utilise la propriété `_framesloaded` pour démarrer un fichier SWF lorsque toutes les images sont chargées. Si toutes les images ne sont pas chargées, la propriété `_xscale` de l'occurrence de clip `loader` est accrue proportionnellement pour créer une barre de progrès.

```
if (_framesloaded >= _totalframes) {
  gotoAndPlay ("Séquence 1", "start");
} else {
  _root.loader._xscale = (_framesloaded/_totalframes)*100;
}
```

Consultez également

Classe MovieClipLoader

MovieClip.getBounds

Disponibilité

Flash Player 5.

Usage

mon_mc.getBounds(*espaceDeCoordonnéesCible*)

Paramètres

espaceDeCoordonnéesCible Le chemin cible du scénario dont vous voulez utiliser le système de coordonnées comme point de référence.

Renvoie

Un objet avec les propriétés `xMin`, `xMax`, `yMin` et `yMax`.

Description

Méthode : renvoie des propriétés qui sont les valeurs minimum et maximum des coordonnées *x* et *y* de l'occurrence spécifiée par *mon_mc* pour le paramètre *espaceDeCoordonnéesCible*.

Remarque : Utilisez `MovieClip.localToGlobal()` et `MovieClip.globalToLocal()` pour convertir les coordonnées locales du clip en coordonnées de scène ou, à l'inverse, les coordonnées de scène en coordonnées locales.

Exemple

Dans l'exemple suivant, l'objet que renvoie la méthode `getBounds()` est affecté à l'identifiant `limitesDeClip`. Vous pouvez alors accéder aux valeurs de chaque propriété et les utiliser dans un script. Dans ce script, une autre occurrence de clip, `clip2`, est placée à côté de `clip`.

```
limitesDeClip = clip.getBounds(_root);
clip2._x = limitesDeClip.xMax;
```

Consultez également

`MovieClip.globalToLocal()`, `MovieClip.localToGlobal()`

MovieClip.getBytesLoaded()

Disponibilité

Flash Player 5.

Usage

mon_mc.getBytesLoaded()

Paramètres

Aucun.

Renvoie

Un entier indiquant le nombre d'octets chargés.

Description

Méthode : renvoie le nombre d'octets déjà chargé (transmis) pour le clip spécifié par *mon_mc*. Vous pouvez comparer cette valeur avec la valeur renvoyée par MovieClip.getBytesTotal() pour déterminer quel pourcentage de clip a été chargé.

Consultez également

MovieClip.getBytesTotal()

MovieClip.getBytesTotal()

Disponibilité

Flash Player 5.

Usage

mon_mc.getBytesTotal()

Paramètres

Aucun.

Renvoie

Un nombre entier indiquant la taille totale, en octets, de *mon_mc*.

Description

Méthode : renvoie la taille, en octets, du clip spécifié par *mon_mc*. Pour les clips externes (le fichier SWF racine ou un clip chargé dans une cible ou un niveau), la valeur renvoyée correspond à la taille du fichier SWF.

Consultez également

MovieClip.getBytesLoaded()

MovieClip.getDepth()

Disponibilité

Flash Player 6.

Usage

mon_mc.getDepth()

Paramètres

Aucun.

Renvoie

Un entier.

Description

Méthode : renvoie la profondeur d'une occurrence de clip. Pour plus d'informations, consultez *Gestion des profondeurs de clip*, page 135.

Consultez également

MovieClip.getInstanceAtDepth(), MovieClip.getNextHighestDepth(), MovieClip.swapDepths()

MovieClip.getInstanceAtDepth()

Disponibilité

Flash Player 7.

Usage

mon_mc.getInstanceAtDepth(*profondeur*)

Paramètres

profondeur Un nombre entier qui spécifie le niveau de profondeur à interroger.

Renvoie

Une référence à l'occurrence de MovieClip située à la profondeur spécifiée ou `undefined` si aucun clip ne se trouve à cette profondeur.

Description

Méthode : vous permet de déterminer si une profondeur particulière 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. Pour plus d'informations, consultez *Gestion des profondeurs de clip*, page 135.

Consultez également

MovieClip.getDepth(), MovieClip.getNextHighestDepth(), MovieClip.swapDepths()

MovieClip.getNextHighestDepth()

Disponibilité

Flash Player 7.

Usage

mon_mc.getNextHighestDepth()

Paramètres

Aucun.

Renvoie

Un entier représentant le prochain ordre de profondeur disponible qui apparaîtra au-dessus de tous les autres objets au même niveau et sur le même calque dans *mon_mc*.

Description

Méthode : vous permet de déterminer une valeur de profondeur qui vous pouvez transmettre à MovieClip.attachMovie(), MovieClip.duplicateMovieClip, ou MovieClip.createEmptyMovieClip() pour vérifier que le clip apparaît au-dessus de tous les autres objets au même niveau et sur le même calque dans le clip actuel. La valeur renvoyée est 0 ou plus (ainsi, les valeurs négatives ne sont pas renvoyées).

Pour plus d'informations, consultez *Gestion des profondeurs de clip*, page 135.

Consultez également

MovieClip.getDepth(), MovieClip.getInstanceAtDepth(), MovieClip.swapDepths()

MovieClip.getSWFVersion()

Disponibilité

Flash Player 7.

Usage

mon_mc.getSWFVersion()

Paramètres

Aucun.

Renvoie

Un entier qui spécifie que la version de Flash Player ciblée lors du chargement du fichier SWF dans *mon_mc* était publié.

Description

Méthode : renvoie un entier indiquant la version de Flash Player pour laquelle l'objet *mon_mc* a été publié. Si *mon_mc* est un fichier JPEG ou si une erreur se produit et que Flash ne peut pas déterminer la version SWF de *mon_mc* , -1 est renvoyé.

MovieClip.getTextSnapshot()

Disponibilité

Programmation : Flash MX 2004.

Lecture : fichiers SWF publiés pour Flash Player 6 ou ultérieur, lus dans Flash Player 7 ou ultérieur.

Usage

```
mon_mc.getTextSnapshot();
```

Paramètres

Aucun.

Renvoie

Un objet TextSnapshot contenant le texte statique de `mon_mc` ou une chaîne vide si `mon_mc` ne contient pas de texte statique.

Description

Méthode : renvoie un objet TextSnapshot contenant le texte de tous les champs de texte statiques dans le clip spécifié ; le texte de clips enfant n'est pas inséré.

Flash concatène le texte et le place dans l'objet TextSnapshot dans un ordre représentant l'ordre d'indexation des champs de texte statiques dans le clip. Les champs de texte ne comportant pas de valeurs d'indexation sont placés par ordre aléatoire dans l'objet et précèdent le texte de champs comportant des valeurs d'indexation. Aucun saut de ligne ou formatage n'indique la fin d'un champ et le début du suivant.

Remarque : Vous ne pouvez pas spécifier de valeur d'indexation pour du texte statique dans Flash. D'autres produits le permettent, Macromedia FlashPaper par exemple.

Le contenu de l'objet TextSnapshot n'est pas dynamique, c'est-à-dire que, si le clip se déplace sur une autre image ou s'il est modifié (si des objets du clip sont ajoutés ou supprimés, par exemple), l'objet TextSnapshot peut ne pas représenter le texte réel du clip. Pour s'assurer que le contenu de l'objet est le contenu réel, émettez de nouveau cette commande si nécessaire.

Consultez également

Objet TextSnapshot

MovieClip.getURL()

Disponibilité

Flash Player 5.

Usage

```
mon_mc.getURL(URL [,fenêtre, variables])
```

Paramètres

URL L'URL où se trouve le document à obtenir.

fenêtre Un paramètre facultatif spécifiant le nom, l'image ou l'expression qui définit la fenêtre ou le cadre HTML dans lequel le document est chargé. Vous pouvez également utiliser l'un des noms cible réservés suivants : `_self` spécifie l'image courante dans la fenêtre courante, `_blank` spécifie une nouvelle fenêtre, `_parent` spécifie le parent de l'image courante et `_top` spécifie l'image de premier niveau dans la fenêtre courante.

variables Un paramètre facultatif spécifiant une méthode d'envoi des variables associées au fichier SWF à charger. S'il n'y a pas de variables, omettez ce paramètre ; sinon, spécifiez si les variables doivent être chargées avec une méthode `GET` ou `POST`. `GET` ajoute les variables à la fin de l'URL et est utilisée pour un petit nombre de variables. `POST` envoie les variables dans un en-tête HTTP distinct et est utilisée pour les longues chaînes de variables.

Renvoie

Rien.

Description

Méthode : charge un document depuis une URL spécifiée dans la fenêtre spécifiée. La méthode `getURL` peut également être utilisée pour transmettre des variables à une autre application définie à l'URL avec une méthode `GET` ou `POST`.

Consultez également

`getURL()`

MovieClip.globalToLocal()

Disponibilité

Flash Player 5.

Usage

mon_mc.globalToLocal(*point*)

Paramètres

point Le nom ou l'identifiant d'un objet créé avec la Classe Object générique. L'objet définit les coordonnées *x* et *y* en tant que propriétés.

Renvoie

Rien.

Description

Méthode : convertit les coordonnées de scène (globales) de l'objet *point* en coordonnées de clip (locales).

Exemple

L'exemple suivant convertit les coordonnées *x* et *y* de l'objet *point* en coordonnées locales du clip.

```
onClipEvent(mouseMove) {
  point = new object();
  point.x = _root._xmouse;
  point.y = _root._ymouse;
  globalToLocal(point);
  trace(_root._xmouse + " " + _root._ymouse);
  trace(point.x + " " + point.y);
  updateAfterEvent();
}
```

Consultez également

MovieClip.getBounds, MovieClip.localToGlobal()

MovieClip.gotoAndPlay()

Disponibilité

Flash Player 5.

Usage

```
mon_mc.gotoAndPlay(image)
```

Paramètres

image Un nombre représentant le numéro de l'image ou une chaîne représentant l'étiquette de l'image vers laquelle la tête de lecture est envoyée.

Renvoie

Rien.

Description

Méthode : démarre la lecture du fichier SWF à partir de l'image spécifiée. Si vous souhaitez spécifier une scène ainsi qu'une image, utilisez `gotoAndPlay`.

MovieClip.gotoAndStop

Disponibilité

Flash Player 5.

Usage

```
mon_mc.gotoAndStop(image)
```

Paramètres

image Le numéro de l'image vers laquelle la tête de lecture est envoyée.

Renvoie

Rien.

Description

Méthode : envoie la tête de lecture à l'image spécifiée de ce clip et l'arrête.

Consultez également

`gotoAndStop()`

MovieClip._height

Disponibilité

Flash Player 4.

Usage

```
mon_mc._height
```

Description

Propriété : la hauteur du clip, en pixels.

Exemple

L'exemple de code suivant définit la hauteur et la largeur d'un clip lorsque l'utilisateur clique sur le bouton de la souris.

```
onClipEvent(mouseDown) {
  _width=200;
  _height=200;
}
```

MovieClip._highquality

Disponibilité

Flash Player 6.

Usage

```
mon_mc._highquality
```

Description

Propriété (globale) : spécifie le niveau d'anti-aliasing appliqué au fichier SWF en cours. Spécifiez 2 (qualité maximum) pour appliquer une qualité élevée avec le lissage bitmap toujours actif. Spécifiez 1 (qualité élevée) pour appliquer l'anti-aliasing ; cela permettra 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 supplanter la propriété `_highquality` globale.

Exemple

```
mon_mc._highquality = 2;
```

Consultez également

`_quality`

MovieClip.hitArea

Disponibilité

Flash Player 6.

Usage

mon_mc.hitArea

Renvoie

Une référence à un clip.

Description

Propriété : désigne un autre clip qui servira de zone active pour un clip de bouton. Si la propriété `hitArea` n'existe pas ou est `null` ou `undefined`, le clip de bouton même est utilisé en tant que zone active. La valeur de la propriété `hitArea` peut être une référence à un objet de clip.

Vous pouvez changer la propriété `hitArea` à tout moment, le clip de bouton modifié prenant immédiatement le nouveau comportement. Le clip désigné comme zone active n'a pas besoin d'être visible, sa forme graphique, bien que non visible, étant soumise à un test d'accès. La propriété `hitArea` peut être lue à partir d'un objet prototype.

MovieClip.hitTest()

Disponibilité

Flash Player 5.

Usage

mon_mc.hitTest(*x, y, baliseDeForme*)

mon_mc.hitTest(*cible*)

Paramètres

x La coordonnée *x* de la zone réactive sur la scène.

y La coordonnée *y* de la zone réactive sur la scène.

Les coordonnées *x* et *y* sont définies dans l'espace de coordonnées global.

cible Le chemin cible de la zone réactive pouvant croiser ou chevaucher l'occurrence spécifiée par *mon_mc*. Le paramètre *cible* représente généralement un bouton ou un champ de saisie de texte.

baliseDeForme Valeur booléenne spécifiant s'il faut évaluer la forme entière de l'occurrence spécifiée (*true*) ou seulement le cadre de délimitation (*false*). Ce paramètre ne peut être spécifié que si la zone réactive est identifiée avec les paramètres des coordonnées *x* et *y*.

Renvoie

Une valeur booléenne `true` si *mon_mc* chevauche la zone réactive, `false` dans les autres cas.

Description

Méthode : évalue l'occurrence spécifiée par *mon_mc* pour voir si elle chevauche ou croise la zone réactive identifiée par *cible* ou les paramètres des coordonnées *x* et *y*.

Usage 1 : compare les coordonnées *x* et *y* avec la forme ou le cadre de délimitation de l'occurrence spécifiée, en fonction du paramètre *baliseDeForme*. Si *baliseDeForme* est défini sur `true`, seule la zone occupée actuellement par l'occurrence sur la scène est évaluée, et si *x* et *y* se recouvrent, une valeur `true` est renvoyée. Cela est utile pour déterminer si le clip se trouve dans une zone sensible spécifiée.

Usage 2 : évalue les cadres de délimitation de *cible* et de l'occurrence spécifiée et renvoie `true` s'ils se croisent ou se recouvrent en un point.

Exemple

L'exemple suivant utilise `hitTest()` avec les propriétés `x_mouse` et `y_mouse` pour déterminer si la souris se trouve sur le cadre de délimitation de la cible :

```
if (hitTest( _root._xmouse, _root._ymouse, false));
```

L'exemple suivant utilise `hitTest()` pour déterminer si le clip `balle` chevauche ou croise le clip `carré` :

```
if(_root.balle.hitTest(_root.carré)){
  trace("balle croise carré");
}
```

Consultez également

`MovieClip.getBounds`, `MovieClip.globalToLocal()`, `MovieClip.localToGlobal()`

MovieClip.lineStyle()

Disponibilité

Flash Player 6.

Usage

mon_mc`.lineStyle ([`*épaisseur*`[, `*rvb*`[, `*alpha*`]]])`

Paramètres

épaisseur Un entier qui indique l'épaisseur de la ligne en points, les valeurs valides allant de 0 à 255. Si aucun nombre n'est spécifié ou si le paramètre est `undefined`, aucune ligne n'est tracée. Si une valeur inférieure à 0 est transmise, Flash utilise 0. La valeur 0 indique une épaisseur très fine, l'épaisseur maximum étant 255. Si une valeur supérieure à 255 est transmise, l'interprète de Flash utilise 255.

rvb Une valeur chromatique hexadécimale (par exemple, le rouge correspond à 0xFF0000, le bleu correspond à 0x0000FF, etc.) de la ligne. Si aucune valeur n'est indiquée, Flash utilise 0x000000 (noir).

alpha Un entier indiquant la valeur alpha de la couleur du trait ; les valeurs valides sont comprises entre 0 et 100. Lorsqu'aucune valeur n'est indiquée, Flash utilise 100 (uni). Si la valeur est inférieure à 0, Flash utilise 0. Si la valeur est supérieure à 100, Flash utilise 100.

Renvoie

Rien.

Description

Méthode : spécifie un style de trait que Flash utilise pour les appels ultérieurs aux méthodes `lineTo()` et `curveTo()`, jusqu'à ce que vous appeliez `lineStyle()` avec des paramètres différents. Vous pouvez appeler la méthode `lineStyle()` au milieu d'une opération de dessin d'un chemin afin de spécifier différents styles pour différents segments de ligne au sein d'un chemin.

Remarque : Les appels de `clear` réinitialisent la méthode `lineStyle()` à `undefined`.

Exemple

Le code suivant trace un triangle avec un trait magenta de 5 points et aucun remplissage.

```
_root.createEmptyMovieClip( "triangle", 1 );
with ( _root.triangle )
{
lineStyle( 5, 0xff00ff, 100 );
moveTo( 200, 200 );
lineTo( 300,300 );
lineTo(100,300);
lineTo( 200, 200 );
}
```

Consultez également

`MovieClip.beginFill()`, `MovieClip.beginGradientFill()`, `MovieClip.clear()`, `MovieClip.curveTo()`, `MovieClip.lineTo`, `MovieClip.moveTo`

MovieClip.lineTo

Disponibilité

Flash Player 6.

Usage

`mon_mc``.lineTo(`*`x`*`, `*`y`*`)`

Paramètres

`x` Un entier indiquant la position horizontale du clip parent par rapport au point d'alignement.

`y` Un entier indiquant la position verticale du clip parent par rapport au point d'alignement.

Renvoie

Rien.

Description

Méthode : dessine une ligne avec le style de trait courant de la position actuelle à (*x*, *y*), la position de dessin actuelle étant alors définie à (*x*, *y*). Si le clip dans lequel vous dessinez contient un contenu créé avec les outils de dessin de Flash, les appels à `lineTo()` sont dessinés au-dessous de ce contenu. Si vous appelez la méthode `lineTo()` avant tout appel à la méthode `moveTo()`, la position actuelle du dessin est par défaut (`0`, `0`). Cette méthode échoue lorsque des paramètres sont absents et la position de dessin courante n'est pas changée.

Exemple

L'exemple suivant trace un triangle sans trait et avec un remplissage bleu partiellement transparent.

```
_root.createEmptyMovieClip ("triangle", 1);
  with (_root.triangle){
    beginFill (0x0000FF, 50);
    lineStyle (5, 0xFF00FF, 100);
    moveTo (200, 200);
    lineTo (300, 300);
    lineTo (100, 300);
    lineTo (200, 200);
    endFill();
}
```

Consultez également

`MovieClip.beginFill()`, `MovieClip.createEmptyMovieClip()`, `MovieClip.endFill()`, `MovieClip.lineStyle()`, `MovieClip.moveTo`

MovieClip.loadMovie

Disponibilité

Flash Player 5.

Usage

```
mon_mc.loadMovie("url" [,variables])
```

Paramètres

url L'URL absolue ou relative du fichier SWF ou JPEG à charger. Un chemin relatif doit être relatif au fichier SWF au niveau 0. Des URL absolues doivent inclure la référence au protocole, comme http:// ou file:///.

variables Un paramètre facultatif spécifiant une méthode HTTP d'envoi ou de chargement des variables. Le paramètre doit être la chaîne `GET` ou `POST`. Omettez ce paramètre si aucune variable ne doit être envoyée. La méthode `GET` ajoute les variables à la fin de l'URL et est utilisée pour des petits nombres de variables. La méthode `POST` envoie les variables dans un en-tête HTTP distinct et est utilisée pour les longues chaînes de variables.

Renvoie

Rien.

Description

Méthode : charge des fichiers SWF ou JPEG dans un clip dans Flash Player pendant la lecture du fichier SWF d'origine.

Conseil : Si vous souhaitez suivre la progression du téléchargement, utilisez `MovieClipLoader.loadClip()` au lieu de cette fonction.

Sans la méthode `loadMovie()`, Flash Player affiche un seul fichier SWF avant de quitter. La méthode `loadMovie()` vous permet d'afficher plusieurs fichiers SWF simultanément et de basculer entre les fichiers SWF sans charger un autre document HTML.

Un fichier SWF ou une image chargée dans un clip hérite des propriétés de position, rotation et échelle de ce clip. Vous pouvez utiliser le chemin cible du clip pour cibler le fichier SWF chargé.

Utilisez la méthode `unloadMovie()` pour retirer les fichiers ou images SWF chargés avec la méthode `loadMovie()`. Utilisez la méthode `loadVariables` pour garder le fichier SWF actif et mettre les variables à jour avec de nouvelles valeurs.

Consultez également

`loadMovie()`, `loadMovieNum()`, `MovieClip.loadVariables()`, `MovieClip.unloadMovie()`, `unloadMovie()`, `unloadMovieNum()`

MovieClip.loadVariables()

Disponibilité

Flash Player 5 ; comportement mis à jour dans Flash Player 7.

Usage

```
mon_mc.loadVariables("url", variables)
```

Paramètres

url L'URL absolue ou relative du fichier externe contenant les variables à charger. Si le fichier SWF résultant de cet appel est ouvert dans un navigateur web, l'*url* doit être du même domaine que le fichier SWF ; pour plus de détails, consultez la description ci-dessous.

variables Un paramètre facultatif spécifiant une méthode HTTP d'envoi des variables. Le paramètre doit être la chaîne `GET` ou `POST`. Omettez ce paramètre si aucune variable ne doit être envoyée. La méthode `GET` ajoute les variables à la fin de l'URL et est utilisée pour des petits nombres de variables. La méthode `POST` envoie les variables dans un en-tête HTTP distinct et est utilisée pour les longues chaînes de variables.

Renvoie

Rien.

Description

Méthode : lit les données depuis un fichier externe et définit les valeurs pour les variables de *mon_mc*. Le fichier externe peut être un fichier texte généré par un script CGI, Active Server Page (ASP) ou un script PHP, et peut contenir n'importe quel nombre de variables.

Cette méthode peut également être utilisée pour mettre à jour des variables dans le clip actif avec de nouvelles valeurs.

Cette méthode nécessite que le texte de l'URL soit au format standard MIME : *application/x-www-form-urlencoded* (format script CGI).

Dans des fichiers SWF exécutés dans une version du lecteur antérieure à Flash Player 7, l'*url* doit être du même super-domaine que le fichier SWF résultant de cet appel. Par exemple, un fichier SWF à l'adresse www.Domaine.com peut charger des variables d'un fichier SWF à l'adresse store.Domaine.com car les deux fichiers sont du même super-domaine de Domaine.com.

Dans des fichiers SWF d'une quelconque version exécutée dans Flash Player 7 ou ultérieur, l'*url* doit être exactement du même domaine (consultez *Fonctions de sécurité de Flash Player*, page 199). Par exemple, un fichier SWF à l'adresse www.Domaine.com peut uniquement charger des variables de fichiers SWF également à l'adresse www.Domaine.com. Si vous souhaitez charger des variables d'un autre domaine, vous pouvez placer un *fichier de régulation inter-domaines* sur le serveur hébergeant le fichier SWF auquel vous accédez actuellement. Pour plus d'informations, consultez *A propos de l'autorisation de chargement de données inter-domaines*, page 201.

Consultez également

`loadMovie()`, `loadVariables()`, `loadvariablesNum()`, `MovieClip.unloadMovie()`

MovieClip.localToGlobal()

Disponibilité

Flash Player 5.

Usage

mon_mc.localToGlobal(*point*)

Paramètres

point Le nom ou l'identifiant d'un objet créé avec la Classe Object, spécifiant les coordonnées *x* et *y* comme propriétés.

Renvoie

Rien.

Description

Méthode : convertit les coordonnées du clip (locales) de l'objet *point* en coordonnées de scène (globales).

Exemple

L'exemple suivant convertit les coordonnées *x* et *y* de l'objet *point*, des coordonnées du clip (locales) en coordonnées de scène (globales). Les coordonnées locales *x* et *y* sont spécifiées à l'aide des propriétés _xmouse et _ymouse pour récupérer les coordonnées *x* et *y* de la position du pointeur de la souris.

```
onClipEvent(mouseMove) {
  point = new object();
  point.x = _xmouse;
  point.y = _ymouse;
  _root.out3 = point.x + " === " + point.y;
  _root.out = _root._xmouse + " === " + _root._ymouse;
  localToGlobal(point);
  _root.out2 = point.x + " === " + point.y;
  updateAfterEvent();
}
```

Consultez également

MovieClip.globalToLocal()

MovieClip._lockroot

Disponibilité

Flash Player 7.

Usage

mon_mc._lockroot

Description

Propriété : spécifie la référence de _root lors du chargement d'un fichier SWF dans un clip. La propriété _lockroot est undefined par défaut. Vous pouvez définir cette propriété dans le fichier SWF en cours de chargement ou dans le gestionnaire qui charge le clip.

Par exemple, supposons que vous possédiez un fichier nommé Jeux.fla qui permet à un utilisateur de choisir un jeu et de le charger (par exemple, Echecs.swf) dans le clip jeux_mc. Vous souhaitez être sûr que, si _root est utilisé dans Echecs.swf, il se rapporte toujours à _root dans Echecs.swf après avoir été chargé dans Jeux.swf. Si vous avez accès à Echecs.fla et pouvez le publier dans Flash Player 7 ou ultérieur, vous pouvez y ajouter cette instruction :

```
this._lockroot = true;
```

Si vous n'avez pas accès à Echecs.fla (par exemple, si vous chargez Echecs.swf depuis le site de quelqu'un d'autre), vous pouvez définir ses propriétés _lockroot lors de son chargement, comme indiqué ci-dessous. Dans ce cas, Echecs.swf peut être publié avec n'importe quelle version de Flash Player, tant que Jeux.swf est publié pour FlashPlayer 7 ou ultérieur.

```
onClipEvent (load)
{
  this._lockroot = true;
}
jeu_mc.loadMovie ("Echecs.swf");
```

Si vous n'utilisez pas l'instruction this._lockroot = true dans aucun des deux fichiers SWF, _root dans Echecs.swf fait référence à _root dans Jeux.swf une fois que Echecs.swf est chargé dans Jeux.swf.

Consultez également

_root, MovieClip.attachMovie(), MovieClip.loadMovie

MovieClip.menu

Disponibilité

Flash Player 7.

Utilisation

mon_mc.menu = *contextMenu*

Paramètres

menuContextuel Un objet ContextMenu.

Description

Propriété : associe l'objet ContextMenu spécifié au clip *mon_mc*. La classe ContextMenu vous permet de modifier le menu contextuel qui apparaît quand l'utilisateur clique avec le bouton droit de la souris (Windows) ou enfonce la touche Contrôle (Macintosh) dans Flash Player.

Exemple

L'exemple suivant affecte l'objet ContextMenu `menu_cm` au clip `content_mc`. L'objet ContextMenu contient une option de menu personnalisée intitulée « Imprimer... » dont le gestionnaire de rappel associé est nommé `doPrint()`.

```
var menu_cm = new ContextMenu();
menu_cm.customItems.push(nouveau ContextMenuItem("Imprimer...", doPrint));
function doPrint(menu, obj) {
  // Code "Imprimer" ici
}
content_mc.menu = menu_cm;
```

Voir aussi

Button.menu, Classe ContextMenu, Classe ContextMenuItem, TextField.menu

MovieClip.moveTo

Disponibilité

Flash Player 6.

Usage

mon_mc.moveTo(*x*, *y*)

Paramètres

x Un entier indiquant la position horizontale du clip parent par rapport au point d'alignement.

y Un entier indiquant la position verticale du clip parent par rapport au point d'alignement.

Renvoie

Rien.

Description

Méthode : place la position de dessin actuelle à (*x*, *y*). Cette méthode échoue lorsque des paramètres sont absents et la position de dessin courante n'est pas changée.

Exemple

Cet exemple trace un triangle avec des lignes magenta de 5 points continues, sans remplissage. La première ligne crée un clip vide pour commencer le dessin. Un type de ligne est défini à l'intérieur de l'instruction `with`, puis la position de dessin de début est indiquée par la méthode `moveTo()`.

```
_root.createEmptyMovieClip( "triangle", 1 );
with ( _root.triangle )
{
lineStyle( 5, 0xff00ff, 100 );
moveTo( 200, 200 );
lineTo( 300,300 );
lineTo(100,300);
lineTo( 200, 200 );
}
```

Consultez également

`MovieClip.createEmptyMovieClip()`, `MovieClip.lineStyle()`, `MovieClip.lineTo`

MovieClip._name

Disponibilité

Flash Player 4.

Usage

```
mon_mc._name
```

Description

Propriété : le nom d'occurrence du clip spécifié par *mon_mc*.

MovieClip.nextFrame()

Disponibilité

Flash Player 5.

Usage

```
mon_mc.nextFrame()
```

Paramètres

Aucun.

Renvoie

Rien.

Description

Méthode : envoie la tête de lecture à l'image suivante et l'arrête.

Consultez également

`nextFrame()`

MovieClip.onData

Disponibilité

Flash Player 6.

Usage

```
mon_mc.onData = function() {
  // vos instructions
}
```

Paramètres

Aucun.

Renvoie

Rien.

Description

Gestionnaire d'événement : invoqué lorsqu'un clip reçoit des données d'un appel `loadVariables()` ou `loadMovie()`. Vous devez définir une fonction exécutée lorsque le gestionnaire d'événement est invoqué.

Ce gestionnaire ne peut être utilisé qu'avec les clips dont vous possédez un symbole associé à une classe dans la bibliothèque. Si vous souhaitez qu'un gestionnaire d'événement soit appelé lorsqu'un clip spécifique reçoit des données, vous devez utiliser `onClipEvent(data)` à la place de ce gestionnaire. Ce dernier est invoqué lorsqu'un clip reçoit des données.

Exemple

L'exemple suivant montre comment utiliser correctement `MovieClip.onData()` et `onClipEvent(data)`.

```
// symbol_mc est un symbole de clip dans la bibliothèque.
// Il est associé à la classe MovieClip.
// La fonction suivante est déclenchée pour chaque occurrence de symbol_mc
// lors de la réception de données.
symbol_mc.onData = function() {
   trace("Le clip a reçu des données");
}
// dynamic_mc est un clip chargé avec MovieClip.loadMovie().
// Ce code tente d'appeler une fonction lorsque le clip est chargé,
// mais sans succès, car le fichier SWF chargé n'est pas un symbole contenu
// dans la bibliothèque et associé à la classe MovieClip.
function output()
{
    trace("Ne sera pas appelé.");
}
dynamic_mc.onData = output;
dynamic_mc.loadMovie("remplacement.swf");
// La fonction suivante est appelée pour tout clip
// recevant des données, qu'il soit dans la bibliothèque ou non.
// Cette fonction est donc appelée lorsque symbol_mc est instancié
// et lorsque remplacement.swf est chargé.
OnClipEvent( data ) {
   trace("Le clip a reçu des données");
}
```

Consultez également

`onClipEvent()`

MovieClip.onDragOut

Disponibilité

Flash Player 6.

Usage

```
mon_mc.onDragOut = function() {
  // vos instructions
}
```

Paramètres

Aucun.

Renvoie

Rien.

Description

Gestionnaire d'événement : invoqué lorsque vous cliquez sur le bouton de la souris et que le pointeur sort de l'objet. Vous devez définir une fonction exécutée lorsque le gestionnaire d'événement est invoqué.

Exemple

L'exemple suivant définit une fonction pour la méthode `onDragOut` qui envoie une action `trace()` au panneau de sortie.

```
mon_mc.onDragOut = function () {
  trace ("onDragOut appelé");
};
```

Consultez également

`MovieClip.onDragOver`

MovieClip.onDragOver

Disponibilité

Flash Player 6.

Usage

```
mon_mc.onDragOver = function() {
  // vos instructions
}
```

Paramètres

Aucun.

Renvoie

Rien.

Description

Gestionnaire d'événement : invoqué lorsque le pointeur est sorti avant d'être repositionné au-dessus du clip. Vous devez définir une fonction exécutée lorsque le gestionnaire d'événement est invoqué.

Exemple

L'exemple suivant définit une fonction pour la méthode `onDragOver` qui envoie une action `trace()` au panneau de sortie.

```
mon_mc.onDragOver = function () {
  trace ("onDragOver appelé");
};
```

Consultez également

`MovieClip.onDragOut`

MovieClip.onEnterFrame

Disponibilité

Flash Player 6.

Usage

```
mon_mc.onEnterFrame = function() {
  // vos instructions
}
```

Paramètres

Aucun.

Renvoie

Rien.

Description

Gestionnaire d'événement : invoqué continuellement à la cadence du fichier SWF. Les actions associées à l'événement de clip `enterFrame` sont traitées avant les actions associées aux images affectées.

Vous devez définir une fonction exécutée lorsque le gestionnaire d'événement est invoqué.

Exemple

L'exemple suivant définit une fonction pour la méthode `onEnterFrame` qui envoie une action `trace()` au panneau de sortie.

```
mon_mc.onEnterFrame = function () {
  trace ("onEnterFrame appelé");
};
```

MovieClip.onKeyDown

Disponibilité

Flash Player 6.

Usage

```
mon_mc.onKeyDown = function() {
  // vos instructions
}
```

Paramètres

Aucun.

Renvoie

Rien.

Description

Gestionnaire d'événement : invoqué lorsqu'un clip a le focus de saisie et qu'une touche est enfoncée. Le gestionnaire d'événement `onKeyDown` est invoqué sans paramètres. Vous pouvez utiliser les méthodes `Key.getAscii()` et `Key.getCode()` pour déterminer quelle touche a été enfoncée. Vous devez définir une fonction exécutée lorsque le gestionnaire d'événement est invoqué.

Le gestionnaire d'événement `onKeyDown` ne fonctionne que si le focus de saisie du clip est activé et défini. D'abord, la propriété `focusEnabled` du clip doit être définie sur la valeur `true`. Ensuite, le clip doit disposer d'un focus. Pour cela, utilisez `Selection.setFocus` ou définissez la touche tabulation pour naviguer vers le clip.

Si vous utilisez `Selection.setFocus`, le chemin du clip doit être affecté à `Selection.setFocus`. D'autres éléments peuvent très facilement reprendre le focus une fois que la souris est déplacée.

Exemple

L'exemple suivant définit une fonction pour la méthode `onKeyDown()` qui envoie une action `trace()` au panneau de sortie.

```
mon_mc.onKeyDown = function () {
  trace ("onKeyDown appelé");
};
```

L'exemple suivant définit le focus de saisie.

```
MovieClip.focusEnabled = true;
Selection.setFocus(MovieClip);
```

Consultez également

`MovieClip.onKeyUp`

MovieClip.onKeyUp

Disponibilité

Flash Player 6.

Usage

```
mon_mc.onKeyUp = function() {
  // vos instructions
}
```

Paramètres

Aucun.

Renvoie

Rien.

Description

Gestionnaire d'événement : invoqué lorsqu'une touche est relâchée. Le gestionnaire d'événement `onKeyUp` est invoqué sans paramètres. Vous pouvez utiliser les méthodes `Key.getAscii()` et `Key.getCode()` pour déterminer quelle touche a été enfoncée. Vous devez définir une fonction exécutée lorsque le gestionnaire d'événement est invoqué.

Le gestionnaire d'événement `onKeyUp` ne fonctionne que si le focus de saisie du clip est activé et défini. D'abord, la propriété `focusEnabled` du clip doit être définie sur la valeur `true`. Ensuite, le clip doit disposer d'un focus. Pour cela, utilisez `Selection.setFocus` ou définissez la touche tabulation pour naviguer vers le clip.

Si vous utilisez `Selection.setFocus`, le chemin du clip doit être affecté à `Selection.setFocus`. D'autres éléments peuvent très facilement reprendre le focus une fois que la souris est déplacée.

Exemple

L'exemple suivant définit une fonction pour la méthode `onKeyUp` qui envoie une action `trace()` au panneau de sortie.

```
mon_mc.onKeyUp = function () {
  trace ("onKeyUp appelé");
};
```

L'exemple suivant définit le focus de saisie :

```
MovieClip.focusEnabled = true;
Selection.setFocus(MovieClip);
```

MovieClip.onKillFocus

Disponibilité

Flash Player 6.

Usage

```
mon_mc.onKillFocus = function (nouveauFocus) {
  // vos instructions
}
```

Paramètres

nouveauFocus L'objet recevant le focus clavier.

Renvoie

Rien.

Description

Gestionnaire d'événement : invoqué lorsqu'un clip perd le focus clavier. La méthode `onKillFocus` reçoit un paramètre, *nouveauFocus*, qui est un objet représentant le nouvel objet recevant le focus. Si aucun objet ne reçoit le focus, *nouveauFocus* contient la valeur `null`.

MovieClip.onLoad

Disponibilité

Flash Player 6.

Usage

```
mon_mc.onLoad = function() {
  // vos instructions
}
```

Paramètres

Aucun.

Renvoie

Rien.

Description

Gestionnaire d'événement : invoqué lorsque le clip est instancié et apparaît dans le scénario. Vous devez définir une fonction exécutée lorsque le gestionnaire d'événement est invoqué.

Ce gestionnaire ne peut être utilisé qu'avec les clips dont vous possédez un symbole associé à une classe dans la bibliothèque. Si vous souhaitez qu'un gestionnaire d'événement soit invoqué lorsqu'un clip spécifique est chargé, par exemple lorsque vous utilisez `MovieClip.loadMovie()` pour charger dynamiquement un fichier SWF, vous devez utiliser `onClipEvent(load)` à la place de ce gestionnaire. Ce dernier est invoqué lorsqu'un clip est chargé.

Exemple

L'exemple suivant montre comment utiliser correctement `MovieClip.onLoad()` et `onClipEvent(load)`.

```
// symbol_mc est un symbole de clip dans la bibliothèque.
// Il est associé à la classe MovieClip.
// La fonction suivante est déclenchée pour chaque occurrence de symbol_mc
// lorsqu'il est instancié et apparaît dans le scénario.
symbol_mc.onLoad = function() {
   trace("Le clip est chargé");
}
// dynamic_mc est un clip chargé avec MovieClip.loadMovie().
// Ce code tente d'appeler une fonction lorsque le clip est chargé,
// mais sans succès, car le fichier SWF chargé n'est pas un symbole contenu
// dans la bibliothèque et associé à la classe MovieClip.
function output()
{
    trace("Ne sera pas appelé.");
}
dynamic_mc.onLoad = output;
dynamic_mc.loadMovie("remplacement.swf");
// La fonction suivante est appelée pour tout clip
// apparaît dans le scénario, qu'il soit dans la bibliothèque ou non.
// Cette fonction est donc appelée lorsque symbol_mc est instancié
// et lorsque remplacement.swf est chargé.
OnClipEvent( load ) {
   trace("Le clip est chargé");
}
```

Consultez également

`onClipEvent()`

MovieClip.onMouseDown

Disponibilité

Flash Player 6.

Usage

```
mon_mc.onMouseDown = function() {
  // vos instructions
}
```

Paramètres

Aucun.

Renvoie

Rien.

Description

Gestionnaire d'événement : invoqué lorsque le bouton de la souris est enfoncé. Vous devez définir une fonction exécutée lorsque le gestionnaire d'événement est invoqué.

Exemple

L'exemple suivant définit une fonction pour la méthode `onMouseDown` qui envoie une action `trace()` au panneau de sortie.

```
mon_mc.onMouseDown = function () {
  trace ("onMouseDown appelé");
}
```

MovieClip.onMouseMove

Disponibilité

Flash Player 6.

Usage

```
mon_mc.onMouseMove = function() {
  // vos instructions
}
```

Paramètres

Aucun.

Renvoie

Rien.

Description

Gestionnaire d'événement : invoqué lorsque la souris se déplace. Vous devez définir une fonction exécutée lorsque le gestionnaire d'événement est invoqué.

Exemple

L'exemple suivant définit une fonction pour la méthode `onMouseMove` qui envoie une action `trace()` au panneau de sortie.

```
mon_mc.onMouseMove = function () {
  trace ("onMouseMove appelé");
};
```

MovieClip.onMouseUp

Disponibilité

Flash Player 6.

Usage

```
mon_mc.onMouseUp = function() {
  // vos instructions
}
```

Paramètres

Aucun.

Renvoie

Rien.

Description

Gestionnaire d'événement : invoqué lorsque le bouton de la souris est relâché. Vous devez définir une fonction exécutée lorsque le gestionnaire d'événement est invoqué.

Exemple

L'exemple suivant définit une fonction pour la méthode `onMouseUp` qui envoie une action `trace()` au panneau de sortie.

```
mon_mc.onMouseUp = function () {
  trace ("onMouseUp appelé");
};
```

MovieClip.onPress

Disponibilité

Flash Player 6.

Usage

```
mon_mc.onPress = function() {
  // vos instructions
}
```

Paramètres

Aucun.

Renvoie

Rien.

Description

Gestionnaire d'événement : invoqué lorsque l'utilisateur clique sur la souris alors que le pointeur est sur un clip. Vous devez définir une fonction exécutée lorsque le gestionnaire d'événement est invoqué.

Exemple

L'exemple suivant définit une fonction pour la méthode `onPress` qui envoie une action `trace()` au panneau de sortie.

```
mon_mc.onPress = function () {
  trace ("onPress appelé");
};
```

MovieClip.onRelease

Disponibilité

Flash Player 6.

Usage

```
mon_mc.onRelease = function() {
  // vos instructions
}
```

Paramètres

Aucun.

Renvoie

Rien.

Description

Gestionnaire d'événement : invoqué lorsqu'un clip de bouton est relâché. Vous devez définir une fonction exécutée lorsque le gestionnaire d'événement est invoqué.

Exemple

L'exemple suivant définit une fonction pour la méthode `onPress` qui envoie une action `trace()` au panneau de sortie.

```
mon_mc.onRelease = function () {
  trace ("onRelease appelé");
};
```

MovieClip.onReleaseOutside

Disponibilité

Flash Player 6.

Usage

```
mon_mc.onReleaseOutside = function() {
  // vos instructions
}
```

Paramètres

Aucun.

Renvoie

Rien.

Description

Gestionnaire d'événement : invoqué lorsque la souris est relâchée pendant que le pointeur se trouve au dehors du clip après l'enfoncement du bouton pendant que le pointeur est à l'intérieur du clip.

Vous devez définir une fonction exécutée lorsque le gestionnaire d'événement est invoqué.

Exemple

L'exemple suivant définit une fonction pour la méthode `onReleaseOutside` qui envoie une action `trace()` au panneau de sortie.

```
mon_mc.onReleaseOutside = function () {
  trace ("onReleaseOutside appelé");
};
```

MovieClip.onRollOut

Disponibilité

Flash Player 6.

Usage

```
mon_mc.onRollOut = function() {
  // vos instructions
}
```

Paramètres

Aucun.

Renvoie

Rien.

Description

Gestionnaire d'événement : invoqué lorsque le pointeur passe à l'extérieur de la zone d'un clip. Vous devez définir une fonction exécutée lorsque le gestionnaire d'événement est invoqué.

Exemple

L'exemple suivant définit une fonction pour la méthode `onRollOut` qui envoie une action `trace()` au panneau de sortie.

```
mon_mc.onRollOut = function () {
  trace ("onRollOut appelé");
};
```

MovieClip.onRollOver

Disponibilité

Flash Player 6.

Usage

```
mon_mc.onRollOver = function() {
  // vos instructions
}
```

Paramètres

Aucun.

Renvoie

Rien.

Description

Gestionnaire d'événement : invoqué lorsque le pointeur passe au-dessus de la zone d'un clip. Vous devez définir une fonction exécutée lorsque le gestionnaire d'événement est invoqué.

Exemple

L'exemple suivant définit une fonction pour la méthode `onRollOver` qui envoie une action `trace()` au panneau de sortie.

```
mon_mc.onRollOver = function () {
  trace ("onRollOver appelé");
};
```

MovieClip.onSetFocus

Disponibilité

Flash Player 6.

Usage

```
mon_mc.onSetFocus = function(ancienFocus){
  // vos instructions
}
```

Paramètres

`ancienFocus` L'objet devant perdre le focus.

Renvoie

Rien.

Description

Gestionnaire d'événement : invoqué lorsqu'un clip reçoit le focus clavier. Le paramètre *`ancienFocus`* est l'objet perdant le focus. Par exemple, si l'utilisateur appuie sur la touche Tab pour faire passer le focus de saisie d'un clip à un champ de texte, *`ancienFocus`* contient l'occurrence de clip.

Si aucun objet n'a précédemment reçu le focus, *`ancienFocus`* contient une valeur `null`.

MovieClip.onUnload

Disponibilité

Flash Player 6.

Usage

```
mon_mc.onUnload = function() {
  // vos instructions
}
```

Paramètres

Aucun.

Renvoie

Rien.

Description

Gestionnaire d'événement : invoqué dans la première image après la suppression du clip du scénario. Flash traite les actions associées au gestionnaire d'événement `onUnload` avant d'associer toute action à l'image affectée. Vous devez définir une fonction exécutée lorsque le gestionnaire d'événement est invoqué.

Exemple

L'exemple suivant définit une fonction pour la méthode `MovieClip.onLoad` qui envoie une action `trace()` au panneau de sortie.

```
mon_mc.onUnload = function () {
  trace ("onUnload appelé");
};
```

MovieClip._parent

Disponibilité

Flash Player 5.

Usage

mon_mc._parent.*propriété*

_parent.*propriété*

Description

Propriété : une référence au clip ou à l'objet contenant le clip ou objet courant. L'objet courant est l'objet contenant le code ActionScript faisant référence à _parent. Utilisez la propriété _parent pour spécifier un chemin relatif aux clips ou objets qui se trouvent au-dessus du clip ou objet actuel.

Vous pouvez utiliser _parent pour monter de plusieurs niveaux dans la liste d'affichage, comme dans l'exemple suivant :

```
_parent._parent._alpha = 20;
```

Consultez également

Button._parent, _root, targetPath, TextField._parent

MovieClip.play()

Disponibilité

Flash Player 5.

Usage

mon_mc.play()

Paramètres

Aucun.

Renvoie

Rien.

Description

Méthode : déplace la tête de lecture dans le clip.

Consultez également

play()

MovieClip.prevFrame()

Disponibilité

Flash Player 5.

Usage

```
mon_mc.prevFrame()
```

Paramètres

Aucun.

Renvoie

Rien.

Description

Méthode : envoie la tête de lecture à l'image précédente et l'arrête.

Consultez également

`prevFrame()`

MovieClip.removeMovieClip()

Disponibilité

Flash Player 5.

Usage

```
mon_mc.removeMovieClip()
```

Paramètres

Aucun.

Renvoie

Rien.

Description

Méthode : renvoie une occurrence de clip créée avec `duplicateMovieClip()`, `MovieClip.duplicateMovieClip`, ou `MovieClip.attachMovie()`.

MovieClip._rotation

Disponibilité

Flash Player 4.

Usage

```
mon_mc._rotation
```

Description

Propriété : la rotation du clip, en degrés, à partir de son orientation d'origine. Les valeurs de 0 à 180 représentent une rotation dans le sens horaire ; les valeurs de 0 à -180 représentent une rotation dans le sens antihoraire. Les valeurs en dehors de cette plage sont ajoutées à ou soustraites de 360 pour obtenir une valeur comprise dans la plage. Par exemple, l'instruction `mon_mc._rotation = 450` est identique à l'instruction `mon_mc._rotation = 90`.

Consultez également

`Button._rotation`, `TextField._rotation`

MovieClip.setMask()

Disponibilité

Flash Player 6.

Usage

mon_mc.setMask(*masque_mc*)

Paramètres

mon_mc Le nom d'occurrence d'un clip à masquer.

masque_mc Le nom d'occurrence d'un clip devant être un masque.

Renvoie

Rien.

Description

Méthode : transforme le clip du paramètre *masque_mc* en masque révélant le clip spécifié par le paramètre *mon_mc*.

Cette méthode permet à des clips multi-images, avec un contenu complexe et multi-calques, d'agir en tant que masques. Vous pouvez activer et désactiver les masques à l'exécution. Cependant, vous ne pouvez pas utiliser le même masque pour masquer plusieurs objets (ce qui est possible en utilisant des calques de masque). Les polices de périphérique présentes dans un clip masqué sont tracées mais pas masquées. Vous ne pouvez pas définir un clip comme étant son propre masque, par exemple, `mon_mc.setMask(mon_mc)`.

Si vous créez un calque de masque qui contient un clip et que vous lui appliquez la méthode `setMask()`, l'appel `setMask()` est prioritaire et son effet est irréversible. Par exemple, vous pourriez avoir un clip dans un calque de masque appelé `MasqueUI` qui masque un autre calque contenant un autre clip appelé `MasqueurUI`. Si, à la lecture du fichier SWF, vous appelez `MasqueUI.setMask(MasqueurUI)`, à partir de ce moment-là, `MasqueUI` est masqué par `MasqueurUI`.

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.

```
MasqueUI.setMask(null);
```

Exemple

L'exemple de code suivant utilise le clip `masqueCercle_mc` pour masquer le clip `leMasqueur_mc`.

```
leMasqueur_mc.setMask(masqueCercle_mc);
```

MovieClip._soundbuftime

Disponibilité

Flash Player 6.

Usage

```
mon_mc._soundbuftime
```

Description

Propriété (globale) : un entier spécifiant le nombre de secondes de mise en tampon d'un son avant sa lecture en flux continu.

MovieClip.startDrag()

Disponibilité

Flash Player 5.

Usage

```
mon_mc.startDrag([verrouiller, [gauche, haut, droite, bas]])
```

Paramètres

verrouiller Une valeur booléenne spécifiant si le clip déplaçable est verrouillé au centre de la position de la souris (`true`) ou verrouillé sur le point auquel l'utilisateur a cliqué sur le clip (`false`). Ce paramètre est facultatif.

gauche, *haut*, *droite*, *bas* Valeurs relatives aux coordonnées du parent du clip spécifiant un rectangle de contrainte pour le clip. Ces paramètres sont facultatifs.

Renvoie

Rien.

Description

Méthode : permet à l'utilisateur de faire glisser le clip spécifié. Le clip reste déplaçable jusqu'à ce qu'il soit explicitement arrêté par un appel de `MovieClip.stopDrag()` ou qu'un autre clip devienne déplaçable. On ne peut faire glisser qu'un seul clip à la fois.

Consultez également

`MovieClip._droptarget`, `startDrag()`, `MovieClip.stopDrag()`

MovieClip.stop()

Disponibilité

Flash Player 5.

Usage

mon_mc.stop()

Paramètres

Aucun.

Renvoie

Rien.

Description

Méthode : arrête la lecture du clip courant.

Consultez également

stop()

MovieClip.stopDrag()

Disponibilité

Flash Player 5.

Usage

mon_mc.stopDrag()

Paramètres

Aucun.

Renvoie

Rien.

Description

Méthode : termine une méthode MovieClip.startDrag() Une animation rendue déplaçable avec cette méthode reste déplaçable jusqu'à l'ajout d'une méthode stopDrag() ou jusqu'à ce qu'une autre animation devienne déplaçable. On ne peut faire glisser qu'un seul clip à la fois.

Consultez également

MovieClip._droptarget, MovieClip.startDrag(), stopDrag()

MovieClip.swapDepths()

Disponibilité

Flash Player 5.

Usage

```
mon_mc.swapDepths(profondeur)
mon_mc.swapDepths(cible)
```

Paramètres

profondeur Un nombre spécifiant le niveau de profondeur auquel *mon_mc* doit être placé.

cible Une chaîne spécifiant l'occurrence de clip dont la profondeur est permutée par l'occurrence spécifiée par *mon_mc*. Les deux occurrences doivent avoir le même clip parent.

Renvoie

Rien.

Description

Méthode : permute l'ordre d'empilement, ou ordre *z* (niveau de profondeur), de l'occurrence spécifiée (*mon_mc*) avec le clip spécifié par le paramètre *cible* ou avec le clip occupant actuellement le niveau spécifié dans le paramètre *profondeur*. Les deux clips doivent avoir le même clip parent. La permutation du niveau de profondeur des clips a pour effet de déplacer un clip devant ou derrière l'autre. Si un clip est en cours d'interpolation lors de l'appel de cette méthode, l'interpolation est arrêtée. Pour plus d'informations, consultez *Gestion des profondeurs de clip*, page 135.

Consultez également

`_level`, `MovieClip.getDepth()`, `MovieClip.getInstanceAtDepth()`, `MovieClip.getNextHighestDepth()`

MovieClip.tabChildren

Disponibilité

Flash Player 6.

Usage

mon_mc.tabChildren

Description

Propriété : undefined par défaut. Si tabChildren a pour valeur undefined ou true, les enfants du clip sont inclus dans l'ordre de tabulation automatique. Si la valeur de tabChildren est false, les enfants du clip ne sont pas inclus dans l'ordre de tabulation automatique.

Exemple

Un objet d'interface de zone de liste construit en tant que clip contient plusieurs éléments. L'utilisateur peut cliquer sur chaque élément pour le sélectionner, de sorte que chaque élément est un bouton. Cependant, seule la zone de liste même devrait être un arrêt de tabulation. Les éléments à l'intérieur de la zone de liste devraient être exclus de l'ordre de tabulation. Pour ce faire, la propriété tabChildren de la zone de liste devrait être définie 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.

Consultez également

Button.tabIndex, mon_mc.tabEnabled, MovieClip.tabIndex, TextField.tabIndex

mon_mc.tabEnabled

Disponibilité

Flash Player 6.

Usage

mon_mc.tabEnabled

Description

Propriété : spécifie si *mon_mc* est inclus dans l'ordre de tabulation automatique. Valeur `undefined` par défaut.

Si `tabEnabled` est `undefined`, l'objet est inclus dans l'ordre de tabulation automatique uniquement s'il définit au moins un gestionnaire de boutons, par exemple `MovieClip.onRelease`. Si `tabEnabled` est `true`, l'objet est inclus dans l'ordre de tabulation automatique. Si la propriété `tabIndex` est également définie avec une valeur, l'objet est également inclus dans l'ordre de tabulation automatique.

Si `tabEnabled` est `false`, l'objet n'est pas inclus dans l'ordre de tabulation automatique ou personnalisé, même si la propriété `tabIndex` est définie. Toutefois, si `MovieClip.tabChildren` a pour valeur `true`, les enfants du clip peuvent rester inclus dans l'ordre de tabulation automatique, même si `tabEnabled` est défini sur la valeur `false`.

Consultez également

`Button.tabEnabled`, `MovieClip.tabChildren`, `MovieClip.tabIndex`, `TextField.tabEnabled`

MovieClip.tabIndex

Disponibilité

Flash Player 6.

Usage

mon_mc.tabIndex

Description

Propriété : permet de définir l'ordre de tabulation automatique des objets d'une animation. La propriété tabIndex est undefined par défaut. Vous pouvez définir tabIndex pour une occurrence de bouton, de clip ou de champ de texte.

Si un objet du fichier SWF contient une propriété tabIndex, l'ordre de tabulation automatique est désactivé et l'ordre de tabulation est calculé en fonction des propriétés tabIndex des objets du fichier SWF. L'ordre de tabulation personnalisé n'inclut que les objets possédant des propriétés tabIndex.

La propriété tabIndex doit être un entier positif. Les objets sont placés dans l'ordre correspondant à leurs propriétés tabIndex, dans un ordre croissant. Un objet dont la valeur tabIndex est 1 précède un objet dont la valeur tabIndex est de 2. L'ordre de tabulation personnalisé ignore les relations hiérarchiques entre les objets d'un fichier SWF. Tous les objets du fichier SWF munis de propriétés tabIndex sont placés dans l'ordre de tabulation. Vous ne devriez pas utiliser la même valeur tabIndex pour plusieurs objets.

Consultez également

Button.tabIndex, TextField.tabIndex

MovieClip._target

Disponibilité

Flash Player 4.

Usage

mon_mc._target

Description

Propriété (lecture seule) : renvoie le chemin cible de l'occurrence de clip spécifié par *mon_mc*.

MovieClip._totalframes

Disponibilité

Flash Player 4.

Usage

mon_mc._totalframes

Description

Propriété (lecture seule) : renvoie le nombre total d'images de l'occurrence de clip spécifiée dans le paramètre *mon_mc*.

MovieClip.trackAsMenu

Disponibilité

Flash Player 6.

Usage

mon_mc.trackAsMenu

Description

Propriété : une propriété booléenne qui indique si d'autres boutons ou clips peuvent recevoir des événements de relâchement de bouton de souris. Cela vous permet de créer des menus. Vous pouvez définir la propriété `trackAsMenu` pour n'importe quel objet de bouton ou clip. Si la propriété `trackAsMenu` n'existe pas, le comportement par défaut est `false`.

Vous pouvez changer la propriété `trackAsMenu` à tout moment, le clip de bouton modifié prenant immédiatement le nouveau comportement.

Consultez également

Button.trackAsMenu

MovieClip.unloadMovie()

Disponibilité

Flash Player 5.

Usage

mon_mc.unloadMovie()

Paramètres

Aucun.

Renvoie

Rien.

Description

Méthode : renvoie le contenu d'une occurrence de clip. Les propriétés d'occurrence et les gestionnaires de clip restent.

Pour supprimer l'occurrence, y compris ses propriétés et les gestionnaires de clip, utilisez MovieClip.removeMovieClip().

Consultez également

MovieClip.attachMovie(), MovieClip.loadMovie, unloadMovie(), unloadMovieNum()

MovieClip._url

Disponibilité

Flash Player 4.

Usage

mon_mc._url

Description

Propriété (lecture seule) : récupère l'URL du fichier SWF d'où le clip a été téléchargé.

MovieClip.useHandCursor

Disponibilité

Flash Player 6.

Usage

mon_mc.useHandCursor

Description

Propriété : une valeur booléenne indiquant si le curseur en forme de main (main) apparaît lorsque la souris passe au-dessus d'un clip de bouton. La valeur par défaut de useHandCursor est true. Si useHandCursor est défini sur la valeur true, la main utilisée pour les boutons s'affiche lorsque la souris passe au-dessus d'un clip de bouton. Si useHandCursor est false, c'est le curseur de flèche qui est utilisé.

Vous pouvez changer la propriété useHandCursor à tout moment, le clip de bouton modifié prenant immédiatement le nouveau comportement. La propriété useHandCursor peut être lue à partir d'un objet prototype.

MovieClip._visible

Disponibilité

Flash Player 4.

Usage

mon_mc._visible

Description

Propriété : une valeur booléenne indiquant si le clip spécifié par *mon_mc* est visible. Les clips qui ne sont pas visibles (propriété _visible définie sur false) sont désactivés. Par exemple, le bouton d'un clip dont la propriété _visible est définie sur la valeur false ne peut pas être cliqué.

Consultez également

Button._visible, TextField._visible

MovieClip._width

Disponibilité

Flash Player 4 en tant que propriété en lecture seule.

Usage

mon_mc._width

Description

Propriété : la largeur du clip, en pixels.

Exemple

L'exemple suivant définit les propriétés de hauteur et de largeur d'un clip lorsque l'utilisateur clique sur la souris.

```
onclipEvent(mouseDown) {
  _width=200;
  _height=200;
}
```

Consultez également

MovieClip._height

MovieClip._x

Disponibilité

Flash Player 3.

Usage

mon_mc._x

Description

Propriété : un entier définissant 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 fait référence au coin supérieur gauche de la scène comme (0, 0). Si le clip se trouve dans un clip qui a subi des transformations, le clip est dans le système de coordonnées locales du clip imbriquant. Donc, pour un clip ayant pivoté de 90 degrés dans le sens inverse des aiguilles d'une montre, les enfants de ce clip héritent d'un système de coordonnées qui a pivoté de 90 degrés dans le sens inverse des aiguilles d'une montre. Les coordonnées du clip font référence à la position du point d'alignement.

Consultez également

MovieClip._xscale, MovieClip._y, MovieClip._yscale

MovieClip._xmouse

Disponibilité

Flash Player 5.

Usage

mon_mc._xmouse

Description

Propriété (lecture seule) : renvoie la coordonnée *x* de la position de la souris.

Consultez également

Classe Mouse, MovieClip._ymouse

MovieClip._xscale

Disponibilité

Flash Player 4.

Usage

mon_mc._xscale

Description

Propriété : détermine l'échelle horizontale (*pourcentage*) du clip telle qu'appliquée à partir du point d'alignement du clip. Le point d'alignement par défaut est (0,0).

Le redimensionnement du système de coordonnées locales affecte les paramètres de propriété _x et _y, définis en pixels entiers. Par exemple, si le clip parent est dimensionné à 50 %, la définition de la propriété _x bouge un objet du clip de la moitié du nombre de pixels d'une animation à 100 %.

Consultez également

MovieClip._x, MovieClip._y, MovieClip._yscale

MovieClip._y

Disponibilité

Flash Player 3.

Usage

mon_mc.*_y*

Description

Propriété : définit la coordonnée *y* du clip par rapport aux coordonnées locales du clip parent. Si un clip se trouve dans le scénario principal, son système de coordonnées fait référence au coin supérieur gauche de la scène comme (0, 0). Si le clip se trouve dans un clip qui a subi des transformations, le clip est dans le système de coordonnées locales du clip imbriquant. Donc, pour un clip ayant pivoté de 90 degrés dans le sens inverse des aiguilles d'une montre, les enfants de ce clip héritent d'un système de coordonnées qui a pivoté de 90 degrés dans le sens inverse des aiguilles d'une montre. Les coordonnées du clip font référence à la position du point d'alignement.

Consultez également

`MovieClip._x`, `MovieClip._xscale`, `MovieClip._yscale`

MovieClip._ymouse

Disponibilité

Flash Player 5.

Usage

mon_mc.*_ymouse*

Description

Propriété (lecture seule) : indique la coordonnée *y* de la position de la souris.

Consultez également

Classe Mouse, `MovieClip._xmouse`

MovieClip._yscale

Disponibilité

Flash Player 4.

Usage

mon_mc.`_yscale`

Description

Propriété : définit l'échelle verticale (*`pourcentage`*) du clip telle qu'appliquée à partir du point d'alignement du clip. Le point d'alignement par défaut est (0,0).

Le redimensionnement du système de coordonnées locales affecte les paramètres de propriété `_x` et `_y`, définis en pixels entiers. Par exemple, si le clip parent est dimensionné à 50 %, la définition de la propriété `_x` bouge un objet du clip de la moitié du nombre de pixels d'une animation à 100 %.

Consultez également

`MovieClip._x`, `MovieClip._xscale`, `MovieClip._y`

Classe MovieClipLoader

Disponibilité

Flash Player 7.

Description

Cette classe vous permet d'implémenter des rappels des écouteurs offrant des informations d'état durant le chargement des fichiers SWF et JPEG (téléchargés) dans les clips. Pour utiliser les fonctionnalités MovieClipLoader, utilisez `MovieClipLoader.loadClip()` plutôt que `loadMovie()` ou `MovieClip.loadMovie()` pour charger les fichiers SWF.

Une fois la commande `MovieClipLoader.loadClip()` activée, les événements suivants se produisent, dans l'ordre spécifié :

- Une fois les premiers octets du fichier téléchargé enregistrés sur disque, l'écouteur `MovieClipLoader.onLoadStart()` est appelé.
- Si vous avez implémenté l'écouteur `MovieClipLoader.onLoadProgress()`, ce dernier est appelé durant le processus de chargement.

  Remarque : Vous pouvez appeler `MovieClipLoader.getProgress()` à tout moment durant le processus de chargement.

- Une fois le fichier téléchargé entier enregistré sur disque, l'écouteur `MovieClipLoader.onLoadComplete()` est appelé.
- Une fois les premières actions d'image du fichier téléchargé effectuées, l'écouteur `MovieClipLoader.onLoadInit()` est appelé.

Une fois `MovieClipLoader.onLoadInit()` appelé, vous pouvez définir les propriétés, utiliser les méthodes, ou bien interagir avec le clip chargé.

Si le chargement du fichier échoue, l'écouteur `MovieClipLoader.onLoadError()` est appelé.

Méthodes de la classe MovieClipLoader

Méthode	Description
MovieClipLoader.addListener()	Enregistre un objet pour recevoir une notification lors de l'appel d'un gestionnaire d'événements MovieClipLoader.
MovieClipLoader.getProgress()	Renvoie le nombre d'octets chargés et le nombre total d'octets pour un fichier en cours de chargement à l'aide de MovieClipLoader.loadClip().
MovieClipLoader.loadClip()	Charge un fichier SWF ou JPEG dans un clip dans Flash Player, lors de la lecture du clip d'origine.
MovieClipLoader.removeListener()	Supprime un objet enregistré à l'aide de MovieClipLoader.addListener().
MovieClipLoader.unloadClip()	Supprime un clip chargé via MovieClipLoader.loadClip().

Ecouteurs de la classe MovieClipLoader

Ecouteur	Description
MovieClipLoader.onLoadComplete()	Appelé si un fichier chargé à l'aide de MovieClipLoader.loadClip() est entièrement téléchargé.
MovieClipLoader.onLoadError()	Appelé si le chargement d'un fichier chargé à l'aide de MovieClipLoader.loadClip() a échoué.
MovieClipLoader.onLoadInit()	Appelé si les actions sur la première image du clip chargées ont été exécutées.
MovieClipLoader.onLoadProgress()	Appelé à chaque fois que le contenu de chargement est enregistré sur disque durant le processus de chargement.
MovieClipLoader.onLoadStart()	Appelé si un appel de MovieClipLoader.loadClip() a commencé le téléchargement d'un fichier.

Constructeur de la classe MovieClipLoader

Disponibilité

Flash Player 7.

Usage

```
new MovieClipLoader()
```

Paramètres

Aucun.

Renvoie

Rien.

Description

Constructeur : crée un objet MovieClipLoader que vous pouvez utiliser pour implémenter un nombre d'écouteurs pour répondre aux événements lors du téléchargement d'un fichier SWF ou JPEG.

Exemple

Pour plus d'informations, consultez `MovieClipLoader.loadClip()`.

Consultez également

`MovieClipLoader.addListener()`

MovieClipLoader.addListener()

Disponibilité

Flash Player 7.

Usage

```
mon_mcl.addListener(objetDecoute)
```

Paramètres

`objetDecoute` Un objet qui écoute une notification de rappel à partir des gestionnaires d'événements MovieClipLoader.

Renvoie

Rien.

Description

Méthode : enregistre un objet pour recevoir une notification lors de l'appel d'un gestionnaire d'événements MovieClipLoader.

Exemple

Pour plus d'informations, consultez `MovieClipLoader.loadClip()`.

Consultez également

`MovieClipLoader.onLoadComplete()`, `MovieClipLoader.onLoadError()`, `MovieClipLoader.onLoadInit()`, `MovieClipLoader.onLoadProgress()`, `MovieClipLoader.onLoadStart()`, `MovieClipLoader.removeListener()`

MovieClipLoader.getProgress()

Disponibilité

Flash Player 7.

Usage

```
mon_mcl.getProgress(cible_mc)
```

Paramètres

cible_mc Un fichier SWF ou JPEG chargé via `MovieClipLoader.loadClip()`.

Renvoie

Un objet qui a deux propriétés intégrées : `bytesLoaded` et `bytesTotal`.

Description

Méthode : renvoie le nombre d'octets chargés et le nombre total d'octets pour un fichier en cours de chargement à l'aide de `MovieClipLoader.loadClip()` ; pour les clips compressés, ce nombre reflète le nombre d'octets compressés. Cette méthode vous permet de demander explicitement ces informations, plutôt que (ou en plus) d'enregistrer une fonction d'écouteur `MovieClipLoader.onLoadProgress()`.

Exemple

Pour plus d'informations, consultez `MovieClipLoader.loadClip()`.

Consultez également

`MovieClipLoader.onLoadProgress()`

MovieClipLoader.loadClip()

Disponibilité

Flash Player 7.

Usage

```
mon_mcl.loadClip("url", cible )
```

Paramètres

url L'URL absolue ou relative du fichier SWF ou JPEG à charger. Un chemin relatif doit être relatif au fichier SWF au niveau 0. Des URL absolues doivent inclure la référence au protocole, comme http:// ou file:///. Les noms de fichiers ne peuvent pas inclure de spécifications de disque dur.

cible Le chemin cible d'un clip, ou un nombre entier spécifiant le niveau, dans Flash Player, auquel ce clip sera chargé. Le clip cible sera remplacé par l'animation ou l'image chargée.

Renvoie

Rien.

Description

Charge un fichier SWF ou JPEG dans un clip dans Flash Player, lors de la lecture du clip d'origine. La méthode loadMovie vous permet d'afficher plusieurs animations en une fois et de basculer entre des animations sans charger un autre document HTML.

L'utilisation de cette méthode au lieu de la méthode `loadMovie()` ou `MovieClip.loadMovie()` offre de nombreux avantages :

- Le gestionnaire `MovieClipLoader.onLoadStart()` est appelé au démarrage du chargement.
- Le gestionnaire `MovieClipLoader.onLoadError()` est appelé si le clip ne peut pas être chargé.
- Le gestionnaire `MovieClipLoader.onLoadProgress()` est appelé durant le processus de chargement.
- Le gestionnaire `MovieClipLoader.onLoadInit()` est appelé une fois les actions sur la première image du clip exécutées, afin que vous puissiez commencer la manipulation du clip chargé.

Une animation ou image chargée dans un clip hérite de ses propriétés de position, rotation et échelle. Vous pouvez utiliser le chemin cible du clip pour cibler l'animation chargée.

Vous pouvez utiliser cette méthode pour charger un ou plusieurs fichiers dans un même clip ou à un même niveau ; les objets d'écoute MovieClipLoader reçoivent l'occurrence de chargement du clip cible en tant que paramètre. Vous pouvez également créer un objet MovieClipLoader différent pour chaque fichier chargé.

Utilisez `MovieClipLoader.unloadClip()` pour supprimer les clips ou les images chargées via cette méthode, ou pour annuler une opération de chargement en cours.

Exemple

L'exemple suivant illustre l'utilisation d'une grande partie des méthodes MovieClipLoader et des écouteurs.

```
// premier ensemble d'écouteurs
var mon_mcl = new MovieClipLoader();
monEcouteur = new Object();
monEcouteur.onLoadStart = function (cible_mc)
{
maTrace ("*********Première occurrence de mon_mcl*********");
maTrace ("Votre chargement a commencé sur le clip . = " + cible_mc);
var loadProgress = mon_mcl.getProgress(cible_mc);
maTrace(loadProgress.bytesLoaded + " = octets chargés au démarrage");
maTrace(loadProgress.bytesTotal + " = total des octets au démarrage");
}
monEcouteur.onLoadProgress = function (cible_mc, loadedBytes, totalBytes)
{
maTrace ("*********Progression de la première occurrence de
  mon_mcl*********");
maTrace ("onLoadProgress() rappelé sur clip " + cible_mc);
maTrace(loadedBytes + " = octets chargés au rappel de progression " );
maTrace(totalBytes + " = total octets au rappel de progression \n");
}
monEcouteur.onLoadComplete = function (cible_mc)
{
maTrace ("*********Première occurrence de mon_mcl*********");
maTrace ("Votre chargement est effectué sur le clip = " + cible_mc);
var loadProgress = mon_mcl.getProgress(cible_mc);
maTrace(loadProgress.bytesLoaded + " = octets chargés à la fin");
maTrace(loadProgress.bytesTotal + " = total des octets à la fin");
}
monEcouteur.onLoadInit = function (cible_mc)
{
maTrace ("*********Première occurrence de mon_mcl*********");
maTrace ("Clip = " + cible_mc + " est maintenant initialisé");
// vous pouvez maintenant effectuer tous les réglages requis, par exemple :
cible_mc._width = 100;
cible_mc._width = 100;
}
objetDecoute.onLoadError() = function (cible_mc, errorCode) {
{
maTrace ("*********Première occurrence de mon_mcl*********");
maTrace ("CODE ERREUR = " + codeErreur);
maTrace ("Votre chargement a échoué sur le clip = " + cible_mc + "\n");
 }
mon_mcl.addListener(monEcouteur);
//Charger maintenant les fichiers dans leurs emplacements cibles.
// charge dans les clips - chaînes utilisées en tant que cibles
mon_mcl.loadClip("http://www.undomaine.quelquepart.com/
  unFichier.swf","_root.monMC");
mon_mcl.loadClip("http://www.undomaine.quelquepart.com/unAutreFichier.swf",
  "_level0.monMC2");
//échec du chargement
mon_mcl.loadClip("http://www.undomaine.quelquepart.com/unFichier.jpg",
  _root.monMC5);

// charge dans les clips - occurrence de clip utilisée en tant que cible.
mon_mcl.loadClip("http://www.undomaine.quelquepart.com/unAutreFichier.jpg",
  _level0.monMC3);
```

```
// charge dans _level1
mon_mcl.loadClip("fichier:///C:/media/images/uneImage.jpg", 1);

//Second ensemble d'écouteurs
var autre_mcl = new MovieClipLoader();
monEcouteur2 = new Object();
monEcouteur2.onLoadStart = function (cible_mc)
{
maTrace("*********Seconde occurrence mon_mcl*********");
maTrace ("Votre chargement a commencé sur le clip22. = " + cible_mc);
var loadProgress = mon_mcl.getProgress(cible_mc);
maTrace(loadProgress.bytesLoaded + " = octets chargés au démarrage");
maTrace(loadProgress.bytesTotal + " = total des octets au démarrage");
}
monEcouteur2.onLoadComplete = function (cible_mc)
{
maTrace("*********Seconde occurrence mon_mcl*********");
maTrace ("Votre chargement est effectué sur le clip = " + cible_mc);
var loadProgress = mon_mcl.getProgress(cible_mc);
maTrace(loadProgress.bytesLoaded + " = octets chargés à la fin");
maTrace(loadProgress.bytesTotal + " = total des octets à la fin");
}
monEcouteur2.onLoadError = function (cible_mc, codeErreur)
{
maTrace("*********Seconde occurrence mon_mcl*********");
maTrace ("CODE ERREUR = " + codeErreur);
maTrace ("Votre chargement a échoué sur le clip = " + cible_mc + "\n");
 }
autre_mcl.addListener(monEcouteur2);
//Charger maintenant les fichiers dans leurs emplacements cibles (à l'aide de
  la seconde occurrence de MovieClipLoader)
autre_mcl.loadClip("http://www.undomaine.quelquepart.com/
  encoreUnAutreFichier.jpg", _root.monMC4);
// Activer l'instruction suivante une fois le téléchargement terminé,
// et une fois mon_mcl.onLoadInit appelé.
// mon_mcl.removeListener(monEcouteur)
// mon_mcl.removeListener(monEcouteur2)
```

Consultez également

MovieClipLoader.unloadClip()

MovieClipLoader.onLoadComplete()

Disponibilité

Flash Player 7.

Usage

```
objetDecoute.onLoadComplete() = function (cible_mc) {
  // vos instructions
}
```

Paramètres

objetDecoute Un objet d'écoute ajouté via `MovieClipLoader.addListener()`.

cible_mc Le clip chargé par une méthode `MovieClipLoader.loadClip()`.

Renvoie

Rien.

Description

Ecouteur : appelé si un fichier chargé à l'aide de `MovieClipLoader.loadClip()` est entièrement téléchargé.

Exemple

Pour plus d'informations, consultez `MovieClipLoader.loadClip()`.

Consultez également

`MovieClipLoader.addListener()`, `MovieClipLoader.onLoadStart()`, `MovieClipLoader.onLoadError()`

MovieClipLoader.onLoadError()

Disponibilité

Flash Player 7.

Usage

```
objetDecoute.onLoadError() = function(cible_mc, codeErreur) {
  // vos instructions
}
```

Paramètres

objetDecoute Un objet d'écoute ajouté via `MovieClipLoader.addListener()`.

cible_mc Le clip chargé par une méthode `MovieClipLoader.loadClip()`.

codeErreur Une chaîne qui explique la cause de l'échec.

Renvoie

L'une des deux chaînes : « URLNotFound » ou « LoadNeverCompleted ».

Description

Ecouteur : appelé en cas d'échec du chargement d'un fichier à l'aide de `MovieClipLoader.loadClip()`.

La chaîne « URLNotFound » est renvoyée si ni l'écouteur `MovieClipLoader.onLoadStart()`, ni l'écouteur `MovieClipLoader.onLoadComplete()` ne sont appelés. Par exemple, si un serveur est en panne ou si le fichier est introuvable, ces écouteurs ne sont pas appelés.

La chaîne « LoadNeverCompleted » est renvoyée si `MovieClipLoader.onLoadStart()` a été appelé, mais pas `MovieClipLoader.onLoadComplete()`. Par exemple, si `MovieClipLoader.onLoadStart()` est appelé mais que le téléchargement est interrompu en raison d'une surcharge du serveur, une panne du serveur, et ainsi de suite, `MovieClipLoader.onLoadComplete()` n'est pas appelé.

Exemple

Pour plus d'informations, consultez `MovieClipLoader.loadClip()`.

MovieClipLoader.onLoadInit()

Disponibilité

Flash Player 7.

Usage

```
objetDecoute.onLoadInit() = function (cible_mc) {
  // vos instructions
}
```

Paramètres

objetDecoute Un objet d'écoute ajouté via `MovieClipLoader.addListener()`.

cible_mc Le clip chargé par une méthode `MovieClipLoader.loadClip()`.

Renvoie

Rien.

Description

Appelé si les actions sur la première image du clip chargées ont été exécutées. Une fois MovieClipLoader.onLoadInit() appelé, vous pouvez définir les propriétés, utiliser les méthodes, ou bien interagir avec le clip chargé.

Exemple

Pour plus d'informations, consultez `MovieClipLoader.loadClip()`.

Consultez également

`MovieClipLoader.onLoadStart()`

MovieClipLoader.onLoadProgress()

Disponibilité

Flash Player 7.

Usage

```
objetDecoute.onLoadProgress() =
  function(cible_mc [, loadedBytes [, totalBytes ] ] ) {
  // vos instructions
}
```

Paramètres

objetDecoute Un objet d'écoute ajouté via `MovieClipLoader.addListener()`.

cible_mc Le clip chargé par une méthode `MovieClipLoader.loadClip()`.

loadedBytes Le nombre d'octets chargés au moment de l'appel de l'écouteur.

totalBytes Le nombre total d'octets contenus dans le fichier en cours de chargement.

Renvoie

Rien.

Description

Ecouteur : appelé à chaque fois que le contenu de chargement est enregistré sur disque durant le processus de chargement (à savoir, entre `MovieClipLoader.onLoadStart()` et `MovieClipLoader.onLoadComplete()`). Vous pouvez utiliser cette méthode pour afficher des informations concernant la progression du téléchargement, à l'aide des paramètres `loadedBytes` et `totalBytes`.

Exemple

Pour plus d'informations, consultez `MovieClipLoader.loadClip()`.

Consultez également

`MovieClipLoader.getProgress()`

MovieClipLoader.onLoadStart()

Disponibilité

Flash Player 7.

Usage

```
objetDecoute.onLoadStart() = function (cible_mc) {
  // vos instructions
}
```

Paramètres

objetDecoute Un objet d'écoute ajouté via `MovieClipLoader.addListener()`.

cible_mc Le clip chargé par une méthode `MovieClipLoader.loadClip()`.

Renvoie

Rien.

Description

Ecouteur : appelé si un appel de `MovieClipLoader.loadClip()` a commencé le téléchargement d'un fichier.

Exemple

Pour plus d'informations, consultez `MovieClipLoader.loadClip()`.

Consultez également

`MovieClipLoader.onLoadError()`, `MovieClipLoader.onLoadInit()`, `MovieClipLoader.onLoadComplete()`

MovieClipLoader.removeListener()

Disponibilité

Flash Player 7.

Usage

```
mon_mcl.removeListener(objetDecoute)
```

Paramètres

objetDecoute Un objet d'écoute ajouté via `MovieClipLoader.addListener()`.

Renvoie

Rien.

Description

Méthode : supprime un objet utilisé pour recevoir une notification lors de l'appel d'un gestionnaire d'événements MovieClipLoader.

Exemple

Pour plus d'informations, consultez `MovieClipLoader.loadClip()`.

MovieClipLoader.unloadClip()

Disponibilité

Flash Player 7.

Usage

mon_mcl.unloadClip(*cible*)

Paramètres

cible La chaîne ou le nombre entier passés dans l'appel correspondant à *mon_mcl*.loadClip().

Renvoie

Rien.

Description

Méthode : supprime un clip chargé via MovieClipLoader.loadClip(). Si vous activez cette commande lors du chargement d'un clip, MovieClipLoader.onLoadError() est appelé.

Consultez également

MovieClipLoader.loadClip()

NaN

Disponibilité

Flash Player 5.

Usage

NaN

Description

Variable : une variable prédéfinie avec la valeur IEEE-754 pour NaN (Not a Number). Pour déterminer si un nombre est NaN, utilisez isNaN().

Consultez également

isNaN(), Number.NaN

ne (pas égal à – spécifique aux chaînes)

Disponibilité

Flash Player 4. Cet opérateur est déconseillé et remplacé par l'opérateur `!= (inégalité)`.

Usage

`expression1` `ne` *`expression2`*

Paramètres

`expression1, expression2` Nombres, chaînes ou variables.

Renvoie

Une valeur booléenne.

Description

Opérateur (comparaison) : compare *`expression1`* avec *`expression2`* et renvoie `true` si *`expression1`* n'est pas égale à *`expression2`* ; sinon, renvoie `false`.

Consultez également

`!= (inégalité)`

Classe NetConnection

Disponibilité

Flash Player 7.

Remarque : Cette classe est également prise en charge dans Flash Player 6, dans le cadre d'une utilisation avec Flash Communication Server. Pour plus d'informations, consultez la documentation de Flash Communication Server.

Description

La classe NetConnection permet de lire des fichiers FLV en flux continu à partir d'un disque local ou d'une adresse HTTP. Pour plus d'informations sur le flux vidéo, consultez *Lecture dynamique des fichiers FLV externes*, page 209.

Méthodes de la classe NetConnection

Méthode	Description
`NetConnection.connect()`	Ouvre une connexion locale qui vous permet de lire les fichiers vidéo (FLV) depuis une adresse HTTP ou depuis un système de fichiers local.

Constructeur de la classe NetConnection

Disponibilité

Flash Player 7.

Remarque : Cette classe est également prise en charge dans Flash Player 6, dans le cadre d'une utilisation avec Flash Communication Server. Pour plus d'informations, consultez la documentation de Flash Communication Server.

Usage

```
new NetConnection()
```

Paramètres

Aucun.

Renvoie

Rien.

Description

Constructeur : crée un objet NetConnection que vous pouvez utiliser pour lire les fichiers de diffusion en flux locaux (FLV). Après la création de l'objet NetConnection, utilisez `NetConnection.connect()` pour effectuer la connexion réelle.

La lecture des fichiers FLV externes offre plusieurs avantages par rapport à l'intégration de vidéo dans un document Flash : performances et gestion de la mémoire améliorées, indépendance des cadences vidéo et Flash. Pour plus d'informations, consultez *Lecture dynamique des fichiers FLV externes*, page 209.

Consultez également

Classe NetStream, `Video.attachVideo()`

NetConnection.connect()

Disponibilité

Flash Player 7.

Remarque : Cette méthode est également prise en charge dans Flash Player 6 dans le cadre d'une utilisation avec Flash Communication Server. Pour plus d'informations, consultez la documentation de Flash Communication Server.

Usage

```
mon_nc.connect(nul);
```

Paramètres

Aucun (vous devez passer `null`).

Renvoie

Rien.

Description

Ouvre une connexion locale lqui vous permet de lire les fichiers vidéo (FLV) depuis une adresse HTTP ou depuis un système de fichiers local.

Consultez également

Classe NetStream

Classe NetStream

Disponibilité

Flash Player 7.

Remarque : Cette classe est également prise en charge dans Flash Player 6, dans le cadre d'une utilisation avec Flash Communication Server. Pour plus d'informations, consultez la documentation de Flash Communication Server.

Description

La classe NetStream fournit des méthodes et des propriétés permettant de lire des fichiers Flash Video (FLV) depuis le système de fichiers local ou une adresse HTTP. Vous devez utiliser un objet NetStream pour diffuser en flux la vidéo via un objet NetConnection. La lecture des fichiers FLV externes offre plusieurs avantages par rapport à l'intégration de vidéo dans un document Flash : performances et gestion de la mémoire améliorées, indépendance des cadences vidéo et Flash. Cette classe offre plusieurs méthodes et propriétés que vous pouvez utiliser pour suivre la progression du fichier lors de son chargement et de sa lecture. Ces méthodes permettent en outre à l'utilisateur de contrôler la lecture (arrêt, pause, et ainsi de suite).

Pour plus d'informations sur le flux vidéo, consultez *Lecture dynamique des fichiers FLV externes*, page 209.

Méthodes de la classe NetStream

Les méthodes et propriétés suivantes des classes NetConnection et NetStream sont utilisées pour contrôler la lecture des fichiers FLV.

Méthode	Objectif
`NetStream.close()`	Ferme le flux mais ne supprime pas l'objet vidéo.
`NetStream.pause()`	Arrête ou relance la lecture d'un flux vidéo.
`NetStream.play()`	Commence la lecture d'un fichier vidéo externe (FLV).
`NetStream.seek()`	Recherche une position spécifique dans le fichier FLV.
`NetStream.setBufferTime()`	Spécifie la durée de la mise en mémoire tampon des données avant le démarrage de l'affichage du flux vidéo.

Propriétés de la classe NetStream

Propriété	Description
`NetStream.bufferLength`	Le nombre de secondes de données actuellement incluses dans la mémoire tampon.
`NetStream.bufferTime`	Lecture seule : le nombre de secondes assignées à la mémoire tampon par `NetStream.setBufferTime()`.
`NetStream.bytesLoaded`	Lecture seule : le nombre d'octets de données chargées dans le lecteur.
`NetStream.bytesTotal`	Lecture seule : la taille totale, en octets, du fichier en cours de chargement dans le lecteur.

Propriété	Description
NetStream.currentFps	Le nombre d'images par seconde en cours d'affichage.
NetStream.time	Lecture seule : la position de la tête de lecture, en secondes.

Gestionnaires d'événement de la classe NetStream

Gestionnaire d'événement	Description
NetStream.onStatus	Appelé à chaque fois qu'un changement d'état ou une erreur est consigné(e) pour l'objet NetStream.

Constructeur de la classe NetStream

Disponibilité

Flash Player 7.

Remarque : Cette classe est également prise en charge dans Flash Player 6, dans le cadre d'une utilisation avec Flash Communication Server. Pour plus d'informations, consultez la documentation de Flash Communication Server.

Usage

```
new NetStream(mon_nc)
```

Paramètres

mon_nc Un objet NetConnection.

Renvoie

Rien.

Description

Constructeur : crée un flux vidéo utilisable pour la lecture des fichiers FLV via l'objet NetConnection spécifié.

Exemple

Le code suivant construit dans un premier temps un nouvel objet NetConnection, `mon_nc`, puis l'utilise pour construire un nouvel objet NetStream nommé `fluxVideo_ns`.

```
mon_nc = new NetConnection();
mon_nc.connect(null);
fluxVideo_ns = new NetStream(mon_nc);
```

Consultez également

Classe NetConnection, Classe NetStream, `Video.attachVideo()`

NetStream.bufferLength

Disponibilité

Flash Player 7.

Remarque : Cette propriété est également prise en charge dans Flash Player 6 dans le cadre d'une utilisation avec Flash Communication Server. Pour plus d'informations, consultez la documentation de Flash Communication Server.

Usage

```
mon_ns.bufferLength
```

Description

Le nombre de secondes de données actuellement incluses dans la mémoire tampon. Vous pouvez utiliser cette propriété en association avec `NetStream.bufferTime` pour évaluer le degré de remplissage de la mémoire tampon (par exemple, pour afficher un retour à l'attention de l'utilisateur qui attend les données à charger dans la mémoire tampon).

Consultez également

`NetStream.bytesLoaded`

NetStream.bufferTime

Disponibilité

Flash Player 7.

Remarque : Cette propriété est également prise en charge dans Flash Player 6 dans le cadre d'une utilisation avec Flash Communication Server. Pour plus d'informations, consultez la documentation de Flash Communication Server.

Usage

```
monFlux.bufferTime
```

Description

Propriété (lecture seule) : le nombre de secondes assignées à la mémoire tampon par `NetStream.setBufferTime()`. La valeur par défaut est .1 (un dizième de seconde). Pour déterminer le nombre de secondes restant actuellement dans la mémoire tampon, utilisez `NetStream.bufferLength`.

Consultez également

`NetStream.time`

NetStream.bytesLoaded

Disponibilité

Flash Player 7.

Usage

mon_ns.bytesLoaded

Description

Propriété (lecture seule) : le nombre d'octets de données chargées dans le lecteur. Vous pouvez utiliser cette propriété en association avec NetStream.bytesTotal pour évaluer le degré de remplissage de la mémoire tampon (par exemple, pour afficher un retour à l'attention de l'utilisateur qui attend les données à charger dans la mémoire tampon).

Consultez également

NetStream.bufferLength

NetStream.bytesTotal

Disponibilité

Flash Player 7.

Usage

mon_ns.bytesLoaded

Description

Propriété (lecture seule) : la taille totale, en octets, du fichier en cours de chargement dans le lecteur.

Consultez également

NetStream.bytesLoaded, NetStream.bufferTime

NetStream.close()

Disponibilité

Flash Player 7.

Remarque : Cette méthode est également prise en charge dans Flash Player 6 dans le cadre d'une utilisation avec Flash Communication Server. Pour plus d'informations, consultez la documentation de Flash Communication Server.

Usage

```
mon_ns.close()
```

Paramètres

Aucun.

Renvoie

Rien.

Description

Méthode : arrête la lecture des données sur le flux vidéo, définit la propriété `NetStream.time` sur 0, et rend le flux vidéo disponible pour une autre utilisation. Cette commande supprime en outre la copie locale du fichier FLV téléchargé via l'adresse HTTP.

Exemple

La fonction `onDisconnect()` suivante ferme une connexion et supprime la copie temporaire du fichier someFile.flv stocké sur le disque local.

```
mon_nc = new NetConnection();
mon_nc.connect(null);
mon_ns = nouvel objet NetStream(mon_nc);
mon_ns.play("http://www.someDomain.com/videos/someFile.flv");

function onDisconnect() {
  mon_ns.close()
}
```

Consultez également

`NetStream.pause()`, `NetStream.play()`

NetStream.currentFps

Disponibilité

Flash Player 7.

Remarque : Cette propriété est également prise en charge dans Flash Player 6 dans le cadre d'une utilisation avec Flash Communication Server. Pour plus d'informations, consultez la documentation de Flash Communication Server.

Usage

```
mon_ns.currentFps
```

Description

Propriété (lecture seule) : le nombre d'images par seconde en cours d'affichage. Si vous exportez des fichiers FLV à lire sur plusieurs systèmes, vous pouvez vérifier la valeur correspondante durant le test afin de déterminer le degré de compression à appliquer lors de l'exportation du fichier.

NetStream.onStatus

Disponibilité

Flash Player 7.

Remarque : Ce gestionnaire est également pris en charge dans Flash Player 6 dans le cadre d'une utilisation avec Flash Communication Server. Pour plus d'informations, consultez la documentation de Flash Communication Server.

Usage

```
mon_ns.onStatus = function(objetInfo) {
  // Votre code ici
}
```

Paramètres

objetInfo Un paramètre défini en fonction du message d'état ou d'erreur. Pour plus d'informations concernant ce paramètre, consultez la section « Description », ci-dessous.

Renvoie

Rien.

Description

Appelé à chaque fois qu'un changement d'état ou une erreur est consigné pour l'objet NetStream. Pour répondre à ce gestionnaire d'événements, vous devez créer une fonction pour le traitement de l'objet d'informations.

L'objet d'informations a une propriété de `code` contenant une chaîne décrivant le résultat du gestionnaire `onStatus`, ainsi qu'une propriété `level` contenant une chaîne `"Status"` ou `"Error"`.

Outre 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 lui répondre, Flash traite une fonction affectée à `System.onStatus` s'il en existe une.

Les événements suivants vous informent lorsque certaines activités NetStream se produisent.

Propriété de code	Propriété de niveau	Signification
`NetStream.Buffer.Empty`	Etat	Les données ne sont pas reçues suffisamment rapidement pour remplir la mémoire tampon. Le flux de données est interrompu jusqu'à ce que la mémoire tampon soit à nouveau remplie ; un `NetStream.Buffer.Full` est alors envoyé, et la lecture du flux vidéo reprend.
`NetStream.Buffer.Full`	Etat	La mémoire tampon est pleine, et la lecture du flux vidéo commence.
`NetStream.Play.Start`	Etat	La lecture a commencé.
`NetStream.Play.Stop`	Etat	La lecture s'est arrêtée.
`NetStream.Play.StreamNotFound`	Erreur	Le fichier FLV transmis par la méthode `play()` est introuvable.

Exemple

L'exemple suivant enregistre les données relatives au flux vidéo dans un fichier-journal.

```
mon_ns.onStatus = function(info)
{
  _root.log_stream += "Etat du flux vidéo.\n";
  _root.log_stream += "Evénement : " + info.code + "\n";
  _root.log_stream += "Type : " + info.level + "\n";
}
```

Consultez également

`System.onStatus`

NetStream.pause()

Disponibilité

Flash Player 7.

Remarque : Cette méthode est également prise en charge dans Flash Player 6 dans le cadre d'une utilisation avec Flash Communication Server. Pour plus d'informations, consultez la documentation de Flash Communication Server.

Usage

```
mon_ns.pause( [ reprisePause ] )
```

Paramètres

reprisePause Facultatif : une valeur booléenne spécifiant la mise en pause de la lecture (`true`) ou la reprise de la lecture (`false`). Si vous oubliez ce paramètre, `NetStream.pause()` fonctionne en tant que bascule : la première fois que cette méthode est appelée sur un flux vidéo spécifique, elle arrête la lecture, et, lors de l'appel suivant, elle relance la lecture.

Renvoie

Rien.

Description

Méthode : arrête ou relance la lecture d'un flux vidéo.

La première fois que vous appelez cette méthode (sans envoyer de paramètre), elle arrête la lecture ; la fois suivante, elle relance la lecture. Vous pouvez associer cette méthode à un bouton sur lequel l'utilisateur doit appuyer pour arrêter ou relancer la lecture.

Exemple

Les exemples suivants illustrent certaines utilisations de cette méthode.

```
mon_ns.pause(); // arrête la lecture lors de la première activation
mon_ns.pause(); // relance la lecture
mon_ns.pause(false); // aucun effet, la lecture continue
mon_ns.pause(); // arrête la lecture
```

Consultez également

`NetStream.close()`, `NetStream.play()`

NetStream.play()

Disponibilité

Flash Player 7.

Remarque : Cette méthode est également prise en charge dans Flash Player 6 dans le cadre d'une utilisation avec Flash Communication Server. Pour plus d'informations, consultez la documentation de Flash Communication Server.

Usage

```
mon_ns.play("nomDeFichier");
```

Paramètres

nomDeFichier Le nom d'un fichier FLV à lire, indiqué entre guillemets. Les deux formats http:// et file:// sont pris en charge ; l'emplacement file:// est toujours relatif à l'emplacement du fichier SWF.

Renvoie

Rien.

Description

Méthode : commence la lecture d'un fichier vidéo externe (FLV). Pour visualiser les données vidéo, vous devez appeler une méthode `Video.attachVideo()` ; le fichier audio diffusé en flux avec la vidéo, ou un fichier FLV contenant uniquement des données audio, sont lus automatiquement.

Pour contrôler les données audio associées à un fichier FLV, vous pouvez utiliser la méthode `MovieClip.attachAudio()` pour acheminer ces données vers le clip ; vous pouvez ensuite créer un objet Sound pour contrôler certains aspects de ces données audio. Pour plus d'informations, consultez `MovieClip.attachAudio()`.

Si le fichier FLV est introuvable, le gestionnaire d'événements `NetStream.onStatus` est appelé. Pour arrêter un flux vidéo en cours de lecture, utilisez `NetStream.close()`.

Vous pouvez lire les fichiers FLV locaux stockés dans le même dossier que le fichier SWF ou dans un sous-dossier ; vous ne pouvez pas naviguer dans un dossier de niveau supérieur. Par exemple, si le fichier SWF se trouve dans un dossier nommé /training, et que vous souhaitez lire un fichier vidéo stocké dans le dossier /training/videos, vous devez utiliser la syntaxe suivante :

```
mon_ns.play("file://videos/nomVideo.flv");
```

Pour lire un fichier vidéo stocké dans le dossier /training, vous devez utiliser la syntaxe suivante :

```
mon_ns.play("file://nomVideo.flv");
```

Exemple

L'exemple suivant illustre certains modes d'utilisation de la commande `NetStream.play()`.

```
// Lire un fichier sur l'ordinateur de l'utilisateur
// le dossier utilisateur_joe est un sous-dossier du dossier
//  dans lequel le fichier SWF est stocké
mon_ns.play("fichier://utilisateur_joe/flash/videos/lectureJune26.flv");

// Lire un fichier sur un serveur
mon_ns.play("http://unServeur.unDomaine.com/flash/video/orientation.flv");
```

Consultez également

`MovieClip.attachAudio()`, `NetStream.close()`, `NetStream.pause()`, `Video.attachVideo()`

NetStream.seek()

Disponibilité

Flash Player 7.

Remarque : Cette méthode est également prise en charge dans Flash Player 6 dans le cadre d'une utilisation avec Flash Communication Server. Pour plus d'informations, consultez la documentation de Flash Communication Server.

Usage

`mon_ns``.seek(`*`nombreDeSecondes`*`)`

Paramètres

`nombreDeSecondes` La valeur temporelle approximative, en secondes, à atteindre dans un fichier FLV. La tête de lecture se déplace vers l'image-clé la plus proche de *`nombreDeSecondes`*.

- Pour revenir au début du flux vidéo, définissez *`nombreDeSecondes`* sur 0.
- Pour rechercher plus loin à partir du début du flux vidéo, indiquez le nombre de secondes voulu. Par exemple, pour positionner la tête de lecture à 15 secondes du début, utilisez *`monFlux`*`.seek(15)`.
- Pour effectuer la recherche en fonction de la position courante, passez *`monFlux`*`.time + `*`n`* ou *`monFlux`*`.time - `*`n`* pour avancer ou reculer de *`n`* secondes, respectivement, par rapport à la position courante. Par exemple, pour reculer de 20 secondes par rapport à la position courante, utilisez `mon_ns.seek(mon_ns.time - 20)`.

Renvoie

Rien.

Description

Méthode : recherche l'image-clé la plus proche du nombre de secondes spécifié par rapport au début du flux vidéo. La lecture du flux vidéo reprend lorsque l'emplacement spécifié est atteint.

Exemple

L'exemple suivant illustre certains modes d'utilisation de la commande `NetStream.seek()`.

```
// Revenir au début du flux vidéo
mon_ns.seek(0);

// Aller à un emplacement situé à 30 secondes du début du flux vidéo
mon_ns.seek(30);

//Reculer de trois minutes par rapport à la position courante
mon_ns.seek(mon_ns.time - 180);
```

Consultez également

`NetStream.play()`, `NetStream.time`

NetStream.setBufferTime()

Disponibilité

Flash Player 7.

Remarque : Cette méthode est également prise en charge dans Flash Player 6 dans le cadre d'une utilisation avec Flash Communication Server. Pour plus d'informations, consultez la documentation de Flash Communication Server.

Usage

`mon_ns``.setBufferTime(`*`nombreDeSecondes`*`)`

Paramètres

`nombreDeSecondes` Le nombre de secondes de données à enregistrer dans la mémoire tampon avant que Flash commence l'affichage des données. La valeur par défaut est .1 (un dizième de seconde).

Description

Spécifie la durée de la mise en mémoire tampon des données avant le début de la lecture du flux vidéo. Par exemple, pour vérifier que les 15 premières secondes du flux vidéo sont lues sans interruption, définissez *`nombreDeSecondes`* sur 15 ; Flash commence la lecture du flux vidéo uniquement après l'enregistrement dans la mémoire tampon de 15 secondes de données.

Consultez également

`NetStream.bufferTime`

NetStream.time

Disponibilité

Flash Player 7.

Remarque : Cette propriété est également prise en charge dans Flash Player 6 dans le cadre d'une utilisation avec Flash Communication Server. Pour plus d'informations, consultez la documentation de Flash Communication Server.

Usage

```
mon_ns.time
```

Description

Propriété (lecture seule) : la position de la tête de lecture, en secondes.

Consultez également

`NetStream.bufferLength`, `NetStream.bytesLoaded`

new

Disponibilité

Flash Player 5.

Usage

```
new constructeur()
```

Paramètres

constructeur Une fonction suivie de paramètres facultatifs entre parenthèses. La fonction est généralement le nom du type d'objet (par exemple, Array, Number ou Object) à construire.

Renvoie

Rien.

Description

Opérateur : crée un nouvel objet, initialement anonyme, et appelle la fonction identifiée par le paramètre *constructeur*. L'opérateur `new` transmet à la fonction les paramètres facultatifs entre parenthèses, ainsi que l'objet nouvellement créé, auquel il est fait référence à l'aide du mot-clé `this`. La fonction constructeur peut ensuite utiliser `this` pour définir les variables de l'objet.

Exemple

L'exemple suivant crée la fonction `Livre()`, puis utilise l'opérateur `new` pour créer les objets `livre1` et `livre2`.

```
function Livre(nom, prix){
  this.nom = nom;
  this.prix = prix;
}

livre1 = new Livre("Château de ma mère", 19,99);
livre2 = new Livre("La rigolade", 10,99);
```

Exemple

L'exemple suivant utilise l'opérateur `new` pour créer une occurrence de l'objet Array avec 18 éléments :

```
parcoursGolf_array = new Array(18);
```

Consultez également

`[] (accès tableau)`, `{} (initialisateur d'objet)`

newline

Disponibilité

Flash Player 4.

Usage

```
newline
```

Paramètres

Aucun.

Renvoie

Rien.

Description

Constante : insère un caractère de retour chariot (`\n`) qui produit une ligne vierge dans le texte généré par votre code. Utilisez `newline` pour faire de la place aux informations récupérées par une fonction ou une action de votre code.

Exemple

L'exemple suivant montre comment `newline` affiche la sortie de l'action `trace()` sur plusieurs lignes.

```
var monNom:String = "Lisa", monAge:Number = 30;
trace(monNom + monAge);
trace(monNom + newline + monAge);
```

nextFrame()

Disponibilité

Flash 2.

Usage

```
nextFrame()
```

Paramètres

Aucun.

Renvoie

Rien.

Description

Fonction : envoie la tête de lecture vers l'image suivante et l'arrête.

Exemple

Dans cet exemple, lorsqu'un utilisateur clique sur le bouton, la tête de lecture passe à l'image suivante et s'arrête.

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

nextScene()

Disponibilité

Flash 2.

Usage

```
nextScene()
```

Paramètres

Aucun.

Renvoie

Rien.

Description

Fonction : envoie la tête de lecture vers l'image 1 de la séquence suivante et l'arrête.

Exemple

Dans cet exemple, lorsqu'un utilisateur relâche le bouton, la tête de lecture est envoyée à l'image 1 de la séquence suivante.

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

Consultez également

`prevScene()`

not

Disponibilité

Flash Player 4. Cet opérateur est déconseillé et remplacé par l'opérateur `! (NOT logique)`.

Usage

```
not expression
```

Paramètres

`expression` Toute variable ou expression convertie en valeur booléenne.

Description

Opérateur : effectue une opération NOT logique dans Flash 4 Player.

Consultez également

`! (NOT logique)`

null

Disponibilité

Flash Player 5.

Usage

```
null
```

Paramètres

Aucun.

Renvoie

Rien.

Description

Constante : une valeur spéciale qui peut être affectée aux variables ou renvoyée par une fonction si aucune donnée n'a été fournie. Vous pouvez utiliser `null` pour représenter les valeurs absentes ou n'ayant pas de type de données défini.

Exemple

Dans un contexte numérique, `null` est équivalant à 0. Les tests d'égalité peuvent être effectués avec `null`. Dans cette instruction, un nœud d'arborescence binaire ne possède pas d'enfant gauche, et le champ pour cet enfant gauche peut donc être défini sur `null`.

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

Classe Number

Disponibilité

Flash Player 5 (est devenu un objet natif dans Flash Player 6, améliorant ainsi les performances de manière significative).

Description

La classe Number est un simple objet enveloppe pour le type de données Number. Vous pouvez manipuler des valeurs numériques primitives à l'aide des méthodes et propriétés associées à la classe Number. Cette classe est identique à la classe Number de JavaScript.

Vous devez utiliser un constructeur lorsque vous appelez les méthodes d'un objet Number, mais il n'est pas nécessaire d'utiliser le constructeur lorsque vous appelez les propriétés d'un objet Number. Les exemples suivants spécifient la syntaxe à employer pour appeler les méthodes et propriétés d'un objet Number.

L'exemple suivant appelle la méthode `toString()` de l'objet Number, qui renvoie la chaîne « 1234 ».

```
monNombre = new Number(1234);
monNombre.toString();
```

Cet exemple appelle la propriété `MIN_VALUE` (également appelée « constante ») d'un objet Number :

```
plusPetit = Number.MIN_VALUE
```

Méthodes de la classe Number

Méthode	Description
`Number.toString()`	Renvoie la représentation chaîne d'un objet Number.
`Number.valueOf`	Renvoie la valeur primitive d'un objet Number.

Propriétés de la classe Number

Propriété	Description
`Number.MAX_VALUE`	Constante représentant le plus grand nombre représentable (IEEE-754 double précision). Ce nombre est approximativement 1.79E+308.
`Number.MIN_VALUE`	Constante représentant le plus petit nombre représentable (IEEE-754 double précision). Ce nombre est approximativement 5e-324.
`Number.NaN`	Constante représentant la valeur de Not a Number (`NaN`).
`Number.NEGATIVE_INFINITY`	Constante représentant la valeur pour l'infini négatif.
`Number.POSITIVE_INFINITY`	Constante représentant la valeur pour l'infini positif. Cette valeur est identique à la variable globale `Infinity`.

Constructeur de la classe Number

Disponibilité

Flash Player 5.

Usage

```
new Number(valeur)
```

Paramètres

`valeur` La valeur numérique de l'objet Number créé ou une valeur à convertir en nombre.

Renvoie

Rien.

Description

Constructeur : crée un nouvel objet Number. Vous devez utiliser le constructeur Number lors de l'utilisation de `Number.toString()` et `Number.valueOf`. Il n'est pas nécessaire d'utiliser de constructeur lorsque vous utilisez les propriétés d'un objet Number. Le constructeur `new Number` est principalement utilisé comme support. Un objet Number est différent de la fonction `Number` qui convertit un paramètre en valeur primitive.

Exemple

Le code suivant construit de nouveaux objets Number.

```
n1 = new Number(3.4);
n2 = new Number(-10);
```

Consultez également

Number

Number.MAX_VALUE

Disponibilité

Flash Player 5.

Usage

```
Number.MAX_VALUE
```

Description

Propriété : le plus grand nombre représentable (IEEE-754 double précision). Ce nombre est approximativement 1.79E+308.

Number.MIN_VALUE

Disponibilité

Flash Player 5.

Usage

```
Number.MIN_VALUE
```

Description

Propriété : le nombre représentable le plus petit (IEEE-754 double précision). Ce nombre est approximativement 5e-324.

Number.NaN

Disponibilité

Flash Player 5.

Usage

```
Number.NaN
```

Description

Propriété : la valeur IEEE-754 représentant Not A Number (NaN).

Consultez également

isNaN(), NaN

Number.NEGATIVE_INFINITY

Disponibilité

Flash Player 5.

Usage

```
Number.NEGATIVE_INFINITY
```

Description

Propriété : 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 qui est renvoyée lorsqu'une opération mathématique ou une fonction renvoient une valeur négative trop grande pour être représentée.

Number.POSITIVE_INFINITY

Disponibilité

Flash Player 5.

Usage

```
Number.POSITIVE_INFINITY
```

Description

Propriété : 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 qui est renvoyée lorsqu'une opération mathématique ou une fonction renvoient une valeur trop grande pour être représentée.

Number.toString()

Disponibilité

Flash Player 5 ; comportement mis à jour dans Flash Player 7.

Usage

```
monNombre.toString(base)
```

Paramètres

base Spécifie la base numérique (entre 2 et 36) à utiliser pour la conversion d'un nombre en chaîne. Si vous ne spécifiez pas le paramètre *base*, la valeur par défaut est 10.

Renvoie

Une chaîne.

Description

Méthode : renvoie la représentation chaîne de l'objet Number spécifié (*monNombre*).

Si *monNombre* est undefined, la valeur renvoyée est la suivante :

- Dans les fichiers publiés pour Flash Player 6 ou les versions antérieures, le résultat est 0.
- Dans les fichiers publiés pour Flash Player 7 ou les versions ultérieures, le résultat est NaN.

Exemple

L'exemple suivant utilise les valeurs 2 et 8 pour le paramètre *radix*, et renvoie une chaîne contenant la représentation correspondante du chiffre 9.

```
monNombre = nouveau chiffre (9);
trace(monNombre.toString(2)); / 1001
trace(monNombre.toString(8)); / 11
```

Consultez également

NaN

Number.valueOf

Disponibilité

Flash Player 5.

Usage

```
monNombre.valueOf()
```

Paramètres

Aucun.

Renvoie

Nombre.

Description

Méthode : renvoie le type de valeur primitif de l'objet Number spécifié.

Number

Disponibilité

Flash Player 4 ; comportement modifié dans Flash Player 7.

Usage

`Number(expression)`

Paramètres

`expression` Une expression à convertir en nombre.

Renvoie

Un nombre ou `NaN`.

Description

Fonction : convertit le paramètre *`expression`* en nombre et renvoie une valeur comme suit :

- 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`* comme un nombre décimal avec un exposant facultatif (1.57505e-3).
- Si *`expression`* est `undefined`, la valeur renvoyée est la suivante :
  - Dans les fichiers publiés pour Flash Player 6 ou les versions antérieures, le résultat est 0.
  - Dans les fichiers publiés pour Flash Player 7 ou les versions ultérieures, le résultat est `NaN`.

Cette fonction est utilisée pour convertir les fichiers Flash 4 contenant des opérateurs déconseillés importés dans l'environnement auteur de Flash 5 ou d'une version ultérieure. Pour plus d'informations, consultez `& (Opérateur AND au niveau du bit)`.

Consultez également

`NaN`, Classe Number

Classe Object

Disponibilité

Flash Player 5 (est devenu un objet natif dans Flash Player 6, améliorant ainsi les performances de manière significative).

Description

La classe Object est à la base de la hiérarchie des classes ActionScript. Cette classe contient un petit sous-ensemble des fonctionnalités fournies par la classe JavaScript Object.

Méthodes de la classe Objet

Méthode	Description
`Object.addProperty()`	Crée une propriété de lecture/définition pour un objet.
`Object.registerClass()`	Associe un symbole de clip à une classe d'objet ActionScript.
`Object.toString()`	Convertit l'objet spécifié en chaîne et le renvoie.
`Object.unwatch()`	Supprime le point de surveillance que `Object.watch()` a créé.
`Object.valueOf()`	Renvoie la valeur primitive d'un objet.
`Object.watch()`	Enregistre un gestionnaire d'événements à appeler lorsqu'une propriété spécifique ou un objet ActionScript sont modifiés.

Propriétés de la classe Objet

Propriété	Description
`Object.__proto__`	Une référence à la propriété `prototype` de la fonction constructeur de l'objet.

Constructeur de la classe Object

Disponibilité

Flash Player 5.

Usage

```
new Object([valeur])
```

Paramètres

valeur Un nombre, une valeur booléenne ou une chaîne à convertir en objet. Ce paramètre est facultatif. Si vous ne spécifiez pas *valeur*, le constructeur crée un nouvel objet sans propriétés définies.

Renvoie

Rien.

Description

Constructeur : crée un nouvel objet Object.

Object.addProperty()

Disponibilité

Flash Player 6. Dans les fichiers de classe externes, vous pouvez utiliser `get` ou `set` plutôt que cette méthode.

Usage

```
monObjet.addProperty( prop, getFunc, setFunc )
```

Paramètres

prop Le nom de la propriété d'objet à créer.

getFunc La fonction qui est invoquée pour récupérer la valeur de la propriété ; ce paramètre est un objet fonction.

setFunc La fonction qui est invoquée pour définir la valeur de la propriété ; ce paramètre est un objet fonction. Si vous transmettez la valeur `null` pour ce paramètre, la propriété est en lecture seule.

Renvoie

Renvoie une valeur de `true` si la propriété est correctement créée ; sinon, renvoie `false`.

Description

Méthode : crée une propriété de lecture/définition. Lorsque Flash lit une propriété de lecture/définition, il invoque la fonction get et la valeur renvoyée par la fonction devient une valeur de *prop*. Lorsque Flash écrit une propriété de lecture/définition, il invoque la fonction set et lui transmet la nouvelle valeur en tant que paramètre. Si une propriété portant le nom donné existe déjà, la nouvelle propriété la remplace.

Une fonction « get » est une fonction sans paramètre. La valeur renvoyée peut être de tout 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 « set » est une fonction qui accepte un paramètre, qui est la nouvelle valeur de la propriété. Par exemple, si la propriété `x` est affectée par l'instruction `x = 1`, le paramètre `1` de type nombre est transmis à la fonction set. La valeur renvoyée par la fonction set est ignorée.

Vous pouvez ajouter des propriété de lecture/définition à des objets prototypes. Si vous ajoutez une propriété de lecture/définition à un objet prototype, 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 endroit, 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 d'un objet prototype hérité, la référence transmise à la fonction de lecture/définition sera l'objet auquel il est fait référence à l'origine, et non l'objet prototype.

En cas d'invocation incorrecte, `Object.addProperty()` peut échouer et provoquer une erreur. Le tableau suivant décrit les erreurs qui peuvent se produire :

Condition d'erreur	Ce qui se produit
prop n'est pas un nom de propriété valide ; par exemple, une chaîne vide.	Renvoie `false` et la propriété n'est pas ajoutée.
getFunc n'est pas un objet de fonction valide.	Renvoie `false` et la propriété n'est pas ajoutée.
setFunc n'est pas un objet de fonction valide.	Renvoie `false` et la propriété n'est pas ajoutée.

Exemple

Usage 1 : un objet a deux méthodes internes, `setQuantity()` et `getQuantity()`. Une propriété, `bookcount`, peut être utilisée pour invoquer ces méthodes lorsqu'elle est définie ou récupérée. Une troisième méthode interne, `getTitle()`, renvoie une valeur en lecture seule associée à la propriété `bookname` :

```
function Book() {
  this.setQuantity = function(numBooks) {
    this.books = numBooks;
  }
  this.getQuantity = function() {
    return this.books;
  }
  this.getTitle = function() {
    return "L'attrape-coeurs";
  }
  this.addProperty("bookcount", this.getQuantity, this.setQuantity);
  this.addProperty("bookname", this.getTitle, null);
}
monLivre = new Book();
monLivre.bookcount = 5;
commande = "Vous avez commandé " + monLivre.bookcount + " exemplaires de " +
  monLivre.bookname;
```

Lorsqu'un script récupère la valeur de `monLivre.bookcount`, l'interprète d'ActionScript invoque automatiquement `monLivre.getQuantity()`. Lorsqu'un script modifie la valeur de `monLivre.bookcount`, l'interprète invoque `monObjet.setQuantity()`. La propriété `bookname` ne spécifie pas une fonction set et les tentatives de modification de `bookname` sont donc ignorées.

Usage 2 : l'exemple de `bookcount` et `bookname` décrit ci-dessus fonctionne, mais les propriétés `bookcount` et `bookname` sont ajoutées à chaque occurrence de l'objet `Book`. Cela signifie que le coût d'utilisation des propriétés est de deux emplacements de propriété pour chaque occurrence de l'objet. Un trop grand nombre de propriétés comme `bookcount` et `bookname` dans une classe pourrait entraîner une forte utilisation de mémoire. Pour éviter ce problème, vous pouvez ajouter des propriétés à `Book.prototype` :

```
function Book () {}
Book.prototype.setQuantity = function(numBooks) {
  this.books = numBooks;
}
Book.prototype.getQuantity = function() {
  return this.books;
}
Book.prototype.getTitle = function() {
  return "L'attrape-coeurs";
}
Book.prototype.addProperty("bookcount", Book.prototype.getQuantity,
  Book.prototype.setQuantity);
Book.prototype.addProperty("bookname", Book.prototype.getTitle, null);
monLivre = new Book();
monLivre.bookcount = 5;
commande = "Vous avez commandé "+monLivre.bookcount+" exemplaires
  de "+monLivre.bookname;
```

Maintenant, les propriétés `bookcount` et `bookname` se trouvent dans un endroit unique : l'objet `Book.prototype`. Cependant, le résultat est le même que pour le code indiqué dans Utilisation 1, qui a ajouté directement `bookcount` et `bookname` à chaque occurrence. Si `bookcount` ou `bookname` est obtenu dans une occurrence `Book`, la chaîne prototype est remontée et la propriété de lecture/définition de `Book.prototype` est détectée.

Usage 3 : les propriétés intégrées `TextField.scroll` et `TextField.maxscroll` sont des propriétés de lecture/définition. L'objet TextField a les méthodes internes `getScroll()`, `setScroll()` et `getMaxScroll()`. Le constructeur TextField crée les propriétés de lecture/définition et les oriente vers les méthodes de lecture/définition internes, comme dans l'exemple suivant :

```
this.addProperty("scroll", this.getScroll, this.setScroll);
this.addProperty("maxscroll", this.getMaxScroll, null);
```

Lorsqu'un script récupère la valeur de `monChampDeTexte.scroll`, l'interprète d'ActionScript invoque automatiquement `monChampDeTexte.getScroll()`. Lorsqu'un script modifie la valeur de `monChampDeTexte.scroll`, l'interprète invoque `monChampDeTexte.setScroll()`. La propriété `maxscroll` ne spécifie pas une fonction set et les tentatives de modification de `maxscroll` sont donc ignorées.

Usage 4 : même si les propriétés intégrées `TextField.scroll` et `TextField.maxscroll` fonctionnent dans l'exemple Usage 3, les propriétés `scroll` et `maxscroll` sont ajoutées à chaque occurrence de l'objet TextField. Cela signifie que le coût d'utilisation des propriétés est de deux emplacements de propriété pour chaque occurrence de l'objet. Un trop grand nombre de propriétés comme `scroll` et `maxscroll` dans une classe pourrait entraîner une forte consommation de mémoire. Au lieu de cela, vous pouvez ajouter les propriétés `scroll` et `maxscroll` à `TextField.prototype` :

```
TextField.prototype.addProperty("scroll", this.getScroll, this.setScroll);
TextField.prototype.addProperty("maxscroll", this.getMaxScroll, null);
```

Les propriétés `scroll` et `maxscroll` existent maintenant seulement en un endroit : l'objet `TextField.prototype`. Cependant, l'effet est le même que celui du code ci-dessus qui ajoutait `scroll` et `maxscroll` directement à chaque occurrence. Si `scroll` ou `maxscroll` est obtenu dans une occurrence TextField, la chaîne de prototype est remontée et la propriété de lecture/définition de `TextField.prototype` est détectée.

Object.__proto__

Disponibilité

Flash Player 5.

Usage

monObjet.__proto__

Paramètres

Aucun.

Description

Propriété : fait référence à la propriété `prototype` de la fonction constructeur qui a créé *monObjet*. La propriété __proto__ est automatiquement affectée à tous les objets lors de leur création. L'interprète d'ActionScript utilise la propriété __proto__ pour accéder à la propriété `prototype` de la fonction constructeur de l'objet, afin de déterminer quelles propriétés et méthodes l'objet hérite de sa classe.

Object.registerClass()

Disponibilité

Flash Player 6. Si vous utilisez des fichiers de classe externes, vous pouvez utiliser le champ de classe ActionScript 2.0 de la boîte de dialogue Propriétés de liaison ou Propriétés du symbole pour associer un objet à une classe, plutôt que d'utiliser cette méthode.

Usage

Object.registerClass(*idDeSymbole*, *laClasse*)

Paramètres

idDeSymbole L'identifiant de liaison du symbole de clip ou l'identifiant de chaîne pour la classe ActionScript.

laClasse Une référence à la fonction constructeur de la classe ActionScript ou `null` pour désenregistrer le symbole.

Renvoie

Si l'enregistrement de classe réussit, une valeur de `true` est renvoyée ; sinon, `false` est renvoyé.

Description

Méthode : associe un symbole de clip à une classe d'objet ActionScript. S'il n'existe aucun symbole, Flash crée une association entre un identifiant de chaîne et une classe d'objet.

Lorsqu'une occurrence du symbole de clip spécifié est placée par le scénario, elle est enregistrée dans la classe spécifiée par le paramètre *laClasse* plutôt que dans la classe MovieClip.

Lorsqu'une occurrence du symbole de clip spécifié est créée à l'aide de `MovieClip.attachMovie()` ou `MovieClip.duplicateMovieClip`, elle est enregistrée dans la classe spécifiée par le paramètre *`laClasse`* plutôt que dans la classe MovieClip. Si *`laClasse`* est `null`, cette méthode supprime toute définition de classe ActionScript associée au symbole de clip ou à l'identifiant de classe spécifié. Pour les symboles de clip, toute occurrence existante du clip reste inchangée, mais les nouvelles occurrences du symbole sont associées à la classe MovieClip par défaut.

Si un symbole est déjà enregistré dans une classe, cette méthode le remplace par le nouvel enregistrement.

Lorsqu'une occurrence de clip est placée par le scénario ou créée à l'aide de `attachMovie()` ou `duplicateMovieClip()`, ActionScript invoque le constructeur pour la classe appropriée à l'aide du mot-clé `this` pointant vers l'objet. La fonction constructeur est invoquée sans paramètres.

Si vous utilisez cette méthode pour enregistrer un clip dans une classe ActionScript autre que MovieClip, le symbole du clip n'hérite pas des méthodes, des propriétés et des événements de la classe MovieClip intégrée, sauf si vous n'incluez la classe MovieClip dans la chaîne prototype de la nouvelle classe. Le code suivant crée une nouvelle classe ActionScript appelée `laClasse`, qui hérite des propriétés de la classe MovieClip :

```
laClasse.prototype = new MovieClip();
```

Consultez également

`MovieClip.attachMovie()`, `MovieClip.duplicateMovieClip`

Object.toString()

Disponibilité

Flash Player 5.

Usage

`monObjet``.toString()`

Paramètres

Aucun.

Renvoie

Une chaîne.

Description

Méthode : convertit l'objet spécifié en chaîne et le renvoie.

Object.unwatch()

Disponibilité

Flash Player 6.

Usage

```
monObjet.unwatch (prop)
```

Paramètres

prop Le nom de la propriété d'objet qui ne doit plus être surveillée, comme une chaîne.

Renvoie

Une valeur booléenne.

Description

Méthode : supprime un point de surveillance que `Object.watch()` a créé. Cette méthode renvoie une valeur de `true` si le point de surveillance a été correctement retiré ; sinon, elle renvoie une valeur de `false`.

Object.valueOf()

Disponibilité

Flash Player 5.

Usage

```
monObjet.valueOf()
```

Paramètres

Aucun.

Renvoie

La valeur primitive de l'objet spécifié ou l'objet lui-même.

Description

Méthode : renvoie la valeur primitive de l'objet spécifié. Si l'objet ne possède pas de valeur primitive, il est lui-même renvoyé.

Object.watch()

Disponibilité

Flash Player 6.

Usage

monObjet.watch(*prop*, *rappel* [, *donnéesUtilisateur*])

Paramètres

prop Une chaîne indiquant le nom de la propriété d'objet à surveiller.

rappel La fonction à invoquer lorsque que la propriété surveillée change. Ce paramètre est un objet de fonction, pas un nom de fonction sous forme de chaîne. La forme de `callback` est `callback(prop, ancienneVal, nouvelleVal, donnéesUtilisateur)`.

donnéesUtilisateur Une donnée ActionScript arbitraire qui est transmise à la méthode *de rappel*. Si le paramètre *donnéesUtilisateur* est omis, `undefined` est transmis à la méthode de rappel. Ce paramètre est facultatif.

Renvoie

Une valeur `true` si le point de surveillance est correctement créé ; sinon, renvoie une valeur `false`.

Description

Enregistre un gestionnaire d'événements à appeler lorsqu'une propriété spécifique ou un objet ActionScript est modifié(e). Lorsque la propriété change, la fonction de rappel est invoquée avec `monObjet` comme objet contenant. Vous devez renvoyer la nouvelle valeur depuis la méthode `object.watch` ou bien une valeur `undefined` est affectée à la propriété d'objet surveillée.

Un point de surveillance peut filtrer (ou annuler) l'affectation de valeur, en renvoyant une `nouvelleVal` modifiée (ou `ancienneVal`). Si vous supprimez une propriété pour laquelle un point de surveillance a été défini, ce point de surveillance ne disparaît pas. Si vous recréez ultérieurement la propriété, le point de surveillance est toujours en vigueur. Pour supprimer un point de surveillance, utilisez la méthode `Object.unwatch`.

Il ne peut exister qu'un seul point de surveillance enregistré sur une propriété. Les appels ultérieurs de `Object.watch()` sur la même propriété remplacent le point de surveillance original.

La méthode `Object.watch()` se comporte de façon similaire à la fonction `Object.watch()` de Netscape JavaScript version 1.2 et ultérieures. La principale différence est le paramètre *donnéesUtilisateur*, qui est un ajout Flash à `Object.watch()` non supporté par Netscape Navigator. Vous pouvez passer le paramètre *donnéesUtilisateur* dans le gestionnaire d'événements et l'utiliser dans ce dernier.

La méthode `Object.watch()` ne peut pas surveiller les propriétés de lecture/définition. Les propriétés de lecture/définition fonctionnent par « évaluation passive » – la valeur de la propriété n'étant pas déterminée tant que la propriété n'est pas interrogée. Une « évaluation passive » est souvent rentable étant donné que la propriété n'est pas constamment mise à jour, mais simplement évaluée lorsqu'il le faut. Cependant, `Object.watch()` a besoin d'évaluer une propriété pour activer des point de surveillance sur celle-ci. Pour fonctionner avec une propriété de lecture/définition, `Object.watch()` a besoin d'évaluer constamment la propriété, ce qui n'est pas rentable.

En général, les propriétés ActionScript prédéfinies, telles que _x, _y, _width et _height, sont des propriétés de lecture/définition et ne peuvent donc pas être surveillées avec la méthode Object.watch().

Consultez également

Object.addProperty(), Object.unwatch()

Objet()

Disponibilité

Flash Player 5.

Usage

```
Objet( [ valeur ] )
```

Paramètres

valeur Un nombre, une chaîne ou une valeur booléenne.

Renvoie

Un objet.

Description

Fonction de conversion : crée un nouvel objet vide, ou convertit le nombre, la chaîne ou la valeur booléenne spécifiés en un objet. Cette commande est équivalente à la création d'un objet à l'aide du constructeur Object (consultez *Constructeur de la classe Object*, page 651).

on()

Disponibilité

Flash 2. Tous les événements ne sont pas supportés dans Flash 2.

Usage

```
on(événementSouris) {
  // vos instructions
}
```

Paramètres

instruction(s) Les instructions à exécuter lorsque *événementSouris* survient.

Un *événementSouris* est un déclenchement appelé « événement ». Lorsque l'événement a lieu, les instructions suivantes dans les accolades sont exécutées. N'importe laquelle des valeurs suivantes peut être spécifiée pour le paramètre *événementSouris :*

- `press` Le bouton de la souris est enfoncé alors que le pointeur se trouve au-dessus du bouton.
- `release` Le bouton de la souris est relâché alors que le pointeur se trouve au-dessus du bouton.
- `releaseOutside` Le bouton de la souris est relâché alors que le pointeur se trouve en dehors du bouton après l'enfoncement du bouton pendant que le pointeur est à l'intérieur du bouton.
- `rollOut` Le pointeur passe en dehors de la zone du bouton.
- `rollOver` Le pointeur de la souris passe au-dessus du bouton.
- `dragOut` Alors que le pointeur se trouve au-dessus du bouton, le bouton de la souris est enfoncé, puis le pointeur sort de la zone du bouton.
- `dragOver` Alors que le pointeur se trouve au-dessus du bouton, le bouton de la souris a été enfoncé, puis le pointeur sort du bouton et est ramené au-dessus du bouton.
- `keyPress ("touche")` La touche spécifiée est enfoncée. Spécifiez un code de touche ou une constante de touche pour la portion touche du paramètre. Pour obtenir la liste des codes de touche associés aux touches d'un clavier standard, consultez l'Annexe C, *Touches du clavier et valeurs de code correspondantes*, page 901 ; pour obtenir la liste des constantes, consultez *Propriétés de la classe Key*, page 453.

Description

Gestionnaire d'événement : spécifie l'événement souris ou touche qui déclenche une action.

Exemple

Dans le script suivant, l'action `startDrag()` est exécutée lorsque l'utilisateur clique avec la souris et le script conditionnel est exécuté lorsque le bouton et l'objet sont relâchés :

```
on(press) {
  startDrag("lapin");
}
on(release) {
  trace(_root.lapin._y);
  trace(_root.lapin._x);
  stopDrag();
}
```

Consultez également

`onClipEvent()`

onClipEvent()

Disponibilité

Flash Player 5.

Usage

```
onClipEvent(événementAnimation){
  // vos instructions
}
```

Paramètres

Un *`événementAnimation`* est un déclenchement appelé *événement*. Lorsque l'événement a lieu, les instructions suivantes dans les accolades sont exécutées. N'importe laquelle des valeurs suivantes peut être spécifiée pour le paramètre *`événementAnimation`* :

- `load` L'action est initiée dès que le clip est instancié et apparaît dans le scénario.
- `unload` L'action est initiée dans la première image après la suppression du clip du scénario. Les actions associées à l'événement de clip `Unload` sont traitées avant que des actions ne soient associées à l'image affectée.
- `enterFrame` L'action est déclenchée continuellement à la cadence du clip. Les actions associées à l'événement de clip `enterFrame` sont traitées avant les actions associées aux images affectées.
- `mouseMove` L'action est initiée chaque fois que la souris est déplacée. Utilisez les propriétés `_xmouse` et `_ymouse` pour déterminer la position actuelle de la souris.
- `mouseDown` L'action est initiée lorsque le bouton gauche de la souris est enfoncé.
- `mouseUp` L'action est initiée lorsque le bouton gauche de la souris est relâché.
- `keyDown` L'action est initiée lorsqu'une touche est enfoncée. Utilisez `Key.getCode()` pour obtenir des informations sur la dernière touche enfoncée.
- `keyUp` L'action est initiée lorsqu'une touche est relâchée. Utilisez la méthode `Key.getCode()` pour obtenir des informations sur la dernière touche enfoncée.

- `data` L'action est initiée lorsque des données sont reçues dans une action `loadVariables()` ou `loadMovie()`. Lorsqu'il est spécifié avec une action `loadVariables()`, l'événement `data` ne survient qu'une seule fois, quand la dernière variable est chargée. Lorsqu'il est spécifié avec une action `loadMovie`, l'événement `data` est répété plusieurs fois, au fur et à mesure que les sections de données sont récupérées.

Description

Gestionnaire d'événement : déclenche des actions définies pour une occurrence de clip spécifique.

Exemple

L'instruction suivante inclut le script provenant d'un fichier externe lorsque le fichier SWF est exporté ; les actions du script inclus sont exécutées au chargement du clip auquel elles sont associées.

```
onClipEvent(load) {
  #include "monScript.as"
}
```

L'exemple suivant utilise `onClipEvent()` avec l'événement de clip `keyDown`. L'événement d'animation `keyDown` est généralement utilisé en conjonction avec une ou plusieurs méthodes et propriétés associées à l'objet Key. Le script ci-dessous utilise `Key.getCode()` pour déterminer la touche enfoncée par l'utilisateur : si la touche enfoncée correspond à la propriété `Key.RIGHT`, l'animation est envoyée vers l'image suivante ; si la touche enfoncée correspond à la propriété `Key.LEFT`, l'animation est envoyée vers l'image précédente.

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

L'exemple suivant utilise `onClipEvent()` avec l'événement d'animation `mouseMove`. Les propriétés `_xmouse` et `_ymouse` suivent la position de la souris.

```
onClipEvent(mouseMove) {
  stageX=_root._xmouse;
  stageY=_root._ymouse;
}
```

Consultez également

Classe Key, `MovieClip._xmouse`, `MovieClip._ymouse`, `on()`, `updateAfterEvent()`

onUpdate

Disponibilité

Flash Player 6.

Usage

```
function onUpdate() {
   ...instructions...;
}
```

Paramètres

Aucun.

Renvoie

Rien.

Description

Gestionnaire d'événements : `onUpdate` est défini pour une vidéo Live Preview utilisée avec un composant. Lorsque l'occurrence d'un composant sur la scène a une animation Aperçu en direct, le programme auteur invoque la fonction `onUpdate` de l'animation Aperçu en direct à chaque modification des paramètres d'une occurrence de composant. La fonction `onUpdate` est invoquée par le programme auteur sans paramètre et sa valeur renvoyée est ignorée. La fonction `onUpdate` doit être déclarée dans le scénario principal de l'animation Aperçu en direct.

La définition d'une fonction `onUpdate` dans une animation Aperçu en direct est facultative.

Pour plus d'informations sur les animations Aperçu en direct, consultez le guide *Utilisation des composants*.

Exemple

La fonction `onUpdate` permet à l'animation Aperçu en direct de mettre à jour son aspect visuel pour correspondre aux nouvelles valeurs des paramètres de composant. Lorsque l'utilisateur modifie la valeur d'un paramètre dans l'inspecteur de propriétés des composants ou dans le panneau Paramètres de composant, `onUpdate` est invoqué. La fonction `onUpdate` exécute également sa propre mise à jour. Par exemple, si le composant inclut un paramètre `couleur`, la fonction `onUpdate` peut altérer la couleur d'un clip dans l'Aperçu en direct pour refléter la nouvelle valeur du paramètre. En outre, elle peut stocker la nouvelle couleur dans une variable interne.

Voici un exemple d'utilisation de la fonction `onUpdate` pour transmettre les valeurs de paramètre au travers d'un clip vide dans l'animation Aperçu en direct. Supposons que vous avez un composant de bouton étiqueté avec une variable `couleurEtiquette`, qui spécifie la couleur de l'étiquette de texte. Le code suivant se trouve dans la première image du scénario principal de l'animation de composant :

```
//Définissez la variable du paramètre textColor pour spécifier la couleur du
  texte de l'étiquette du bouton.
buttonLabel.textColor = couleurEtiquette;
```

Dans l'animation Aperçu en direct, placez un clip vide nommé « xch » dans l'animation Aperçu en direct. Placez ensuite le code suivant dans la première image de l'animation Aperçu en direct. Ajoutez « xch » au chemin de la variable `couleurEtiquette`, pour passer la variable dans le clip `mon_mc` :

```
//Ecrivez une fonction onUpdate, en ajoutant "mon_mc." aux noms des variables
  de paramètre :
function onUpdate (){
  buttonLabel.textColor = mon_mc.couleurEtiquette;
}
```

or

Disponibilité

Flash 4. Cet opérateur est déconseillé et remplacé par l'opérateur `|| (OR logique)`.

Usage

condition1 `or` *condition2*

Paramètres

condition1,2 Une expression qui détermine `true` ou `false`.

Renvoie

Rien.

Description

Opérateur : évalue *condition1* et *condition2* et, si une des expressions est `true`, l'expression entière est `true`.

Consultez également

`|| (OR logique)`, `| (OR au niveau du bit)`

ord

Disponibilité

Flash Player 4. Cette fonction est abandonnée et remplacée par les méthodes et propriétés de la classe String.

Usage

```
ord(caractère)
```

Paramètres

caractère Le caractère à convertir en code ASCII.

Renvoie

Rien.

Description

Fonction de chaîne : convertit des caractères en code ASCII.

Consultez également

Classe String

_parent

Disponibilité

Flash Player 5.

Usage

```
_parent.propriété
_parent._parent.propriété
```

Description

Identifiant : spécifie ou renvoie une référence au clip ou objet contenant le clip ou objet courant. L'objet courant est l'objet contenant le code ActionScript faisant référence à `_parent`. Utilisez `_parent` pour spécifier un chemin relatif aux clips ou objets qui se trouvent au-dessus du clip ou objet actuel.

Exemple

Dans l'exemple suivant, le clip `desk` est un enfant du clip `salleDeClasse`. Lorsque le script ci-dessous est exécuté dans le clip `desk`, la tête de lecture atteint l'image 10 du scénario du clip `salleDeClasse`.

```
_parent.gotoAndStop(10);
```

Consultez également

`_root`, `targetPath`

parseFloat()

Disponibilité

Flash Player 5.

Usage

```
parseFloat(chaîne)
```

Paramètres

chaîne La chaîne à lire et à convertir en nombre à virgule flottante.

Renvoie

Un nombre ou `NaN`.

Description

Fonction : convertit une chaîne en nombre à virgule flottante. La fonction analyse et renvoie les nombres d'une chaîne jusqu'à ce qu'elle atteigne un caractère qui n'appartient pas au nombre initial. Si la chaîne ne commence pas par un nombre qui peut être analysé, `parseFloat` renvoie `NaN`. Les espaces précédant les entiers valides sont ignorés, tout comme les caractères non numériques à droite.

Exemple

Les exemples suivants utilisent la fonction `parseFloat` pour évaluer différents types de nombres.

`parseFloat("-2")` renvoie `-2`

`parseFloat("2.5")` renvoie `2.5`

`parseFloat("3.5e6")` renvoie `3.5e6` ou `3500000`

`parseFloat("foobar")` renvoie `NaN`

`parseFloat(" 5.1")` renvoie `5.1`

`parseFloat("3.75math")` renvoie `3.75`

`parseFloat("0tasDeTrucs")` renvoie `0`

Consultez également

`NaN`

parseInt

Disponibilité

Flash Player 5.

Usage

```
parseInt(expression, [radix])
```

Paramètres

expression Une chaîne à convertir en entier.

radix Optionnel : un entier représentant le radical (base) du nombre à analyser. Les valeurs légales sont comprises entre 2 et 36.

Renvoie

Un nombre ou `NaN`.

Description

Fonction : convertit une chaîne en entier. Si la chaîne spécifiée dans les paramètres ne peut pas être convertie en un nombre, la fonction renvoie `NaN`. Les chaînes commençant par 0x sont interprétées comme des nombres hexadécimaux. Les entiers commençant par 0 ou spécifiant une base de 8 sont interprétés comme des nombres octaux. Les espaces précédant les entiers valides sont ignorés, tout comme les caractères non numériques à droite.

Exemple

Les exemples suivants utilisent la fonction `parseInt` pour évaluer différents types de nombres.

```
parseInt("3.5")
// renvoie 3

parseInt("chose")
// renvoie NaN

parseInt("4foo")
// renvoie 4
```

Les exemples suivants illustrent les conversions hexadécimales :

```
parseInt("0x3F8")
// renvoie 1016

parseInt("3E8", 16)
// renvoie 1000
```

L'exemple suivant présente une conversion binaire :

```
parseInt("1010", 2)
// renvoie 10 (la représentation décimale du binaire 1010)
```

Les exemples suivants présentent une analyse de nombres octaux :

```
parseInt("0777")
parseInt("777", 8)
// renvoie 511 (la représentation décimale de l'octal 777)
```

play()

Disponibilité

Flash 2.

Usage

```
play()
```

Paramètres

Aucun.

Renvoie

Rien.

Description

Fonction : fait avancer la tête de lecture dans le scénario.

Exemple

Le code suivant utilise une instruction `if` pour vérifier la valeur d'un nom entré par l'utilisateur. Si l'utilisateur entre `Steve`, l'action `play()` est appelée et la tête de lecture avance dans le scénario. Si l'utilisateur entre autre chose que `Steve`, le fichier SWF n'est pas lu et un champ de texte contenant le nom de variable `alert` est affiché.

```
stop();
if (nom == "Steve") {
  play();
} else {
  alert="Vous n'êtes pas Steve !";
}
```

prevFrame()

Disponibilité

Flash 2.

Usage

```
prevFrame()
```

Paramètres

Aucun.

Renvoie

Rien.

Description

Fonction : envoie la tête de lecture vers l'image précédente et l'arrête. Si l'image actuelle est l'image 1, la tête de lecture ne bouge pas.

Exemple

Lorsque l'utilisateur clique sur un bouton auquel le gestionnaire suivant est affecté, la tête de lecture est envoyée à l'image précédente.

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

Consultez également

MovieClip.prevFrame()

prevScene()

Disponibilité

Flash 2.

Usage

```
prevScene()
```

Paramètres

Aucun.

Renvoie

Rien.

Description

Fonction : envoie la tête de lecture vers l'image 1 de la séquence précédente et l'arrête.

Consultez également

nextScene()

print()

Disponibilité

Flash Player 4.20.

Remarque : Si vous utilisez Flash Player7 ou ultérieur, vous pouvez créer un objet PrintJob, qui vous permet (ainsi que l'utilisateur) de mieux contrôler le processus d'impression. Pour plus d'informations, consultez l'entrée Classe PrintJob.

Usage

```
print("cible", "régionDimpression")
```

Paramètres

cible Le nom d'occurrence d'un clip à imprimer. Par défaut, toutes les images de l'occurrence cible sont imprimées. Affectez une étiquette #p aux images que vous souhaitez imprimer dans ce clip.

régionDimpression Un modificateur qui définit la zone d'impression du clip. Mettez ce paramètre entre guillemets et spécifiez l'une des valeurs suivantes :

- `bmovie` Désigne le cadre de délimitation d'une image spécifique dans une animation comme zone d'impression de toutes les images imprimables de l'animation. Affectez une étiquette d'image #b à l'image dont vous souhaitez utiliser le cadre de délimitation comme région d'impression.
- `bmax` Désigne un composite de tous les cadres de délimitation de l'ensemble des images imprimables comme zone d'impression. Spécifiez le paramètre `bmax` lorsque la taille des images de votre animation varie.
- `bframe` Indique que le cadre de délimitation de chaque image imprimable est utilisé comme zone d'impression pour cette image. Cela modifie la région d'impression pour chaque image et modifie la taille des objets pour qu'ils correspondent à la région d'impression. Utilisez `bframe` si vous avez des objets de tailles diverses dans chaque image et si vous souhaitez que chaque objet remplisse la page imprimée.

Renvoie

Rien.

Description

Fonction : imprime le clip *cible* en fonction des limites spécifiées dans le paramètre (`bmovie`, `bmax` ou `bframe`). Affectez une étiquette #p aux images que vous souhaitez imprimer dans le clip cible. Bien que `print()` fournisse des impressions de meilleure qualité que `printAsBitmap()`, elle ne peut pas être utilisée pour imprimer des clips utilisant les transparences alpha ou des effets de couleur spéciaux.

Si vous utilisez `bmovie` pour le paramètre *régionDimpression*, mais que vous n'affectez pas d'étiquette #b à une image, la zone d'impression est déterminée par la taille de scène de la vidéo chargée. L'animation n'hérite pas de la taille de scène de l'animation principale.

Tous les éléments imprimables d'une animation doivent être complètement chargés avant de commencer l'impression.

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

Exemple

L'exemple suivant imprime toutes les images imprimables du clip `mon_mc` selon la zone d'impression définie par le cadre de délimitation de l'image portant l'étiquette d'image `#b` :

```
print(mon_mc,"bmovie");
```

L'exemple suivant imprime toutes les images imprimables de `mon_mc` avec une zone d'impression définie par le cadre de délimitation de chaque image :

```
print(mon_mc,"bframe");
```

Consultez également

`printAsBitmap()`, `printAsBitmapNum`, Classe PrintJob, `printNum()`

printAsBitmap()

Disponibilité

Flash Player 4.20.

Remarque : Si vous utilisez Flash Player7 ou ultérieur, vous pouvez créer un objet PrintJob, qui vous permet (ainsi que l'utilisateur) de mieux contrôler le processus d'impression. Pour plus d'informations, consultez l'entrée Classe PrintJob.

Usage

```
printAsBitmap(cible, "régionDimpression")
```

Paramètres

`cible` Le nom d'occurrence d'un clip à imprimer. Par défaut, toutes les images de l'animation sont imprimées. Affectez une étiquette `#p` aux images que vous souhaitez imprimer dans cette animation.

`régionDimpression` Un modificateur qui définit la zone d'impression de l'animation. Mettez ce paramètre entre guillemets et spécifiez l'une des valeurs suivantes :

- `bmovie` Désigne le cadre de délimitation d'une image spécifique dans une animation comme zone d'impression de toutes les images imprimables de l'animation. Affectez une étiquette d'image `#b` à l'image dont vous souhaitez utiliser le cadre de délimitation comme région d'impression.
- `bmax` Désigne un composite de tous les cadres de délimitation de l'ensemble des images imprimables comme zone d'impression. Spécifiez le paramètre `bmax` lorsque la taille des images de votre animation varie.
- `bframe` Indique que le cadre de délimitation de chaque image imprimable est utilisé comme zone d'impression pour cette image. Cela modifie la région d'impression pour chaque image et modifie la taille des objets pour qu'ils correspondent à la région d'impression. Utilisez `bframe` si vous avez des objets de tailles diverses dans chaque image et si vous souhaitez que chaque objet remplisse la page imprimée.

Renvoie

Rien.

Description

Fonction : imprime le clip *cible* en tant que bitmap, en fonction des contours spécifiés dans le paramètre (`bmovie`, `bmax`, ou `bframe`). Utilisez `printAsBitmap()` pour imprimer des animations contenant des images avec des objets qui utilisent des transparences ou des effets de couleur. L'action `printAsBitmap()` imprime avec la résolution la plus élevée possible sur l'imprimante pour conserver la définition et la qualité les plus élevées possibles.

Si votre animation ne contient aucune transparence alpha ou effet de couleur, Macromedia vous recommande d'utiliser `print()` pour obtenir des résultats de meilleure qualité.

Si vous utilisez `bmovie` pour le paramètre *régionDimpression*, mais que vous n'affectez pas d'étiquette `#b` à une image, la zone d'impression est déterminée par la taille de scène de la vidéo chargée. L'animation n'hérite pas de la taille de scène de l'animation principale.

Tous les éléments imprimables d'une animation doivent être complètement chargés avant de commencer l'impression.

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

Consultez également

`print()`, `printAsBitmapNum`, Classe PrintJob, `printNum()`

printAsBitmapNum

Disponibilité

Flash Player 5.

Remarque : Si vous utilisez Flash Player7 ou ultérieur, vous pouvez créer un objet PrintJob, qui vous permet (ainsi que l'utilisateur) de mieux contrôler le processus d'impression. Pour plus d'informations, consultez l'entrée Classe PrintJob.

Usage

```
printAsBitmapNum(niveau, "régionDimpression")
```

Paramètres

niveau Le niveau de Flash Player à imprimer. Par défaut, toutes les images du niveau sont imprimées. Affectez une étiquette `#p` aux images que vous souhaitez imprimer dans ce niveau.

régionDimpression Un modificateur qui définit la zone d'impression de l'animation. Mettez ce paramètre entre guillemets et spécifiez l'une des valeurs suivantes :

- `bmovie` Désigne le cadre de délimitation d'une image spécifique dans une animation comme zone d'impression de toutes les images imprimables de l'animation. Affectez une étiquette d'image `#b` à l'image dont vous souhaitez utiliser le cadre de délimitation comme région d'impression.
- `bmax` Désigne un composite de tous les cadres de délimitation de l'ensemble des images imprimables comme zone d'impression. Spécifiez le paramètre `bmax` lorsque la taille des images de votre animation varie.

- `bframe` Indique que le cadre de délimitation de chaque image imprimable est utilisé comme zone d'impression pour cette image. Cela modifie la région d'impression pour chaque image et modifie la taille des objets pour qu'ils correspondent à la région d'impression. Utilisez `bframe` si vous avez des objets de tailles diverses dans chaque image et si vous souhaitez que chaque objet remplisse la page imprimée.

Renvoie

Aucun.

Description

Fonction : imprime un niveau dans Flash Player en tant que bitmap, en fonction des contours spécifiés dans le paramètre (`bmovie`, `bmax`, ou `bframe`). Utilisez `printAsBitmapNum()` pour imprimer des animations contenant des images avec des objets qui utilisent des effets de transparence ou de couleur. L'action `printAsBitmapNum()` imprime dans la résolution la plus élevée sur l'imprimante pour préserver autant de définition et de qualité que possible. Pour calculer la taille de fichier imprimable d'une image à imprimer en bitmap, multipliez la largeur en pixels par la hauteur en pixels par la résolution de l'imprimante.

Si votre animation ne contient pas de transparences alpha ou d'effets de couleur, il est recommandé d'utiliser `printNum()` pour un résultat de meilleure qualité.

Si vous utilisez `bmovie` pour le paramètre *`régionDimpression`*, mais que vous n'affectez pas d'étiquette `#b` à une image, la zone d'impression est déterminée par la taille de scène de la vidéo chargée. L'animation n'hérite pas de la taille de scène de l'animation principale.

Tous les éléments imprimables d'une animation doivent être complètement chargés avant de commencer l'impression.

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

Consultez également

`print()`, `printAsBitmap()`, Classe PrintJob, `printNum()`

Classe PrintJob

Disponibilité

Flash Player 7.

Description

La classe PrintJob vous permet de créer un contenu et de l'imprimer sur une ou plusieurs pages. Cette classe, en plus d'offrir des fonctions d'impression améliorées disponibles avec la méthode `print()`, permet de rendre le contenu dynamique hors écran, d'inviter les utilisateurs à l'aide d'une seule boîte de dialogue d'impression et d'imprimer un document non mis à l'échelle dans des proportions qui correspondent aux proportions du contenu. Cette fonctionnalité est particulièrement utile pour rendre et imprimer un contenu dynamique externe, comme le contenu et le texte dynamique d'une base de données.

En outre, grâce aux propriétés renseignées par `PrintJob.start()`, votre document peut accéder aux paramètres de l'imprimante utilisateur, tels que la hauteur, la largeur et l'orientation de la page, et vous pouvez configurer le document pour qu'il formate dynamiquement le contenu Flash adapté aux paramètres de l'imprimante.

Méthodes de la classe PrintJob

Vous devez utiliser les méthodes de la classe PrintJob dans l'ordre spécifié dans le tableau suivant.

Méthode	Description
`PrintJob.start()`	Affiche les boîtes de dialogue d'impression du système d'exploitation et démarre la mise en attente.
`PrintJob.addPage()`	Ajoute une page au spouleur d'impression.
`PrintJob.send()`	Envoie les pages mises en attente vers l'imprimante.

Constructeur de la classe PrintJob

Disponibilité

Flash Player 7.

Usage

```
ma_pj = new PrintJob()
```

Paramètres

Aucun.

Renvoie

Rien.

Description

Constructeur : crée un objet PrintJob que vous pouvez utiliser pour imprimer une ou plusieurs pages.

Pour implémenter une tâche d'impression, utilisez ces méthodes dans la séquence indiquée :

```
// créer un objet PrintJob
ma_pj = new PrintJob();                          // instancier l'objet

// afficher la boîte de dialogue d'impression
ma_pj.start();                                   // initialiser la tâche
  d'impression

// ajouter la zone spécifiée à la tâche d'impression
// répéter une fois pour chaque page à imprimer
ma_pj.addPage([params]);                         // envoyer la(les) page(s) au
  spouleur
ma_pj.addPage([params]);
ma_pj.addPage([params]);
ma_pj.addPage([params]);

// envoyer des pages du spouleur vers l'imprimante
ma_pj.send();                                     // imprimer page(s)

// nettoyer
delete ma_pj;                                      // supprimer l'objet
```

Dans votre propre implémentation des objets PrintJob, vérifiez les valeurs renvoyées par `PrintJob.start()` et `PrintJob.addPage()` avant de poursuivre l'impression. Consultez les exemples pour `PrintJob.addPage()`.

Vous ne pouvez pas créer d'objet PrintJob tant que tous les objets PrintJob déjà créés n'ont pas été désactivés (à savoir, qu'ils ont réussi ou échoué). Si vous tentez de créer un second objet PrintJob (en appelant `new PrintJob()`) tandis que le premier objet PrintJob est encore actif, le second objet PrintJob n'est pas créé.

Exemple

Pour plus d'informations, consultez `PrintJob.addPage()`.

Consultez également

`PrintJob.addPage()`, `PrintJob.send()`, `PrintJob.start()`

PrintJob.addPage()

Disponibilité

Flash Player 7.

Usage

ma_pj.addPage(*target* [, *printArea*] [, *options*] [, *frameNumber*])

Paramètres

target Le niveau ou le nom de l'occurrence du clip à imprimer. Définissez un nombre pour spécifier un niveau (par exemple, 0 est l'animation _root) ou une chaîne (entre guillemets) pour spécifier le nom de l'occurrence d'un clip.

printArea Un objet facultatif qui spécifie la zone à imprimer, au format suivant :

{xMin:*topLeft*, xMax:*topRight*, yMin:*bottomLeft*, yMax:*bottomRight*}

Les coordonnées que vous spécifiez pour *printArea* représentent les pixels d'écran par rapport au point d'alignement de l'animation _root (si *target*=0) ou du niveau ou clip spécifié par *target*. Vous devez fournir les quatre coordonnées. Les valeurs de largeur (xMax-xMin) et de hauteur (yMax-yMin) doivent chacune être supérieure à 0.

Les points sont des unités de mesure d'impression, et les pixels sont des unités de mesure d'écrans. Un point équivaut à un pixel en termes de taille. Vous pouvez utiliser les équivalences suivantes pour convertir les pouces ou les centimètres en twips, en pixels ou en points (un twip équivaut à 1/20 de pixel) :

- 1 pixel = 1 point = 1/72 pouce = 20 twips
- 1 pouce = 72 pixels = 72 points = 1 440 twips
- 1 cm = 567 twips

Remarque : Si vous avez précédemment utilisé `print()`, `printAsBitmap()`, `printAsBitmapNum` ou `printNum()` pour imprimer depuis Flash, vous avez utilisé une étiquette d'image `#b` pour spécifier la zone à imprimer. Lorsque vous utilisez la méthode `addPage()`, vous devez utiliser le paramètre *printArea* pour spécifier la zone d'impression ; les étiquettes d'image `#b` sont ignorées.

Si vous omettez le paramètre *printArea* ou s'il n'est pas correctement défini, la totalité de la zone *target* est imprimée. Si vous ne souhaitez pas spécifier une valeur pour *printArea* mais voulez en spécifier une pour *options* ou *frameNumber*, définissez `null` pour *printArea*.

options Un paramètre facultatif qui indique si l'impression est effectuée au format vectoriel ou bitmap, de la manière suivante :

```
{printAsBitmap:Boolean}
```

Par défaut, les pages sont imprimées au format vectoriel. Pour imprimer *target* au format bitmap, définissez `true` pour `printAsBitmap`. La valeur par défaut est `false`, ce qui représente une demande d'impression vectorielle. Tenez compte des suggestions suivantes lors de la détermination de la valeur à utiliser :

- Si le contenu en cours d'impression comprend une image bitmap, utilisez `{printAsBitmap:true}` pour inclure les transparences et les effets de couleur.
- Si le contenu ne comprend pas d'image bitmap, omettez ce paramètre ou utilisez `{printAsBitmap:false}` pour imprimer le contenu sous forme de graphiques vectoriels afin de bénéficier d'une qualité d'image élevée.

Si *options* est omis ou défini de manière incorrecte, l'impression vectorielle est implémentée. Si vous ne souhaitez pas spécifier une valeur pour *options* mais souhaitez en spécifier une pour *frameNumber*, définissez `null` pour *options*.

frameNumber Un nombre optionnel qui vous permet de spécifier l'image à imprimer ; notez qu'aucun paramètre ActionScript défini sur l'image n'est appelé. Si vous omettez ce paramètre, l'image actuelle de *target* est imprimée.

Remarque : Si vous avez précédemment utilisé `print()`, `printAsBitmap()`, `printAsBitmapNum` ou `printNum()` pour imprimer depuis Flash, vous avez peut-être utilisé une étiquette d'image `#p` sur plusieurs images pour spécifier les images à imprimer. Pour utiliser `PrintJob.addPage()` afin d'imprimer plusieurs images, vous devez émettre une commande `PrintJob.addPage()` pour chaque image ; les étiquettes d'image `#p` sont ignorées. Pour savoir comment programmer cette opération, consultez l'exemple figurant plus bas dans cette entrée.

Renvoie

Une valeur booléenne `true` si la page a été envoyée avec succès vers le spouleur d'impression, `false` dans les autres cas.

Description

Méthode : envoie le niveau ou le clip spécifié sous la forme d'une page unique au spouleur. Avant d'utiliser cette méthode, vous devez utiliser `PrintJob.start()` ; après avoir appelé `PrintJob.addPage()` une ou plusieurs fois pour une tâche d'impression, vous devez utiliser `PrintJob.send()` pour envoyer les pages mises en attente vers l'imprimante.

Si cette méthode renvoie `false` (par exemple, si vous n'avez pas appelé `PrintJob.start()` ou si l'utilisateur a annulé la tâche d'impression), les appels ultérieurs de `PrintJob.addPage()` échoueront. Toutefois, si les appels antérieurs à `PrintJob.addPage()` ont réussi, la commande `PrintJob.send()` finale envoie les pages traitées par le spouleur à l'imprimante.

Si vous avez défini une valeur pour *printArea*, les coordonnées `xMin` et `yMin` correspondent à l'angle supérieur gauche (coordonnées 0,0) de la zone imprimable de la page ; cette zone est déterminée par les propriétés `pageHeight` et `pageWidth` définies par `PrintJob.start()`. La sortie d'imprimante s'alignant avec l'angle supérieur gauche de la zone imprimable de la page, elle est attachée à droite et/ou en bas si la zone définie dans *printArea* est plus grande que la zone imprimable de la page. Si vous n'avez pas transmis de valeur pour *printArea* et que la scène est plus grande que la zone imprimable, le même type de découpage est effectué.

Si vous souhaitez mettre à l'échelle un clip avant de l'imprimer, définissez ses propriétés `MovieClip._xscale` et `MovieClip._yscale` avant d'appeler cette méthode, puis redéfinissez-les sur leurs valeurs d'origine après l'appel. L'échelle d'un clip n'a aucun rapport avec *`printArea`*. En d'autres termes, si vous spécifiez que vous imprimez une zone de dimensions 50 x 50 pixels, 2 500 pixels sont imprimés. Si vous avez mis le clip à l'échelle, ces mêmes 2 500 pixels sont imprimés, mais aux nouvelles dimensions.

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

Exemple

L'exemple suivant permet d'émettre la commande `addPage()` de différentes manières :

```
mon_btn.onRelease = function ()
{
  var pageCount = 0;

  var ma_pj = new PrintJob();

  if (ma_pj.start())
  {
    // Imprimer la totalité de l'image actuelle de l'animation _root au format
  vectoriel
    if (ma_pj.addPage(0))
    {
      pageCount++;

      // En démarrant à 0,0, imprimer une zone de 400 pixels de large et
  500 pixels de haut
      // de l'image actuelle de l'animation _root au format vectoriel
      if (ma_pj.addPage(0, {xMin:0,xMax:400,yMin:0,yMax:500}))
      {
        pageCount++;

        // En démarrant à 0,0, imprimer une zone de 400 pixels de large et
  500 pixels de haut
        // de l'image 1 de l'animation _root au format bitmap
        if (ma_pj.addPage(0, {xMin:0,xMax:400,yMin:0,yMax:500},
            {printAsBitmap:true}, 1));
        {
          pageCount++;

          // En démarrant avec 50 pixels à droite de 0,0 et 70 pixels en bas,
          // imprimer une zone de 500 pixels de large et 600 pixels de haut
          // de l'image 4 du niveau 5 au format vectoriel
          if (ma_pj.addPage(5, {xMin:50,xMax:550,yMin:70,yMax:670},null, 4))
          {
            pageCount++;

            // En commençant à 0,0, imprimer une zone de 400 pixels de
  largeur
            // et 400 pixels de hauteur de la page 3 du clip "danse_mc"
            // au format bitmap
            if (ma_pj.addPage("danse_mc",
                {xMin:0,xMax:400,yMin:0,yMax:400},{printAsBitmap:true}, 3))
            {
              pageCount++;
```

```
                // En commençant à 0,0, imprimer une zone de 400 pixels de
    largeur
                // et 600 pixels de hauteur de la page 3 du clip "danse_mc"
                // au format vectoriel, à 50 % de sa taille réelle
                var x = Danse_mc._xscale;
                var y = clipDeDanse._yscale;
                Danse_mc._xscale = 50;
                Danse_mc._yscale = 50;

                if (ma_pj.addPage("danse_mc",
                     {xMin:0,xMax:400,yMin:0,yMax:600},null, 3))
                {
                  pageCount++;
                }

                Danse_mc._xscale = x;
                Danse_mc._yscale = y;
              }
            }
          }
        }
      }

    if (pageCount)
    {
      ma_pj.send();
    }
    delete ma_pj;
}
```

Consultez également

PrintJob.send(), PrintJob.start()

PrintJob.send()

Disponibilité

Flash Player 7.

Usage

```
ma_pj.send()
```

Paramètres

Aucun.

Renvoie

Rien.

Description

Méthode : est utilisée après `PrintJob.start()` et `PrintJob.addPage()` pour envoyer des pages mises en attente vers l'imprimante.

Exemple

Pour plus d'informations, consultez `PrintJob.addPage()`.

Consultez également

`PrintJob.addPage()`, `PrintJob.start()`

PrintJob.start()

Disponibilité

Flash Player 7.

Usage

```
ma_pj.start()
```

Paramètres

Aucun.

Renvoie

Une valeur booléenne `true` si l'utilisateur clique sur OK à l'affichage des boîte de dialogue d'impression ou `false` si l'utilisateur clique sur Annuler ou si une erreur se produit.

Description

Méthode : affiche les boîtes de dialogue d'impression du système d'exploitation et démarre la mise en attente. Les boîtes de dialogue d'impression permettent à l'utilisateur de modifier les paramètres d'impression, puis de charger les propriétés en lecture seule suivantes (notez que 1 point équivaut à 1 pixel) :

Propriété	Caractères	Unités	Remarques
`PrintJob.paperHeight`	Number	Points	Hauteur générale du papier
`PrintJob.paperWidth`	Number	Points	Largeur générale du papier
`PrintJob.pageHeight`	Number	Points	Hauteur de la zone imprimable réelle sur la page ; toutes les marges définies par l'utilisateur sont ignorées
`PrintJob.pageWidth`	Number	Points	Largeur de la zone imprimable réelle sur la page ; toutes les marges définies par l'utilisateur sont ignorées
`PrintJob.orientation`	String	Sans objet	« Portrait » ou « Paysage »

Une fois que l'utilisateur a cliqué sur OK dans la boîte de dialogue d'impression, le lecteur commence à mettre en attente une tâche d'impression dans le système d'exploitation. Vous devez activer toutes les commandes ActionScript qui affectent l'impression, avant de commencer à utiliser les commandes `PrintJob.addPage()` pour démarrer l'envoi des pages au spouleur. Si vous le souhaitez, vous pouvez utiliser les propriétés de largeur, de hauteur et d'orientation renvoyées par cette méthode pour déterminer le formatage de l'impression.

Etant donné que l'utilisateur reçoit des informations telles que « Impression de la page 1 » immédiatement après avoir cliqué sur OK, vous devez appeler les commandes `PrintJob.addPage()` et `PrintJob.send()` dès que possible.

Si cette méthode renvoie `false` (par exemple, si l'utilisateur clique sur Annuler au lieu de OK), les appels ultérieurs à `PrintJob.addPage()` et `PrintJob.send()` échouent. Cependant, si vous testez cette valeur renvoyée et n'envoyez pas les commandes `PrintJob.addPage()`, vous devrez encore supprimer l'objet PrintJob pour vous assurer que le spouleur d'impression est nettoyé, comme dans l'exemple ci-dessous.

```
var ma_pj = new PrintJob();
var myResult = ma_pj.start();
  if(monRésultat){
    // instructions addPage() et send() ici
  }
delete ma_pj;
```

Exemple

Pour plus d'informations, consultez `PrintJob.addPage()`.

Consultez également

`PrintJob.addPage()`, `PrintJob.send()`

printNum()

Disponibilité

Flash Player 5.

Remarque : Si vous utilisez Flash Player7 ou ultérieur, vous pouvez créer un objet PrintJob, qui vous permet (ainsi que l'utilisateur) de mieux contrôler le processus d'impression. Pour plus d'informations, consultez l'entrée Classe PrintJob.

Usage

```
printNum (niveau, "régionDimpression")
```

Paramètres

niveau Le niveau de Flash Player à imprimer. Par défaut, toutes les images du niveau sont imprimées. Affectez une étiquette #p aux images que vous souhaitez imprimer dans ce niveau.

régionDimpression Un modificateur qui définit la zone d'impression de l'animation. Mettez ce paramètre entre guillemets et spécifiez l'une des valeurs suivantes :

- `bmovie` Désigne le cadre de délimitation d'une image spécifique dans une animation comme zone d'impression de toutes les images imprimables de l'animation. Affectez une étiquette d'image #b à l'image dont vous souhaitez utiliser le cadre de délimitation comme région d'impression.
- `bmax` Désigne un composite de tous les cadres de délimitation de l'ensemble des images imprimables comme zone d'impression. Spécifiez le paramètre `bmax` lorsque la taille des images de votre animation varie.
- `bframe` Indique que le cadre de délimitation de chaque image imprimable est utilisé comme zone d'impression pour cette image. Cela modifie la région d'impression pour chaque image et modifie la taille des objets pour qu'ils correspondent à la région d'impression. Utilisez `bframe` si vous avez des objets de tailles diverses dans chaque image et si vous souhaitez que chaque objet remplisse la page imprimée.

Renvoie

Rien.

Description

Fonction : imprime le niveau de Flash Player en fonction des limites spécifiées dans le paramètre *régionDimpression* ("bmovie", "bmax", "bframe"). Affectez une étiquette #p aux images que vous souhaitez imprimer dans le clip cible. Bien que l'utilisation de `printNum()` offre des impressions de meilleure qualité que l'utilisation de `printAsBitmapNum()`, vous ne pouvez pas utiliser `printNum()` pour imprimer des animations contenant des transparences alpha ou des effets de couleur spéciaux.

Si vous utilisez `bmovie` pour le paramètre *régionDimpression*, mais que vous n'affectez pas d'étiquette #b à une image, la zone d'impression est déterminée par la taille de scène de la vidéo chargée. L'animation n'hérite pas de la taille de scène de l'animation principale.

Tous les éléments imprimables d'une animation doivent être complètement chargés avant de commencer l'impression.

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

Consultez également

`print()`, `printAsBitmap()`, `printAsBitmapNum`, Classe PrintJob

private

Disponibilité

Flash Player 6.

Usage

```
class nomClasse{
  private var nom;
  private function nom() {
    // vos instructions
  }
}
```

Remarque : Pour utiliser ce mot-clé, vous devez définir ActionScript 2.0 et Flash Player 6 ou version ultérieure dans l'onglet Flash de la boîte de dialogue Paramètres de publication de votre fichier FLA. Ce mot-clé n'est supporté que lorsqu'il est utilisé dans des fichiers de scripts externes et non dans des scripts écrits dans le panneau Actions.

Paramètres

nom Le nom de la variable ou de la fonction à spécifier en tant que variable ou fonction privée.

Description

Mot-clé : 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 dernière. Par défaut, une variable ou une fonction est disponible pour toutes les classes qui l'appellent. Utilisez ce mot-clé pour restreindre l'accès à une variable ou une fonction. Pour plus d'informations, consultez *Contrôle de l'accès des membres*, page 172.

Vous pouvez uniquement utiliser ce mot-clé dans les définitions de classes, et non dans les définitions d'interfaces.

Consultez également

`public`, `static`

public

Flash Player6

Usage

```
class nomClasse{
  public var nom;
  public function nom() {
    // vos instructions
  }
}
```

Remarque : Pour utiliser ce mot-clé, vous devez définir ActionScript 2.0 et Flash Player 6 ou version ultérieure dans l'onglet Flash de la boîte de dialogue Paramètres de publication de votre fichier FLA. Ce mot-clé n'est supporté que lorsqu'il est utilisé dans des fichiers de scripts externes et non dans des scripts écrits dans le panneau Actions.

Paramètres

nom Le nom de la variable ou de la fonction à spécifier en tant que variable ou fonction privée.

Description

Mot-clé : spécifie qu'une variable ou une fonction est disponible pour toutes les classes qui l'appellent. Les variables et les fonctions étant publiques par défaut, ce mot-clé est utilisé principalement pour des raisons stylistiques. Par exemple, vous pouvez utiliser ce mot-clé dans un but d'homogénéisation dans un bloc de code qui contient également des variables privées ou statiques.

Exemple

Les deux blocs de code suivants sont fonctionnellement identiques.

```
private var age:Number;
public var nom:String;
static var naissance:Date;

private var age:Number;
var nom:String;
static var naissance:Date;
```

Pour plus d'informations, consultez *Contrôle de l'accès des membres*, page 172.

Consultez également

`private`, `static`

_quality

Disponibilité

Flash Player 5.

Usage

```
_quality
```

Description

Propriété (globale) : définit ou récupère la qualité de rendu utilisée pour une animation. Les polices de périphérique sont toujours aliasées et ne sont donc pas affectées par la propriété `_quality`.

La propriété `_quality` peut être définie avec les valeurs suivantes :

- `"LOW"` Qualité de rendu minimum. Les graphiques ne sont pas anti-aliasés et les bitmaps ne sont pas lissés.
- `"MEDIUM"` Qualité de rendu moyenne. Les graphiques sont anti-aliasés avec une grille 2x2, en pixels, mais les bitmaps ne sont pas lissés. Convient aux animations ne contenant pas de texte.
- `"HIGH"` Qualité de rendu élevée. Les graphiques sont anti-aliasés avec une grille 4x4, en pixels, et les bitmaps sont lissés si l'animation est statique. Il s'agit du paramètre de qualité de rendu par défaut utilisé par Flash.
- `"BEST"` Qualité de rendu très élevée. Les graphiques sont anti-aliasés avec une grille 4x4, en pixels, et les bitmaps sont toujours lissés.

Exemple

L'exemple suivant définit la qualité de rendu sur `LOW`

```
_quality = "LOW";
```

Consultez également

`_highquality`, `toggleHighQuality()`

random

Disponibilité

Flash Player 4. Cette fonction est déconseillée dans Flash 5 ; utilisez plutôt `Math.random()`.

Usage

```
random(valeur)
```

Paramètres

`valeur` Un entier.

Renvoie

Un entier.

Description

Fonction : renvoie un entier aléatoire entre 0 et un de moins que l'entier spécifié dans le paramètre *`valeur`*.

Exemple

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

```
random(5);
```

Consultez également

`Math.random()`

removeMovieClip()

Disponibilité

Flash Player 4.

Usage

```
removeMovieClip(cible)
```

Paramètres

`cible` Le chemin cible d'une occurrence de clip créé avec `duplicateMovieClip()` ou le nom d'occurrence d'un clip créé avec `MovieClip.attachMovie()` ou `MovieClip.duplicateMovieClip`.

Renvoie

Rien.

Description

Fonction : supprime le clip spécifié.

Consultez également

`duplicateMovieClip()`, `MovieClip.duplicateMovieClip`, `MovieClip.attachMovie()`, `MovieClip.removeMovieClip()`

return

Disponibilité

Flash Player 5.

Usage

`return[expression]`

Paramètres

`expression` Une chaîne, un nombre, un tableau ou un objet à évaluer et à renvoyer comme valeur de la fonction. Ce paramètre est facultatif.

Renvoie

Le paramètre *`expression`* évalué, si fourni.

Description

Instruction : spécifie la valeur renvoyée par une fonction. L'action `return` évalue *`expression`* et renvoie le résultat en tant que valeur de la fonction dans laquelle elle est exécutée. L'action `return` entraîne l'arrêt de la fonction et son remplacement par la valeur renvoyée. Si l'instruction `return` est utilisée seule, elle renvoie `null`.

Vous ne pouvez pas renvoyer plusieurs valeurs. Si vous tentez de le faire, seule la dernière valeur est renvoyée. Dans l'exemple suivant, `c` est renvoyé :

```
return a, b, c ;
```

Exemple

L'exemple suivant utilise l'action `return` à l'intérieur du corps de la fonction `somme()` pour renvoyer la valeur de la somme des trois paramètres. La ligne de code suivante appelle la fonction `somme()` et affecte la valeur renvoyée à la variable `nouvelleValeur` :

```
function somme(a, b, c){
  return a + b + c;
}
nouvelleValeur = somme(4, 32, 78);
trace(nouvelleValeur);
// envoie 114 dans le panneau de sortie
```

Consultez également

`function`

_root

Disponibilité

Flash Player 5.

Usage

```
_root.clip
_root.action
_root.propriété
```

Paramètres

clip Le nom d'occurrence d'un clip.

action Une action ou une méthode.

propriété Une propriété de l'objet MovieClip.

Description

Propriété : spécifie ou renvoie une référence au scénario de l'animation principale. Si une animation possède plusieurs niveaux, le scénario principal de l'animation 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.

La spécification de `_root` est identique à l'utilisation de la notation à barre oblique (`/`) pour spécifier un chemin absolu au sein du niveau courant.

Remarque : Si une animation contenant `_root` est chargée dans une autre animation, `_root` fait référence au scénario de l'animation en cours de chargement et non au scénario contenant `_root`. Si vous souhaitez vérifier que `_root` fait référence au scénario de l'animation chargée même si elle est chargée dans une autre animation, utilisez `MovieClip._lockroot`.

Exemple

L'exemple suivant arrête le scénario du niveau contenant le script en cours d'exécution :

```
_root.stop();
```

L'exemple suivant envoie le scénario dans le niveau actuel à l'image 3 :

```
_root.gotoAndStop(3);
```

Consultez également

`MovieClip._lockroot`, `_parent`, `targetPath`

scroll

Disponibilité

Flash Player 4.

Usage

```
nomDeVariableDeChampDeTexte.scroll = x
```

Description

Propriété : une propriété maintenant déconseillée qui contrôle l'affichage des informations d'un champ de texte associé à une variable. La propriété `scroll` définit l'endroit où le champ de texte commence à afficher le contenu ; après l'avoir définie, le Flash Player la met à jour au fur et à mesure que l'utilisateur fait défiler le champ de texte. La propriété `scroll` est utile pour diriger les utilisateurs vers un paragraphe spécifique dans un long passage, ou pour créer des champs de texte défilants. Cette propriété peut être récupérée et modifiée.

Exemple

Le code suivant est associé à un bouton Haut qui fait défiler le champ de texte `monTexte` :

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

Consultez également

`TextField.maxscroll`, `TextField.scroll`

Classe Selection

Disponibilité

Flash Player 5.

Description

La classe Selection vous permet de définir et de contrôler le champ de texte dans lequel est situé le point d'insertion, c'est-à-dire, le champ qui a le focus. Les index de la plage de sélection sont basés sur zéro (par exemple, la première position est 0, la deuxième est 1, et ainsi de suite).

Il n'existe aucune fonction du constructeur pour la classe Selection, car il ne peut y avoir qu'un seul champ de texte avec focus à la fois.

Méthodes de la classe Selection

Méthode	Description
`Selection.addListener`	Enregistre un objet pour la réception de la notification lorsque `onSetFocus` est invoqué.
`Selection.getBeginIndex()`	Renvoie l'index au début de la plage de sélection. Renvoie -1 s'il n'y a pas d'index ou de champ sélectionné courant.
`Selection.getCaretIndex()`	Renvoie la position courante du point d'insertion dans la plage de sélection avec focus. Renvoie -1 s'il n'y a pas de position de point d'insertion ou de plage de sélection avec focus.

Méthode	Description
`Selection.getEndIndex()`	Renvoie l'index de la fin de la plage de sélection. Renvoie -1 s'il n'y a pas d'index ou de champ sélectionné courant.
`Selection.getFocus()`	Renvoie le nom de la variable pour le champ de texte avec focus. Renvoie `null` s'il n'existe aucun champ de texte avec focus.
`Selection.removeListener`	Supprime un objet enregistré avec `addListener()`.
`Selection.setFocus`	Règle le champ de texte associé à la variable spécifiée.
`Selection.setSelection()`	Définit les index de début et de fin de la plage de sélection.

Ecouteurs de la classe Selection

Ecouteur	Description
`Selection.onSetFocus`	Notifié lorsque le focus de saisie change.

Selection.addListener

Disponibilité

Flash Player 6.

Usage

```
Selection.addListener(nouvelEcouteur)
```

Paramètres

nouvelEcouteur Un objet avec une méthode `onSetFocus`.

Renvoie

Rien.

Description

Méthode : enregistre un objet pour la réception de notifications de changement de focus clavier. Lorsque le focus change (par exemple, lorsque `Selection.setFocus` est invoqué), la méthode `onSetFocus` de tous les objets à l'écoute enregistrés avec `addListener()` est invoquée. Plusieurs objets peuvent attendre des notifications de changement de focus. Si l'écouteur *nouvelEcouteur* est déjà enregistré, aucun changement n'a lieu.

Selection.getBeginIndex()

Disponibilité

Flash Player 5.

Usage

```
Selection.getBeginIndex()
```

Paramètres

Aucun.

Renvoie

Un entier.

Description

Méthode : renvoie l'index du début de la plage de sélection. Si aucun index n'existe ou si aucun champ de texte courant n'a le focus, la méthode renvoie -1. Les index de la plage de sélection sont basés sur zéro (par exemple, la première position est 0, la deuxième est 1, et ainsi de suite).

Selection.getCaretIndex()

Disponibilité

Flash Player 5.

Usage

```
Selection.getCaretIndex()
```

Paramètres

Rien.

Renvoie

Un entier.

Description

Méthode : renvoie l'index de la position du point d'insertion clignotant (caret). Si aucun point d'insertion clignotant n'est affiché, la méthode renvoie -1. Les index de la plage de sélection sont basés sur zéro (par exemple, la première position est 0, la deuxième est 1, etc.).

Selection.getEndIndex()

Disponibilité

Flash Player 5.

Usage

```
Selection.getEndIndex()
```

Paramètres

Aucun.

Renvoie

Un entier.

Description

Méthode : renvoie l'index de fin de la zone de sélection avec le focus courant. Si aucun index n'existe ou s'il n'y a aucune plage de sélection avec le focus, la méthode renvoie -1. Les index de la plage de sélection sont basés sur zéro (par exemple, la première position est 0, la deuxième est 1, et ainsi de suite).

Selection.getFocus()

Disponibilité

Flash Player 5. Les noms d'occurrence des boutons et des champs de texte fonctionnent dans Flash Player 6 et ultérieur.

Usage

```
Selection.getFocus()
```

Paramètres

Aucun.

Renvoie

Une chaîne ou `null`.

Description

Méthode : renvoie le nom de variable du champ de texte ayant le focus. Si aucun champ de texte n'a le focus, la méthode renvoie `null`. Si le focus actuel est un bouton et que le bouton est un objet Button, `getFocus()` renvoie le chemin cible sous forme de chaîne. Si le focus actuel est un champ de texte et que le champ de texte est un objet TextField, `getFocus()` renvoie le chemin cible sous forme de chaîne.

Si un clip de bouton est le bouton où se trouve actuellement le focus, `Selection.getFocus()` renvoie le chemin cible du clip de bouton. Si un champ de texte avec un nom d'occurrence a actuellement le focus, `Selection.getFocus()` renvoie le chemin cible de l'objet TextField. Sinon, il renvoie le nom de variable du champ de texte.

Selection.onSetFocus

Disponibilité

Flash Player 6.

Usage

```
unEcouteur.onSetFocus = function(ancienFocus, nouveauFocus){
instructions;
}
```

Description

Ecouteur : notifié lorsque le focus de saisie change. Pour utiliser `onSetFocus`, vous devez créer un objet d'écoute. Vous pouvez alors définir une fonction pour `onSetFocus` et utiliser la méthode `addListener` pour enregistrer l'écouteur avec l'objet Selection, comme dans l'exemple suivant :

```
unEcouteur = new Object();
unEcouteur.onSetFocus = function () { ... };
Selection.addListener(unEcouteur);
```

Les écouteurs permettent à différents morceaux de code de coopérer étant donné que plusieurs écouteurs peuvent recevoir une notification concernant un seul événement.

Consultez également

`Selection.addListener`

Selection.removeListener

Disponibilité

Flash Player 6.

Usage

```
Selection.removeListener(écouteur)
```

Paramètres

`écouteur` L'objet qui ne reçoit plus de notification de focus.

Renvoie

Si l'*`écouteur`* a été correctement retiré de la méthode renvoie `true`. Si l'*`écouteur`* n'a pas été correctement retiré, par exemple si l'*`écouteur`* n'apparaissait pas dans la liste des écouteurs de l'objet Selection, la méthode renvoie `false`.

Description

Méthode : supprime un objet précédemment enregistré avec `addListener()`.

Selection.setFocus

Disponibilité

Flash Player 5. Les noms d'occurrence des boutons et des clips ne fonctionnent que dans Flash Player 6 et ses versions ultérieures.

Usage

```
Selection.setFocus("nomDoccurrence")
```

Paramètres

nomDoccurrence Chaîne spécifiant le chemin vers le nom d'occurrence d'un bouton, d'un clip ou d'un champ de texte.

Renvoie

Un événement.

Description

Méthode : donne le focus au champ de texte, bouton ou clip sélectionnable (modifiable) spécifié par *nomDoccurrence*. Le paramètre *nomDoccurrence* doit être une chaîne littérale du chemin de cette occurrence. Vous pouvez utiliser la notation à point ou la notation à 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 en points.

Si `null` est transmis, le focus actuel est retiré.

Exemple

L'exemple suivant donne le focus à un champ de texte associé à `maVar`, sur le scénario principal. Etant donné que le paramètre *nomDoccurrence* est un chemin absolu, vous pouvez appeler l'action à partir de tout scénario.

```
Selection.setFocus("_root.maVar");
```

Dans l'exemple suivant, le champ de texte associé à `maVar` est dans un clip appelé `monClip` dans le scénario principal. Vous pouvez utiliser l'un ou l'autre des deux chemins suivants pour définir le focus, le premier étant relatif, le deuxième absolu.

```
Selection.setFocus("monClip.maVar");
Selection.setFocus("_root.monClip.maVar");
```

Selection.setSelection()

Disponibilité

Flash Player 5.

Usage

```
Selection.setSelection(début, fin)
```

Paramètres

début L'index de début de la plage de sélection.

fin L'index de fin de la plage de sélection.

Renvoie

Rien.

Description

Méthode : définit la plage de sélection du champ de texte avec focus. La nouvelle plage de sélection commencera à l'index spécifié dans le paramètre *début* et finira à l'index spécifié dans le paramètre *fin*. Les index de la plage de sélection sont basés sur zéro (par exemple, la première position est 0, la deuxième est 1, et ainsi de suite). Cette méthode n'a aucun effet si aucun champ de texte n'a le focus.

set

Disponibilité

Flash Player 6.

Usage

```
function set propriété(nomVar) {
  // vos instructions
}
```

Remarque : Pour utiliser ce mot-clé, vous devez définir ActionScript 2.0 et Flash Player 6 ou version ultérieure dans l'onglet Flash de la boîte de dialogue Paramètres de publication de votre fichier FLA. Ce mot-clé n'est supporté que lorsqu'il est utilisé dans des fichiers de scripts externes et non dans des scripts écrits dans le panneau Actions.

Paramètres

propriété Mot utilisé en référence à la propriété à laquelle la commande `set` accède ; cette valeur doit être identique à celle utilisée dans la commande `get` correspondante.

nomVar La variable locale qui définit la valeur en cours d'affectation.

Renvoie

Rien.

Description

Mot-clé : permet une « définition » implicite des propriétés associées aux objets, en fonction des classes définies dans les fichiers de classes externes. L'utilisation de méthodes de définition implicites vous permet d'accéder aux propriétés des objets, sans accéder directement à ces objets. Les méthodes get/set implicites sont des abréviations syntaxiques de la méthode `Object.addProperty()` dans ActionScript 1.

Pour plus d'informations, consultez *Méthodes get/set implicites*, page 181.

Consultez également

`get`, `Object.addProperty()`

Variable set

Disponibilité

Flash Player 4.

Usage

```
set(variable, expression)
```

Paramètres

variable Un identifiant devant contenir la valeur du paramètre de *l'expression*.

expression Une valeur affectée à la variable.

Renvoie

Rien.

Description

Instruction : affecte une valeur à une variable. Une *variable* est un conteneur d'informations. 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 contenir n'importe quel type de données (par exemple, Chaîne, Nombre, Booléen, Objet ou Clip). Le scénario de chaque fichier SWF et clip possède son propre jeu de variables et chaque variable possède sa propre valeur indépendante des variables des autres scénarios.

La saisie stricte des données n'est pas autorisée au sein d'une instruction `set`. Si vous utilisez cette instruction pour définir une variable sur une valeur dont le type de données est différent du type associé à la variable d'un fichier de classe, aucune erreur de compilation n'est renvoyée.

Exemple

Cet exemple définit une variable appelée `orig_x_pos` qui stocke la position originale de l'axe *x* du clip `vaisseau` afin de pouvoir le réinitialiser à sa position de début ultérieurement dans le fichier SWF.

```
on(release) {
  set("pos_x_orig", getProperty ("vaisseau", _x ));
}
```

Le code précédent donne le même résultat que le suivant :

```
on(release) {
  pos_x_orig = vaisseau._x;
}
```

Consultez également

`var`, `call()`

setInterval()

Disponibilité

Flash Player 6.

Usage

```
setInterval(NomDeFonction, intervalle [, param1, param2, ..., paramN])
```

Paramètres

NomDeFonction Un nom de fonction ou une référence à une fonction anonyme.

intervalle Le temps en millisecondes qui s'écoule entre les appels du paramètre *nomDeFonction*.

param1, param2, ..., paramN Paramètres facultatifs transmis à la *fonction* ou au paramètre *NomDeMéthode*.

Renvoie

Un identifiant d'intervalle que vous pouvez transmettre à `clearInterval` pour annuler l'intervalle.

Description

Fonction : appelle une fonction, une méthode ou un objet à intervalles périodiques pendant la lecture d'un fichier SWF. Vous pouvez utiliser une fonction d'intervalle pour mettre à jour des variables d'une base de données ou mettre à jour un temps affiché.

Si *intervalle* est inférieur à la cadence du fichier SWF (par exemple, 10 images par seconde est égal à 100 millisecondes), la fonction d'intervalle est appelée aussi près que possible de *intervalle*. Vous devez utiliser la fonction `updateAfterEvent()` pour assurer une actualisation de l'écran à une fréquence appropriée. Si *intervalle* est supérieur à la cadence du fichier SWF, la fonction d'intervalle est uniquement appelée chaque fois que la tête de lecture entre dans une image, afin de minimiser l'impact de chaque actualisation de l'écran.

Exemple

Usage 1 : l'exemple suivant appelle une fonction anonyme toutes les 1000 millisecondes (toutes les secondes).

```
setInterval( function(){ trace("intervalle appelé"); }, 1000 );
```

Usage 2 : l'exemple suivant définit deux gestionnaires d'événements, et appelle chacun d'eux. Les deux appels de `setInterval()` envoient la chaîne `"intervalle appelé"` vers le panneau de sortie toutes les 1 000 millisecondes. Le premier appel de `setInterval()` appelle la fonction `callback1()`, qui contient une action `trace()`. Le deuxième appel de `setInterval()` transmet la chaîne `"intervalle appelé"` à la fonction `callback2()` en tant que paramètre.

```
function callback1() {
  trace("intervalle appelé");
}

function callback2(param) {
  trace(param);
}
```

```
setInterval( callback1, 1000 );
setInterval( callback2, 1000, "intervalle appelé" );
```

Usage 3 : cet exemple utilise une méthode d'un objet. Vous devez utiliser cette syntaxe lorsque vous voulez appeler une méthode qui est définie pour un objet.

```
obj = new Object();
obj.interval = function() {
  trace("fonction d'intervalle appelée");
}

setInterval( obj, "intervalle", 1000 );

obj2 = new Object();
obj2.interval = function(s) {
  trace(s);
}
setInterval( obj2, "intervalle", 1000, "fonction d'intervalle appelée" );
```

Vous devez utiliser la deuxième forme de la syntaxe `setInterval()` pour appeler la méthode d'un objet, comme suit :

```
setInterval( obj2, "intervalle", 1000, "fonction d'intervalle appelée" );
```

Consultez également

`clearInterval()`, `updateAfterEvent()`

setProperty()

Disponibilité

Flash Player 4.

Usage

```
setProperty("cible", propriété, valeur/expression)
```

Paramètres

cible Le chemin d'accès au nom d'occurrence du clip dont la propriété est à définir.

propriété La propriété à définir.

valeur La nouvelle valeur littérale de la propriété.

expression Une équation qui équivaut à la nouvelle valeur de la propriété.

Renvoie

Rien.

Description

Fonction : modifie la valeur d'une propriété de clip pendant la lecture de l'animation.

Exemple

Cette instruction définit la propriété `_alpha` du clip nommé `étoile` sur 30 % lorsque le bouton est cliqué :

```
on(release) {
  setProperty("étoile", _alpha, "30");
}
```

Consultez également

`getProperty`

Classe SharedObject

Disponibilité

Flash Player 6.

Description

Les objets partagés sont très puissants : ils permettent de partager des données en temps réel entre les objets persistants situés sur l'ordinateur de l'utilisateur. Les objets locaux partagés sont parfois appelés « cookies ».

Vous pouvez utiliser les objets locaux partagés afin de maintenir une persistance locale. Il s'agit de la manière la plus simple d'utiliser un objet partagé. Par exemple, vous pouvez appeler `SharedObject.getLocal()` pour créer un objet partagé, tel qu'un calculateur avec mémoire, dans le lecteur. Comme l'objet partagé est localement persistant, Flash enregistre ses attributs de données sur l'ordinateur de l'utilisateur à la fin du fichier SWF. Lors de la prochaine exécution du fichier SWF, le calculateur contiendra les valeurs qu'il contenait à la fin du fichier SWF. En outre, si vous définissez les propriétés de l'objet partagé sur `null` avant la fin du fichier SWF, le calculateur s'ouvre sans aucune valeur préalable lors de la prochaine exécution du fichier SWF.

Pour créer un objet local partagé, utilisez la syntaxe suivante :

```
// Créer un objet local partagé
so = SharedObject.getLocal("machin");
```

Considérations sur l'espace disque local

Les objets locaux partagés sont toujours persistants sur le client, dans la limite de mémoire et d'espace disque disponibles.

Par défaut, Flash peut enregistrer localement les objets distants partagés persistants jusqu'à une taille de 100 K. Lorsque vous tentez d'enregistrer un objet plus volumineux, le lecteur Flash affiche la boîte de dialogue Enregistrement local, qui permet à l'utilisateur d'accepter ou de refuser un stockage local pour le domaine qui demande l'accès. Vérifiez que votre scène mesure au moins 215 x 138 pixels : la taille minimale nécessaire pour que Flash affiche cette boîte de dialogue.

Si l'utilisateur clique sur Autoriser, l'objet est enregistré et `SharedObject.onStatus` est appelé avec une propriété de `code` de `SharedObject.Flush.Success`. Si l'utilisateur clique sur Refuser, l'objet n'est pas enregistré et `SharedObject.onStatus` est appelé avec une propriété de `code` de `SharedObject.Flush.Failed`.

Pour spécifier les paramètres d'enregistrement local permanents pour un domaine spécifique, l'utilisateur peut également cliquer avec le bouton droit de la souris (Windows) ou tout en appuyant sur la touche Contrôle (Macintosh) lors de la lecture d'un fichier SWF, choisir Paramètres, puis ouvrir le panneau Enregistrement local.

La liste suivante résume l'interaction entre les choix d'espace disque de l'utilisateur et les objets partagés :

- Si l'utilisateur sélectionne Jamais, les objets ne sont jamais enregistrés localement et toutes les commandes `SharedObject.flush()` appelées pour l'objet renvoie la valeur `false`.
- Si l'utilisateur sélectionne Illimité (en déplaçant le curseur au maximum vers la droite), les objets sont enregistrés localement jusqu'à la limite de l'espace disque disponible.
- Si l'utilisateur sélectionne Aucun (en déplaçant le curseur au maximum vers la gauche), toutes les commandes `SharedObject.flush()` émises pour l'objet renvoient « `en attente` » et le lecteur demande à l'utilisateur si un espace disque supplémentaire peut être affecté pour faire de la place à l'objet, comme expliqué ci-dessus.

- Si l'utilisateur sélectionne 10 Ko, 100 Ko, 1 Mo ou 10 Mo, les objets sont enregistrés localement et `SharedObject.flush()` renvoie la valeur `true` si l'objet entre dans l'espace indiqué. Si davantage d'espace est nécessaire, `SharedObject.flush()` renvoie « `en attente` » et le lecteur demande à l'utilisateur si un espace disque supplémentaire peut être affecté pour faire de la place à l'objet, comme indiqué ci-dessus.

De plus, si l'utilisateur sélectionne une valeur inférieure à la quantité d'espace disque actuellement utilisé pour des données persistantes localement, le lecteur avertit l'utilisateur que tout objet partagé localement et enregistré est supprimé.

Remarque : Dans Flash Player, aucune limite de taille n'existe pour l'environnement auteur.

Méthodes de la classe SharedObject

Méthode	Description
`SharedObject.clear()`	Purge toutes les données de l'objet partagé et supprime l'objet partagé du disque.
`SharedObject.flush()`	Ecrit immédiatement un objet persistant partagé localement dans un fichier local.
`SharedObject.getLocal()`	Renvoie une référence à un objet persistant partagé localement disponible uniquement pour le client actuel.
`SharedObject.getSize()`	Obtient la taille actuelle de l'objet partagé, en octets.

Propriétés de la classe SharedObject

Propriété (lecture seule)	Description
`SharedObject.data`	La collecte des attributs affectés à la propriété `data` de l'objet ; ces attributs peuvent être partagés et/ou stockés.

Gestionnaires d'événement de la classe SharedObject

Gestionnaire d'événement	Description
`SharedObject.onStatus`	Invoqué à chaque fois qu'un message d'erreur, un avertissement ou des informations sont envoyés pour un objet partagé.

Constructeur de la classe SharedObject

Pour plus d'informations sur la création d'objets partagés localement, consultez `SharedObject.getLocal()`.

SharedObject.clear()

Disponibilité

Flash Player 7.

Usage

```
mon_so.clear()
```

Paramètres

Aucun.

Renvoie

Rien.

Description

Méthode : purge toutes les données issues de l'objet partagé et supprime l'objet partagé du disque. La référence à *mon_so* est toujours active et *mon_so* est actuellement vide.

SharedObject.data

Disponibilité

Flash Player 6.

Usage

monObjetLocalPartagé.data

Description

Propriété (lecture seule) : la collecte des attributs affectés à la propriété `data` de l'objet ; ces attributs peuvent être partagés et/ou stockés. Chaque attribut peut être l'objet de n'importe quel type ActionScript ou JavaScript de base : Tableau, Nombre, Booléen, etc. Par exemple, les lignes suivantes affectent des valeurs à différents aspects d'un objet partagé :

```
itemsArray = new Array(101,346,483);
currentUserIsAdmin = true;
currentUserName = "Raymonde" ;
so.data.itemNumbers = itemsArray ;
so.data.adminPrivileges = currentUserIsAdmin ;
so.data.userName = currentUserName ;
```

Tous les attributs de la propriété `data` d'un objet partagé sont enregistrés si l'objet est persistant.

Remarque : N'affectez pas de valeurs directement à la propriété `data` d'un objet partagé, comme dans `so.data=uneValeur` ; Flash ignore ces affectations.

Pour supprimer les attributs d'objets locaux partagés localement, utilisez un code comme `delete so.data.attributeName` ; définir un attribut sur `nul` ou `undefined` pour un objet local partagé n'entraîne pas sa suppression.

Pour créer des valeurs « privées » pour un objet partagé (des valeurs disponibles uniquement pour l'occurrence de client alors que l'objet est en cours d'utilisation et non stockées avec l'objet lorsqu'il est fermé), créez des propriétés qui ne sont pas appelées `data` pour les stocker, comme indiqué dans l'exemple suivant.

```
so.favoriteColor = "bleu";
so.favoriteNightClub = "The Bluenote Tavern";
so.favoriteSong = "My World is Blue";
```

Exemple

L'exemple suivant définit le flux actuel vers la sélection de l'utilisateur.

```
curStream = _root.so.data.msgList[selected].streamName;
```

Consultez également

Classe Sound

SharedObject.flush()

Disponibilité

Flash Player 6.

Usage

monObjetLocalPartagé.flush([*espaceDisqueMin*])

Paramètres

espaceDisqueMin Nombre entier facultatif précisant le nombre d'octets qui doivent être affectés pour cet objet. La valeur par défaut est 0.

Renvoie

Une valeur booléenne `true` ou `false` ou une valeur de chaîne « `en attente` ».

- Si l'utilisateur a autorisé le stockage d'informations locales pour les objets de ce domaine et si la quantité d'espace affecté est suffisante pour stocker l'objet, cette méthode renvoie `true`. (Si vous avez spécifié une valeur pour *espaceDisqueMin*, la quantité d'espace affectée doit être au moins égale à la valeur `true` à renvoyer).
- Si l'utilisateur a autorisé le stockage d'informations locales pour des objets de ce domaine, mais si la quantité d'espace affectée n'est pas suffisante pour stocker l'objet, cette méthode renvoie « `en attente` ».
- Si l'utilisateur a toujours refusé le stockage d'informations locales pour des objets de ce domaine ou si Flash est incapable d'enregistrer l'objet pour une raison quelconque, cette méthode renvoie `false`.

Description

Méthode : écrit immédiatement un objet persistant partagé localement dans un fichier local. Si vous n'utilisez pas cette méthode, Flash écrit l'objet partagé dans un fichier lorsque la session de l'objet partagé se termine (c'est-à-dire lorsque le fichier SWF est fermé, lorsque l'objet partagé est déposé dans la corbeille car il ne possède plus aucune référence ou lorsque vous appelez `SharedObject.data`).

Si cette méthode renvoie « `en attente` », Flash Player affiche une boîte de dialogue demandant à l'utilisateur d'augmenter la quantité d'espace disque disponible pour les objets de ce domaine. Pour que l'espace occupé par l'objet partagé puisse « grandir » lors d'un prochain enregistrement, en évitant ainsi des valeurs de retour « `en attente` », définissez une valeur *espaceDisqueMin*. Lorsque Flash tente d'écrire le fichier, il recherche le nombre d'octets définis pour *espaceDisqueMin*, plutôt que de rechercher un espace juste suffisant pour enregistrer l'objet partagé à sa taille actuelle.

Par exemple, si l'on prévoit qu'un objet partagé va atteindre la taille maximale de 500 octets, même s'il peut être plus petit au départ, définissez 500 pour *espaceDisqueMin*. Si Flash demande à l'utilisateur d'affecter un espace disque pour l'objet partagé, il demandera 500 octets. Une fois la quantité d'espace nécessaire allouée par l'utilisateur, Flash n'a plus besoin de demander d'espace supplémentaire lors des prochaines tentatives pour purger l'objet (tant que sa taille ne dépasse pas500 octets).

Une fois que l'utilisateur a répondu à la boîte de dialogue, cette méthode est rappelée et renvoie `true` ou `false`. De plus, `SharedObject.onStatus` est invoqué avec une propriété de `code` `SharedObject.Flush.Success` ou `SharedObject.Flush.Failed`.

Pour plus d'informations, consultez *Considérations sur l'espace disque local*, page 701.

Exemple

La fonction suivante obtient un objet partagé `SO` et remplit les propriétés enregistrables à l'aide des paramètres fournis par l'utilisateur. Finalement, `flush()` est appelé pour enregistrer les paramètres et affecter un minimum de 1 000 octets d'espace disque.

```
this.SyncSettingsCore=function(nomso, écraser, paramètres)
{
  var SO=SharedObject.getLocal(nomso, "http://www.mondomaine.fr/app/sys");

  // index de la liste des paramètres
  var i;

  // Pour chaque valeur spécifiée dans les paramètres :
  // Si l'écrasement renvoie true, définissez le paramètre persistant sur la
  valeur fournie.
  // Si l'écrasement renvoie false, cherchez le paramètre persistant, sauf
  s'il
  // n'y en a pas, auquel cas, définissez-le sur la valeur fournie.
  for (i in settings) {
    if (override || (SO.data[i] == null)) {
      SO.data[i]= settings[i];
    } else {
      settings[i]= SO.data[i];
    }
  }
  SO.flush(1000);
}
```

SharedObject.getLocal()

Disponibilité

Flash Player 6.

Usage

```
SharedObject.getLocal(nomObjet [, cheminLocal])
```

Remarque : La syntaxe correcte est `SharedObject.getLocal`. Pour affecter un objet à une variable, utilisez une syntaxe telle que *monObjetLocalPartagé*=`SharedObject.getLocal`.

Paramètres

nomObjet Nom de l'objet. Le nom peut inclure des barres obliques (/) ; par exemple, `travail/adresses` est un nom légal. Les espaces ne sont pas autorisés dans un nom d'objet partagé, ni les caractères suivants :

```
~ % & \ ; :  " ' , < > ? #
```

cheminLocal Un paramètre de chaîne facultatif qui précise le chemin complet ou partiel vers le fichier SWF qui a créé l'objet partagé et qui détermine où l'objet partagé est stocké localement. La valeur par défaut est le chemin complet.

Renvoie

Une référence à un objet partagé qui est localement persistant et disponible uniquement pour le client actuel. Si Flash ne peut pas créer ni trouver l'objet partagé (par exemple, si *cheminLocal* a été spécifié mais si aucun répertoire de ce type n'existe), cette méthode renvoie `null`.

Description

Méthode : renvoie une référence à un objet partagé qui est localement persistant et disponible uniquement pour le client actuel.

Pour éviter les collisions de nom, Flash repère l'emplacement du fichier SWF qui crée l'objet partagé. Par exemple, si un fichier SWF sur www.maSociete.fr/apps/stockwatcher.swf crée un objet partagé appelé `portefeuille`, cet objet partagé n'entre pas en conflit avec un autre objet appelé `portefeuille` créé par un fichier SWF sur www.votreSociete.fr/photoshoot.swf, car les fichiers SWF proviennent de deux répertoires différents.

Exemple

L'exemple suivant enregistre la dernière image entrée par un utilisateur dans un objet `kookie` partagé localement.

```
// Obtenir kookie
so = sharedobject.getlocal("kookie");

// Obtenir l'utilisateur de kookie et atteindre le numéro d'image enregistré
  pour cet utilisateur.
if (so.data.user != undefined) {
  this.user = so.data.user;
  this.gotoAndStop(so.data.frame);
}
```

Le bloc de code suivant est placé sur chaque image du fichier SWF.

```
// Sur chaque image, appelez la fonction rememberme pour enregistrer le numéro
  d'image.
fonction rememberme() {
  so.data.frame=this._currentFrame;
  so.data.user="John";
}
```

SharedObject.getSize()

Disponibilité

Flash Player 6.

Usage

monObjetLocalPartagé.getSize()

Paramètres

Aucun.

Renvoie

Valeur numérique indiquant la taille de l'objet partagé, en octets.

Description

Méthode : obtenir la taille actuelle de l'objet partagé, en octets.

Flash calcule la taille d'un objet partagé en passant en revue chacune de ses propriétés de données ; plus l'objet possède de propriétés de données, plus il faut de temps pour estimer sa taille. Pour cette raison, l'estimation de la taille de l'objet peut engendrer un coût de traitement important. Par conséquent, vous voulez peut-être éviter d'utiliser cette méthode, sauf besoin particulier.

Exemple

L'exemple suivant donne la taille de l'objet partagé `so`.

```
var tailleObjPart= this.so.getSize();
```

SharedObject.onStatus

Disponibilité

Flash Player 6.

Usage

```
monObjetLocalPartagé.onStatus = function(objetInfo) {
  // vos instructions
}
```

Paramètres

objetInfo Un paramètre défini selon le message d'état.

Renvoie

Rien.

Description

Gestionnaire d'événement : invoqué chaque fois qu'un message d'erreur, un avertissement ou des informations sont envoyés pour un objet partagé. Si vous souhaitez répondre à ce gestionnaire d'événement, vous devez créer une fonction pour traiter l'objet d'information généré par l'objet partagé.

L'objet d'informations a une propriété de `code` contenant une chaîne décrivant le résultat du gestionnaire `onStatus`, ainsi qu'une propriété `level` contenant une chaîne `"Status"` ou `"Error"`.

Outre 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 lui répondre, Flash traite une fonction affectée à `System.onStatus` s'il en existe une.

Les événements suivants vous informent lorsque certaines activités SharedObject se produisent.

Propriété de code	Propriété de niveau	Signification
`SharedObject.Flush.Failed`	Erreur	Une commande `SharedObject.flush()` qui renvoie "pending" a échoué (l'utilisateur n'a pas alloué d'espace disque supplémentaire pour l'objet partagé lorsque Flash Player a affiché le panneau Enregistrement local).
`SharedObject.Flush.Success`	Etat	Une commande `SharedObject.flush()` renvoyant « en attente » a réussi (l'utilisateur a affecté un espace disque supplémentaire pour l'objet partagé).

Consultez également

`SharedObject.getLocal()`, `System.onStatus`

Classe Sound

Disponibilité

Flash Player 5.

Description

La classe Sound vous permet de contrôler le son dans une animation. Vous pouvez ajouter des sons à un clip à partir de la bibliothèque pendant la lecture de l'animation et contrôler ces sons. Si vous ne spécifiez pas de cible (cible) lorsque vous créez un nouvel objet Sound, vous pouvez utiliser les méthodes pour contrôler le son de toute l'animation.

Vous devez utiliser le constructeur `new Sound` pour créer un objet Sound avant d'appeler les méthodes de la classe Sound.

Méthodes de la classe Sound

Méthode	Description
`Sound.attachSound()`	Attache le son spécifié dans le paramètre.
`Sound.getBytesLoaded()`	Renvoie le nombre d'octets chargés pour le son spécifié.
`Sound.getBytesTotal()`	Renvoie la taille du son, en octets.
`Sound.getPan()`	Renvoie la valeur de l'appel `setPan()` précédent.
`Sound.getTransform()`	Renvoie la valeur de l'appel `setTransform()` précédent.
`Sound.getVolume()`	Renvoie la valeur de l'appel `setVolume()` précédent.
`Sound.loadSound`	Charge un fichier MP3 dans Flash Player.
`Sound.setPan`	Définit la balance gauche/droite du son.
`Sound.setTransform`	Définit la quantité de chaque canal, gauche et droite, à lire dans chaque haut-parleur.
`Sound.setVolume`	Définit le niveau du volume pour un son.
`Sound.start()`	Démarre la lecture d'un son depuis le début ou, en option, depuis un point défini dans le paramètre.
`Sound.stop`	Arrête le son spécifié ou tous les sons en cours de lecture.

Propriétés de la classe Sound

Propriété	Description
`Sound.duration`	Durée d'un son, en millisecondes.
`Sound.ID3`	Donne accès aux métadonnées qui font partie d'un fichier MP3.
`Sound.position`	Nombre de millisecondes de lecture du son.

Gestionnaires d'événement de la classe Sound

Gestionnaire d'événement	Description
`Sound.onID3`	Invoqué chaque fois que de nouvelles données ID3 sont disponibles.
`Sound.onLoad`	Invoqué au chargement d'un son.
`Sound.onSoundComplete`	Invoqué à la fin de la lecture d'un son.

Constructeur de la classe Sound

Disponibilité

Flash Player 5.

Usage

```
new Sound([cible])
```

Paramètres

`cible` L'occurrence de clip sur laquelle agit l'objet Sound. Ce paramètre est facultatif.

Renvoie

Rien.

Description

Constructeur : crée un nouvel objet Sound pour un clip spécifié. Si vous ne spécifiez pas d'occurrence cible, l'objet Sound contrôle tous les sons de l'animation.

Exemple

L'exemple suivant crée un nouvel objet Sound nommé `son_global`. La deuxième ligne appelle la méthode `setVolume()` et règle le volume de tous les sons de l'animation à 50 %.

```
son_global = new Sound();
son_global.setVolume(50);
```

L'exemple suivant crée une nouvelle occurrence de l'objet Sound, lui transmet le clip cible *`mon_mc`* et appelle la méthode *`start`*, qui démarre tous les sons de *`mon_mc`*.

```
animation_sound = new Sound(mon_mc);
animation_sound.start();
```

Sound.attachSound()

Disponibilité

Flash Player 5.

Usage

```
mon_sound.attachSound("NomId")
```

Paramètres

NomId L'identifiant d'un son exporté dans la bibliothèque. L'identifiant est situé dans la boîte de dialogue Propriétés de liaison.

Renvoie

Rien.

Description

Méthode : associe le son spécifié par le paramètre *NomId* à l'objet Sound spécifié. Le son doit se trouver dans la bibliothèque du fichier SWF actuel et être spécifié pour l'exportation dans la boîte de dialogue Propriétés de liaison. Vous devez appeler `Sound.start()` pour démarrer la lecture du son.

Pour être sûr de pouvoir contrôler le son à partir de toute scène du fichier SWF, placez-le dans le scénario principal du fichier SWF.

Sound.duration

Disponibilité

Flash Player 6.

Usage

```
mon_sound.duration
```

Description

Propriété (lecture seule) : la durée d'un son, en millisecondes.

Sound.getBytesLoaded()

Disponibilité

Flash Player 6.

Usage

mon_sound.getBytesLoaded()

Paramètres

Aucun.

Renvoie

Un entier indiquant le nombre d'octets chargés.

Description

Méthode : renvoie le nombre d'octets chargés (en flux continu) pour l'objet Sound spécifié. Vous pouvez comparer la valeur de `getBytesLoaded` avec celle de `getBytesTotal` pour déterminer le pourcentage d'un son qui a été chargé.

Consultez également

Sound.getBytesTotal()

Sound.getBytesTotal()

Disponibilité

Flash Player 6.

Usage

mon_sound.getBytesTotal()

Paramètres

Aucun.

Renvoie

Un entier indiquant la taille totale, en octets, de l'objet Sound spécifié.

Description

Méthode : renvoie la taille, en octets, de l'objet Sound spécifié.

Consultez également

Sound.getBytesLoaded()

Sound.getPan()

Disponibilité

Flash Player 5.

Usage

```
mon_sound.getPan();
```

Paramètres

Aucun.

Renvoie

Un entier.

Description

Méthode : renvoie le niveau de balance défini dans le dernier appel `setPan` sous la forme d'un entier compris entre -100 (gauche) et 100 (droite). 0 règle le canal gauche et le canal droit au même niveau. Le paramètre de panoramique contrôle la balance gauche-droite des sons actuels et futurs d'un fichier SWF.

Cette méthode est cumulable avec les méthodes `setVolume()` ou `setTransform()`.

Consultez également

`Sound.setPan`

Sound.getTransform()

Disponibilité

Flash Player 5.

Usage

```
mon_sound.getTransform();
```

Paramètres

Aucun.

Renvoie

Un objet avec des propriétés qui contiennent les valeurs de pourcentage de canal pour l'objet son spécifié.

Description

Méthode : renvoie les informations de transformation de son pour l'objet Sound spécifié défini avec le dernier appel `Sound.setTransform`.

Sound.getVolume()

Disponibilité

Flash Player 5.

Usage

mon_sound.getVolume()

Paramètres

Aucun.

Renvoie

Un entier.

Description

Méthode : renvoie le niveau du volume du son sous la forme d'un entier compris entre 0 et 100, avec 0 pour éteint et 100 pour volume maximum. Le paramètre par défaut est 100.

Consultez également

Sound.setVolume

Sound.ID3

Disponibilité

Flash Player 6 ; comportement mis à jour dans Flash Player 7.

Usage

mon_sound.ID3

Description

Propriété (lecture seule) : donne accès aux métadonnées qui font 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 effectuer une requête sur ces propriétés. Les balises ID3 utilisant le jeu de caractères UTF-8 sont les seules à être supportées.

Flash Player 6 version 40 et les versions ultérieures utilisent la propriété Sound.id3 pour traiter les balises ID3 1.0 et ID3 1.1. Flash Player 7 prend également en charge les balises ID3 2.0, notamment les balises 2.3 et 2.4. Pour la compabilité amont, les balises Sound.id3 et Sound.ID3 sont prises en charge. Les conseils de code sont uniquement pris en charge pour l'utilisation des minuscules dans les balises id3 (consultez *Utilisation des conseils de code*, page 68).

Le tableau suivant répertorie les balises ID3 standard et le type de contenu correspondant ; vous pouvez les demander au format *mon_sound*.ID3.COMM, *mon_sound*.ID3.TIME, etc. Les fichiers MP3 sont susceptibles de contenir des balises non répertoriées dans ce tableau ; Sound.ID3 permet également d'y accéder.

Propriété	Description
COMM	Commentaire
TALB	Album/animation/titre du spectacle
TBPM	Battements par minute
TCOM	Compositeur
TCON	Type de contenu
TCOP	Message de copyright
TDAT	Date
TDLY	Cadence de la liste musicale
TENC	Encodé par
TEXT	Parolier/auteur
TFLT	Type de fichier
TIME	Heure
TIT1	Description générique du contenu
TIT2	Titre/nom de la chanson/description du contenu
TIT3	Sous-titre/qualité description
TKEY	Note initiale
TLAN	Langages
TLEN	Longueur
TMED	Type de média
TOAL	Album/animation/titre du spectacle d'origine
TOFN	Nom de fichier d'origine
TOLY	Paroliers/auteurs d'origine
TOPE	Artistes/interprètes d'origine
TORY	Année originelle de parution
TOWN	Propriétaire du fichier/titulaire de la licence
TPE1	Principaux interprètes/solistes
TPE2	Groupe/orchestre/accompagnement
TPE3	Chef d'orchestre/qualité interprète
TPE4	Interprété, remixé ou modifié par
TPOS	Partie d'une série

Propriété	Description
TPUB	Editeur
TRCK	Numéro de piste/emplacement dans la série
TRDA	Dates des enregistrements
TRSN	Nom de la station de radio sur Internet
TRSO	Propriétaire de la station de radio sur Internet
TSIZ	Taille
TSRC	ISRC (international standard recording code - code international normalisé des enregistrements)
TSSE	Logiciel/matériel et paramètres utilisés pour l'encodage
TYER	Année
WXXX	Image de lien URL

Flash Player 6 prenait en charge plusieurs balises ID31.0. Si ces balises ne se trouvent pas dans le fichier MP3, mais que ce fichier contient les balises ID3 2.0 correspondantes, les balises ID3 2.0 sont copiées dans les propriétés ID3 1.0, tel que le montre le tableau suivant. Ce processus offre une compatibilité amont avec les scripts déjà enregistrés 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

Exemple

Consultez `Sound.onID3` pour obtenir un exemple d'utilisation de cette propriété.

Consultez également

`Sound.attachSound()`, `Sound.loadSound`

Sound.loadSound

Disponibilité

Flash Player 6.

Usage

```
mon_sound.loadSound("url", isStreaming)
```

Paramètres

url L'emplacement d'un fichier son MP3 sur un serveur.

isStreaming Valeur booléenne qui indique si le son est lu en flux continu (`true`) ou s'il s'agit d'un son d'événement (`false`).

Renvoie

Rien.

Description

Méthode : charge un fichier MP3 dans un objet Sound. Vous pouvez utiliser le paramètre *isStreaming* pour indiquer si le son est lu en flux continu ou s'il s'agit d'un son d'événement.

Les sons d'événement sont complètement chargés avant d'être lus. Ils sont gérés par la classe Sound d'ActionScript et répondent à toutes les méthodes et propriétés de cette classe.

Les sons en flux continu sont lus pendant leur téléchargement. La lecture commence lorsque suffisamment de données ont été reçues pour démarrer le décompresseur.

Tous les MP3 (événement ou en continu) chargés à l'aide de cette méthode sont enregistrés dans le cache du navigateur sur le système de l'utilisateur.

Exemple

L'exemple suivant charge un son d'événement :

```
mon_sound.loadSound( "http://cheminDuServeur:port/nomDuFichierMp3", false);
```

L'exemple suivant charge un son en flux continu :

```
mon_sound.loadSound( "http://cheminDuServeur:port/nomDuFichierMp3", false);
```

Consultez également

`Sound.onLoad`

Sound.onID3

Disponibilité

Flash Player 7.

Usage

```
mon_sound.onID3 = function(){
  // vos instructions
}
```

Paramètres

Aucun.

Renvoie

Rien.

Description

Gestionnaire d'événement : invoqué à chaque fois que de nouvelles données ID3 sont disponibles pour un fichier MP3 que vous chargez en utilisant `Sound.attachSound()` ou `Sound.loadSound`. Ce gestionnaire permet d'accéder aux données ID3 sans procéder à une interrogation. Si les balises ID3 1.0 et ID3 2.0 sont présentes dans un fichier, ce gestionnaire est appelé à deux reprises.

Exemple

L'exemple suivant renvoie les propriétés ID3 de chanson.mp3 dans le panneau Sortie.

```
mon_sound = new Sound();
mon_sound.onID3 = function(){
  for( var prop in mon_sound.ID3 ){
    trace( prop + " : "+ mon_sound.ID3[prop] );
  }
}
mon_sound.loadSound("chanson.mp3", false);
```

Consultez également

`Sound.attachSound()`, `Sound.ID3`, `Sound.loadSound`

Sound.onLoad

Disponibilité

Flash Player 6.

Usage

```
mon_sound.onLoad = function(succès){
  // vos instructions
}
```

Paramètres

succès Une valeur booléenne `true` si *mon_sound* a été correctement chargé, false dans le cas contraire.

Renvoie

Rien.

Description

Gestionnaire d'événement : automatiquement invoqué au chargement d'un son. Vous devez créer une fonction exécutée lorsque ce gestionnaire est invoqué. Vous pouvez utiliser une fonction anonyme ou une fonction nommée (pour un exemple de chaque, consultez `Sound.onSoundComplete`). Ce gestionnaire doit être défini avant l'appel de *mon_sound*`.loadSound()`.

Consultez également

`Sound.loadSound`

Sound.onSoundComplete

Disponibilité

Flash Player 6.

Usage

```
mon_sound.onSoundComplete = function(){
  // vos instructions
}
```

Paramètres

Aucun.

Renvoie

Rien.

Description

Gestionnaire d'événement : automatiquement invoqué à la fin de la lecture d'un son. Vous pouvez utiliser ce gestionnaire pour déclencher des événements dans un fichier SWF au terme de la lecture d'un son.

Vous devez créer une fonction exécutée lorsque ce gestionnaire est invoqué. Vous pouvez utiliser une fonction anonyme ou une fonction nommée.

Exemple

Usage 1 : l'exemple suivant utilise une fonction anonyme :

```
mon_sound = new Sound();
mon_sound.attachSound("monIDSon");
mon_sound.onSoundComplete = function() {
  trace("monIDSon est terminé");
}
mon_sound.start();
```

Usage 2 : l'exemple suivant utilise une fonction nommée :

```
function callback1() {
  trace("monIDSon est terminé");
}

mon_sound = new Sound();
mon_sound.attachSound("monIDSon");
mon_sound.onSoundComplete = rappel1;
mon_sound.start();
```

Consultez également

Sound.onLoad

Sound.position

Disponibilité

Flash Player 6.

Usage

```
mon_sound.position
```

Description

Propriété (lecture seule) : renvoie le nombre de millisecondes écoulées depuis le début de la lecture d'un son. Si le son est lu en boucle, la position est remise à 0 au début de chaque boucle.

Sound.setPan

Disponibilité

Flash Player 5.

Usage

```
mon_sound.setPan(balance);
```

Paramètres

balance Un entier spécifiant la balance gauche-droite d'un son. Les valeurs correctes sont comprises entre -100 et 100, avec -100 pour le canal de gauche, 100 pour le canal de droite et 0 pour répartir le son d'une manière uniforme entre les deux canaux.

Renvoie

Un entier.

Description

Méthode : détermine la façon dont le son est réparti dans les canaux droit et gauche (haut-parleurs). Pour les sons mono, *balance* détermine le haut-parleur (gauche ou droit) par lequel passe le son.

Exemple

L'exemple suivant crée une occurrence de l'objet Sound appelé `mon_sound` et associe un son à l'identifiant `L7` de la bibliothèque. Il appelle également les méthodes `setVolume()` et `setPan()` pour contrôler le son `L7`.

```
onClipEvent(mouseDown) {
// créer un objet Sound
  mon_sound = new Sound(this);
// associer un son de la bibliothèque
  mon_sound.attachSound("L7");
// définir le volume à 50 %
  mon_sound.setVolume(50);
// désactiver le son dans le canal droit
  mon_sound.setPan(-100);
// démarrer à partir de 30 secondes et lire le son 5 fois
  mon_sound.start(30, 5);
```

Consultez également

`Sound.attachSound()`, `Sound.setPan`, `Sound.setTransform`, `Sound.setVolume`, `Sound.start()`

Sound.setTransform

Disponibilité

Flash Player 5.

Usage

mon_sound.setTransform(*objetDeTransformationDeSon*)

Paramètres

objetDeTransformationDeSon Un objet créé à l'aide du constructeur de la classe générique Object.

Renvoie

Rien.

Description

Méthode : définit les informations de transformation (balance) du son pour un objet Sound.

Le paramètre *objetDeTransformationDeSon* est un objet créé à l'aide de la méthode constructeur de la classe générique Object avec des paramètres spécifiant la manière dont le son est réparti dans les canaux gauche et droit (haut-parleurs).

Les sons utilisent un espace disque et une mémoire considérables. Les sons stéréo utilisant deux fois plus de données que les sons mono, il est généralement conseillé d'utiliser des sons mono de 6 bits et 22 kHz. Vous pouvez utiliser la méthode `setTransform()` pour lire les sons mono en stéréo, les sons stéréo en mono et pour ajouter des effets sonores intéressants.

Les paramètres de *objetDeTransformationDeSon* sont les suivants :

`ll` Une valeur de pourcentage spécifiant la quantité de l'entrée gauche à lire dans le haut-parleur gauche (0–100).

`lr` Une valeur de pourcentage spécifiant la quantité de l'entrée droite à lire dans le haut-parleur gauche.

`rr` Une valeur de pourcentage spécifiant la quantité de l'entrée droite à lire dans le haut-parleur droit.

`rl` Une valeur de pourcentage spécifiant la quantité de l'entrée gauche à lire dans le haut-parleur droit.

Le résultat net de ces paramètres est représenté par la formule suivante :

```
sortieGauche = entrée_gauche * ll + entrée_droite * lr
sortieDroite = entrée_droite * rr + entrée_gauche * rl
```

Les valeurs de `entrée_gauche` ou `entrée_droite` sont déterminées par le type de son (mono ou stéréo) du fichier SWF.

Les sons stéréo divisent l'entrée de son de manière égale entre les haut-parleurs droit et gauche et ont les paramètres de transformation par défaut suivants :

```
ll = 100
lr = 0
rr = 100
rl = 0
```

Les sons mono sont lus dans le haut-parleur gauche et ont les paramètres de transformation par défaut suivants :

```
ll = 100
lr = 100
rr = 0
rl = 0
```

Exemple

L'exemple suivant illustre un résultat pouvant être obtenu avec la méthode `setTransform()`, mais pas avec les méthodes `setVolume()` ou `setPan()`, même associées.

Le code suivant crée un objet `monObjetDeTransformationDeSon` et définit ses propriétés de manière à ce que le son des deux canaux soit lu uniquement dans le canal gauche.

```
monObjetDeTransformationDeSon = new Object;
monObjetDeTransformationDeSon.ll = 100;
monObjetDeTransformationDeSon.lr = 100;
monObjetDeTransformationDeSon.rr = 0;
monObjetDeTransformationDeSon.rl = 0;
```

Pour appliquer l'objet `monObjetDeTransformationDeSon` à un objet Sound, vous devez alors transmettre l'objet à l'objet Sound à l'aide de la méthode `setTransform()`, comme suit :

```
mon_sound.setTransform(monObjetDeTransformationDeSon);
```

L'exemple suivant lit un son stéréo en mono ; les paramètres de l'objet `monObjetDeTransformationDeSonMono` sont les suivants :

```
monObjetDeTransformationDeSonMono = new Object;
monObjetDeTransformationDeSonMono.ll = 50;
monObjetDeTransformationDeSonMono.lr = 50;
monObjetDeTransformationDeSonMono.rr = 50;
monObjetDeTransformationDeSonMono.rl = 50;
mon_sound.setTransform(monObjetDeTransformationDeSonMono);
```

Cet exemple lit le canal gauche à la moitié de sa capacité et ajoute le reste du canal gauche dans le canal droit ; les paramètres de l'objet `monObjetDeTransformationDeSonMoitié` sont les suivants :

```
monObjetDeTransformationDeSonMoitié = new Object;
monObjetDeTransformationDeSonMoitié.ll = 50;
monObjetDeTransformationDeSonMoitié.lr = 0;
monObjetDeTransformationDeSonMoitié.rr = 100;
monObjetDeTransformationDeSonMoitié.rl = 50;
mon_sound.setTransform(monObjetDeTransformationDeSonMoitié);
```

Consultez également

Classe Object

Sound.setVolume

Disponibilité

Flash Player 5.

Usage

mon_sound.setVolume(*volume*)

Paramètres

volume Un nombre compris entre 0 et 100 et correspondant au niveau du volume. 100 est le volume maximum et 0 le volume nul. Le paramètre par défaut est 100.

Renvoie

Rien.

Description

Méthode : définit le volume pour l'objet Sound.

Exemple

L'exemple suivant définit le volume sur 50 % et transfère progressivement le son du haut-parleur gauche vers le droit :

```
onClipEvent(load) {
  i = -100;
  mon_sound = new Sound();
  mon_sound.setVolume(50);
}
onClipEvent (enterFrame) {
  if (i <= 100) {
    mon_sound.setPan(i++);
  }
}
```

Consultez également

Sound.setPan, Sound.setTransform

Sound.start()

Disponibilité

Flash Player 5.

Usage

```
mon_sound.start([décalageSecondes, boucle])
```

Paramètres

décalageSecondes Un paramètre facultatif permettant de démarrer la lecture du son à un point spécifique. Par exemple, si vous avez un son de 30 secondes et que vous souhaitez démarrer la lecture du son au milieu, spécifiez 15 pour le paramètre *décalageSecondes*. Le son n'est pas retardé de 15 secondes, la lecture démarrant à la seconde 15.

boucle Un paramètre facultatif permettant de spécifier le nombre de répétitions du son.

Renvoie

Rien.

Description

Méthode : démarre la lecture du dernier son associé, depuis le début si aucun paramètre n'est spécifié, ou à l'endroit du son spécifié dans le paramètre *décalageSecondes*.

Consultez également

Sound.stop

Sound.stop

Disponibilité

Flash Player 5.

Usage

```
mon_sound.stop(["nomIdentifiant"])
```

Paramètres

nomIdentifiant Un paramètre facultatif spécifiant un son spécifique à arrêter. Le paramètre *nomIdentifiant* doit se trouver entre guillemets (" ").

Renvoie

Rien.

Description

Méthode : arrête tous les sons en cours de lecture si aucun paramètre n'est spécifié, ou uniquement le son spécifié dans le paramètre *nomIdentifiant*.

Consultez également

Sound.start()

_soundbuftime

Disponibilité

Flash Player 4.

Usage

```
_soundbuftime = entier
```

Paramètres

entier Le nombre de secondes précédant le démarrage de la lecture en continu du fichier SWF.

Description

Propriété (globale) : établit le nombre de secondes de mise en tampon d'un son en flux continu. La valeur par défaut est de 5 secondes.

Classe Stage

Disponibilité

Flash Player 6.

Description

La classe Stage est une classe de premier niveau dont les méthodes, les propriétés et les gestionnaires sont accessibles sans constructeur.

Les méthodes et les propriétés de cette classe permettent d'accéder aux informations sur les limites d'un fichier SWF et de les manipuler.

Méthodes de la classe Stage

Méthode	Description
`Stage.addListener`	Ajoute un objet d'écoute permettant de détecter le redimensionnement d'un fichier SWF.
`Stage.removeListener`	Supprime un objet d'écoute de l'objet Stage.

Propriétés de la classe Stage

Propriété	Description
`Stage.align`	Alignement du fichier SWF dans le lecteur ou dans le navigateur.
`Stage.height`	Hauteur de la scène, en pixels.
`Stage.scaleMode`	Echelle actuelle du fichier SWF.
`Stage.showMenu`	Affiche ou masque les éléments par défaut du menu contextuel Flash Player.
`Stage.width`	Largeur de la scène, en pixels.

Gestionnaires d'événement de la classe Stage

Gestionnaire d'événement	Description
`Stage.onResize`	Invoqué lorsque `Stage.scaleMode` a pour valeur `"noScale"` et que le fichier SWF est redimensionné.

Stage.addListener

Disponibilité

Flash Player 6.

Usage

```
Stage.addListener(monEcouteur)
```

Paramètres

monEcouteur Un objet qui attend une notification de rappel de la part de l'événement `Stage.onResize`.

Renvoie

Rien.

Description

Méthode : détecte le redimensionnement d'un fichier SWF (uniquement si `Stage.scaleMode = "noScale"`). La méthode `addListener()` ne fonctionne pas avec le paramètre d'échelle d'animation par défaut (`"showAll"`), ni avec d'autres paramètres d'échelle (`"exactFit"` et `"noBorder"`).

Pour utiliser `addListener()`, vous devez d'abord créer un *objet écouteur*. Les objets d'écoute de Stage reçoivent une notification envoyée par `Stage.onResize`.

Exemple

Cet exemple crée un objet d'écoute appelé `monEcouteur`. Il utilise ensuite `monEcouteur` pour appeler `onResize` et définir une fonction qui sera appelée lorsque `onResize` sera déclenché. Enfin, le code ajoute l'objet `monEcouteur` à la liste de rappel de l'objet Stage. Les objets d'écoute permettent à plusieurs objets d'attendre des notifications de redimensionnement.

```
monEcouteur = new Object();
monEcouteur.onResize = function () { ... }
Stage.scaleMode = "noScale"
Stage.addListener(monEcouteur);
```

Consultez également

`Stage.onResize`, `Stage.removeListener`

Stage.align

Disponibilité

Flash Player 6.

Usage

```
Stage.align
```

Description

Propriété : indique l'alignement actuel du fichier SWF dans le lecteur ou dans le navigateur.

Le tableau suivant répertorie les valeurs de la propriété `align`. Les valeurs ne figurant pas dans le tableau centrent le fichier SWF dans la zone du lecteur ou du navigateur.

Value	Vertical	Horizontal
`"T"`	top	center
`"B"`	bottom	center
`"L"`	center	left
`"R"`	center	right
`"TL"`	top	left
`"TR"`	top	right
`"BL"`	bottom	left
`"BR"`	bottom	right

Stage.height

Disponibilité

Flash Player 6.

Usage

```
Stage.height
```

Description

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 du lecteur. Lorsque la valeur de `Stage.scaleMode` n'est pas `"noScale"`, `height` représente la hauteur du fichier SWF.

Consultez également

`Stage.align`, `Stage.scaleMode`, `Stage.width`

Stage.onResize

Disponibilité

Flash Player 6.

Usage

```
monEcouteur.onResize = function (){
  // vos instructions
}
```

Paramètres

Aucun.

Paramètres

Aucun.

Renvoie

Rien.

Description

Gestionnaire d'événement : invoqué lorsque `Stage.scaleMode` a pour valeur `"noScale"` et que le fichier SWF est redimensionné. Vous pouvez utiliser ce gestionnaire d'événement pour écrire une fonction qui dispose les objets sur la scène lorsqu'un fichier SWF est redimensionné.

Exemple

L'exemple suivant affiche un message dans le panneau de sortie lorsque la scène est redimensionnée.

```
Stage.scaleMode = "noScale"
monEcouteur = new Object();
monEcouteur.onResize = function () {
  trace("La taille de la scène est maintenant" + Stage.width + " par " +
  Stage.height);
}
Stage.addListener(monEcouteur);
// plus tard, appel de Stage.removeListener(monEcouteur)
```

Consultez également

`Stage.addListener`, `Stage.removeListener`

Stage.removeListener

Disponibilité

Flash Player 6.

Usage

```
Stage.removeListener(monEcouteur)
```

Paramètres

monEcouteur Un objet ajouté à la liste de rappel à l'aide de la méthode `addListener()`.

Renvoie

Une valeur booléenne.

Description

Méthode : retire un objet d'écoute créé avec `addListener()`.

Consultez également

`Stage.addListener`

Stage.scaleMode

Disponibilité

Flash Player 6.

Usage

```
Stage.scaleMode = "valeur"
```

Description

Propriété : indique l'échelle actuelle du fichier SWF sur la scène. La propriété `scaleMode` oblige le fichier SWF à adopter 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` à la valeur par défaut de `"showAll"`.

Stage.showMenu

Disponibilité

Flash Player 6.

Usage

```
Stage.showMenu
```

Description

Propriété (lecture-écriture) : indique si les éléments par défaut du menu contextuel Flash Player sont affichés ou masqués. Si `showMenu` est défini sur la valeur `true` (valeur par défaut), toutes les options du menu contextuel s'affichent. Si `showMenu` est défini sur la valeur `false`, seule l'option Paramètres s'affiche.

Consultez également

Classe ContextMenu, Classe ContextMenuItem

Stage.width

Disponibilité

Flash Player 6.

Usage

```
Stage.width
```

Description

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 du lecteur. Lorsque la valeur de `Stage.scaleMode` n'est pas `"noScale"`, `width` représente la largeur du fichier SWF.

Consultez également

`Stage.align`, `Stage.height`, `Stage.scaleMode`

startDrag()

Disponibilité

Flash Player 4.

Usage

```
startDrag(cible,[verrouiller ,gauche, haut, droite, bas])
```

Paramètres

cible Le chemin cible du clip à déplacer.

verrouiller Une valeur booléenne spécifiant si le clip déplaçable est verrouillé au centre de la position de la souris (*true*) ou verrouillé sur le point auquel l'utilisateur a cliqué sur le clip (*false*). Ce paramètre est facultatif.

gauche, *haut*, *droite*, *bas* Valeurs relatives aux coordonnées du parent du clip spécifiant un rectangle de contrainte pour le clip. Ces paramètres sont facultatifs.

Renvoie

Rien.

Description

Fonction : autorise le déplacement du clip *cible* pendant la lecture de l'animation. Vous ne pouvez déplacer qu'un seul clip à la fois. Une fois l'opération `startDrag()` exécutée, le clip reste déplaçable jusqu'à ce qu'il soit explicitement arrêté par `stopDrag()` ou jusqu'à ce qu'une action `startDrag` soit appelée pour un autre clip.

Exemple

Pour créer un clip que l'utilisateur pourra positionner à n'importe quel endroit, associez les actions `startDrag()` et `stopDrag()` à un bouton dans le clip.

```
on(press) {
  startDrag(this,true);
}
on(release) {
  stopDrag();
}
```

Consultez également

`MovieClip._droptarget`, `MovieClip.startDrag()`, `stopDrag()`

static

Disponibilité

Flash Player 6.

Usage

```
class nomClasse{
  static var nom;
  static function nom() {
    // vos instructions
  }
}
```

Remarque : Pour utiliser ce mot-clé, vous devez définir ActionScript 2.0 et Flash Player 6 ou version ultérieure dans l'onglet Flash de la boîte de dialogue Paramètres de publication de votre fichier FLA. Ce mot-clé n'est supporté que lorsqu'il est utilisé dans des fichiers de scripts externes et non dans des scripts écrits dans le panneau Actions.

Paramètres

nom Le nom de la variable ou fonction que vous souhaitez définir comme statique.

Description

Mot clé : indique la création d'une seule variable ou fonction par classe au lieu d'une création dans chaque objet à partir de cette classe. Pour plus d'informations, consultez *Membres d'occurrence et de classe*, page 173.

Vous pouvez utiliser ce mot clé dans les définitions de classe uniquement, pas dans les définitions d'interface.

Consultez également

`private`, `public`

stop()

Disponibilité

Flash 2.

Usage

```
stop()
```

Paramètres

Aucun.

Renvoie

Rien.

Description

Fonction : interrompt la lecture du fichier SWF actuel. L'utilisation la plus fréquente de cette action sert à contrôler les clips à l'aide de boutons.

stopAllSounds()

Disponibilité

Flash Player 3.

Usage

```
stopAllSounds()
```

Paramètres

Aucun.

Renvoie

Rien.

Description

Fonction : arrête tous les sons en cours de lecture dans un fichier SWF sans arrêter la tête de lecture. La lecture des sons définis en flux continu reprendra lorsque la tête de lecture passera dans les images où ils se trouvent.

Exemple

Le code suivant peut être appliqué à un bouton qui, lorsque l'utilisateur clique dessus, arrête tous les sons du fichier SWF.

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

Consultez également

Classe Sound

stopDrag()

Disponibilité

Flash Player 4.

Usage

```
stopDrag()
```

Paramètres

Aucun.

Renvoie

Rien.

Description

Fonction : interrompt l'opération de déplacement en cours.

Exemple

Cette instruction arrête l'action de déplacement de l'occurrence `mon_mc` lorsque l'utilisateur relâche le bouton de la souris :

```
on(press) {
  startDrag("mon_mc");
}
on(release) {
  stopDrag();
}
```

Consultez également

`MovieClip._droptarget`, `MovieClip.stopDrag()`, `startDrag()`

" " (délimiteur de chaîne)

Disponibilité

Flash Player 4.

Usage

"*texte*"

Paramètres

texte Un caractère.

Renvoie

Rien.

Description

Délimiteur de chaîne : lorsqu'ils sont utilisés avant et après des caractères, les guillemets droits indiquent que les caractères ont une valeur littérale et sont considérés comme une *chaîne* (pas une variable, ni une valeur numérique, ni un autre élément ActionScript).

Exemple

Cette exemple utilise des guillemets pour indiquer que la valeur de la variable *devinette* est la chaîne littérale « Prince Edward Island » et non le nom d'une variable. La valeur de province est une variable, et non une valeur littérale ; pour déterminer la valeur de province, la valeur de *devinette* doit être localisée.

```
devinette = "Prince Edward Island";

on(release) {
  province = devinette;
  trace(province);
}

// affiche Prince Edward Island dans le panneau de sortie
```

Consultez également

Classe String, String

Classe String

Disponibilité

Flash Player 5 (est devenu un objet natif dans Flash Player 6, améliorant ainsi les performances de manière significative).

Description

La classe String est une enveloppe pour le type de données primitives de la chaîne et contient les méthodes et les propriétés permettant de manipuler les types de valeurs de chaîne primitives. Vous pouvez convertir la valeur de n'importe quel objet en chaîne avec la fonction `String()`.

Toutes les méthodes de la classe String, à l'exception de `concat()`, `fromCharCode()`, `slice()` et `substr()`, sont génériques. Cela signifie que les méthodes appellent elles-mêmes `this.toString()` avant d'effectuer leurs opérations et que vous pouvez utiliser ces méthodes avec des objets autres que String.

Tous les index de chaîne étant basés sur zéro, l'index du dernier caractère de toute chaîne `x` est `x.length - 1`.

Vous pouvez appeler n'importe quelle méthode de la classe String en utilisant la méthode constructeur `new String` ou en utilisant une valeur de chaîne littérale. Si vous spécifiez une chaîne littérale, l'interprète d'ActionScript la convertit automatiquement en objet String temporaire, appelle la méthode puis supprime l'objet String temporaire. Vous pouvez également utiliser la propriété `String.length` avec une chaîne littérale.

Ne confondez pas une chaîne littérale avec un objet String. Dans l'exemple suivant, la première ligne de code crée la chaîne littérale `s1` et la seconde ligne de code crée une occurrence de l'objet String `s2`.

```
s1 = "machin"
s2 = new String("machin")
```

Utilisez des chaînes littérales, à moins que vous n'ayez spécifiquement besoin d'utiliser un objet String.

Méthodes de la classe String

Méthode	Description
`String.charAt`	Renvoie le caractère situé à un endroit spécifique dans une chaîne.
`String.charCodeAt`	Renvoie la valeur du caractère situé à l'index donné sous la forme d'un entier à 16 bits compris entre 0 et 65535.
`String.concat()`	Combine le texte de deux chaînes et renvoie une nouvelle chaîne.
`String.fromCharCode()`	Renvoie une chaîne constituée des caractères spécifiés dans les paramètres.
`String.indexOf`	Renvoie la position de la première occurrence d'une sous-chaîne spécifiée.
`String.lastIndexOf`	Renvoie la position de la dernière occurrence d'une sous-chaîne spécifiée.
`String.slice`	Extrait une section d'une chaîne et renvoie une nouvelle chaîne.
`String.split`	Scinde un objet String en tableau de chaînes en séparant la chaîne en sous-chaînes.

Méthode	Description
String.substr	Renvoie un nombre spécifique de caractères dans une chaîne en commençant à un endroit donné.
String.substring	Renvoie les caractères entre deux index dans une chaîne.
String.toLowerCase	Convertit la chaîne en minuscules et renvoie le résultat ; ne change pas le contenu de l'objet d'origine.
String.toUpperCase	Convertit la chaîne en majuscules et renvoie le résultat ; ne change pas le contenu de l'objet d'origine.

Propriétés de la classe String

Propriété	Description
String.length	Un entier différent de zéro indique le nombre de caractères dans l'objet String spécifié.

Constructeur de la classe String

Disponibilité

Flash Player 5.

Usage

```
new String(valeur)
```

Paramètres

valeur La valeur initiale du nouvel objet String.

Renvoie

Rien.

Description

Constructeur : crée un nouvel objet String.

Consultez également

String, " " (délimiteur de chaîne)

String.charAt

Disponibilité

Flash Player 5.

Usage

ma_str.charAt(*index*)

Paramètres

index Un entier spécifiant la position d'un caractère dans la chaîne. Le premier caractère est indiqué par `0` et le dernier caractère est indiqué par *ma_str*.length-1.

Renvoie

Un caractère.

Description

Méthode : renvoie le caractère à la position spécifiée par le paramètre *index*. Si *index* n'est pas un nombre compris entre 0 et ma_str.length - 1, une chaîne vide est renvoyée.

Cette méthode est semblable à String.charCodeAt sauf que la valeur renvoyée est un caractère et non un code de caractère entier à 16 bits.

Exemple

Dans l'exemple suivant, cette méthode est appelée sur la première lettre de la chaîne "Chris".

```
ma_str = new String("Chris");
i = ma_str.charCodeAt(0); // i = "C"
```

String.charCodeAt

Disponibilité

Flash Player 5.

Usage

ma_str.charCodeAt(*index*)

Paramètres

index Un entier spécifiant la position d'un caractère dans la chaîne. Le premier caractère est indiqué par 0 et le dernier caractère est indiqué par *ma_str*.length-1.

Renvoie

Un entier.

Description

Méthode : renvoie un entier de 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 renvoyé.

Cette méthode est semblable à String.charAt sauf que la valeur renvoyée est un code de caractère entier à 16 bits, pas un caractère.

Exemple

Dans l'exemple suivant, cette méthode est appelée sur la première lettre de la chaîne "Chris".

```
ma_str = new String("Chris");
i = ma_str.charCodeAt(0); // i = 67
```

String.concat()

Disponibilité

Flash Player 5.

Usage

ma_str.concat(*valeur1,...valeurN*)

Paramètres

valeur1,...valeurN Zéro ou plusieurs valeurs à concaténer.

Renvoie

Une chaîne.

Description

Méthode : combine la valeur de l'objet String aux paramètres et renvoie la nouvelle chaîne, la valeur d'origine (*ma_str*) restant inchangée.

String.fromCharCode()

Disponibilité

Flash Player 5.

Usage

```
String.fromCharCode(c1,c2,...cN)
```

Paramètres

c1,c2,...cN Entiers décimaux qui représentent des valeurs ASCII.

Renvoie

Une chaîne.

Description

Méthode : renvoie une chaîne constituée des caractères représentés par les valeurs ASCII des paramètres.

Exemple

Cet exemple utilise la méthode `fromCharCode()` pour insérer un caractère « @ » dans l'adresse électronique.

```
adresse = "chien" + String.fromCharCode(64) + "niche.net";
trace(adresse); // chien@niche.net
```

String.indexOf

Disponibilité

Flash Player 5.

Usage

```
ma_str.indexOf(sousChaîne, [indexDébut])
```

Paramètres

sousChaîne Un entier ou une chaîne spécifiant la sous-chaîne à rechercher dans *ma_str*.

indexDébut Un entier facultatif spécifiant le point de début dans *ma_str* à partir duquel effectuer la recherche de la sous-chaîne.

Renvoie

La position de la première occurrence de la sous-chaîne spécifiée ou -1.

Description

Méthode : recherche la chaîne et renvoie la position de la première occurrence de *sousChaîne* trouvée au niveau de ou après *indexDébut* dans la chaîne appelante. Si *sousChaîne* n'est pas trouvée, la méthode renvoie -1.

Consultez également

`String.lastIndexOf`

String.lastIndexOf

Disponibilité

Flash Player 5.

Usage

```
ma_str.lastIndexOf(sousChaîne, [indexDébut])
```

Paramètres

sousChaîne Un entier ou une chaîne spécifiant la chaîne à rechercher.

indexDébut Un entier facultatif spécifiant le point de départ de la recherche de la *sousChaîne*.

Renvoie

La position de la dernière occurrence de la sous-chaîne spécifiée ou -1.

Description

Méthode : recherche la chaîne, de droite à gauche, et renvoie l'index de la dernière occurrence de *sousChaîne* trouvée avant *indexDébut* dans la chaîne appelante. Si *sousChaîne* n'est pas trouvée, la méthode renvoie -1.

Consultez également

`String.indexOf`

String.length

Disponibilité

Flash Player 5.

Usage

```
ma_str.length
```

Description

Propriété : un entier différent de zéro indiquant le nombre de caractères dans l'objet String spécifié.

Tous les index de chaîne étant basés sur zéro, l'index du dernier caractère de toute chaîne `x` est `x.length - 1`.

String.slice

Disponibilité

Flash Player 5.

Usage

```
ma_str.slice(début, [fin])
```

Paramètres

début Un nombre spécifiant l'index du point de début de la section. Si *début* est un nombre négatif, le point de début est déterminé à partir de la fin de la chaîne, -1 étant le dernier caractère.

fin Un entier qui est égal à 1+ l'index du point de fin de la section. Le caractère indexé par le paramètre *fin* n'est pas inclus dans la chaîne extraite. Si ce paramètre est omis, `string.length` est utilisé. Si `fin` est un nombre négatif, le point de fin est déterminé en décomptant depuis la fin de la chaîne, -1 étant le dernier caractère.

Renvoie

Une sous-chaîne de la chaîne spécifiée.

Description

Méthode : renvoie une chaîne qui inclut le caractère *début* et tous les caractères jusqu'au caractère *fin* (qui est exclus). L'objet String d'origine n'est pas modifié. Si le paramètre *fin* n'est pas spécifié, la fin de la sous-chaîne est la fin de la chaîne. Si la valeur de *début* est supérieure ou égale à la valeur de *fin*, la méthode renvoie une chaîne vide.

Exemple

L'exemple suivant définit une variable, `texte`, crée une occurrence de l'objet String, `ma_str` et lui transmet la variable `texte`. La méthode `slice()` extrait une section de la chaîne contenue dans la variable et `trace()` l'envoie vers le panneau de sortie. L'exemple suivant illustre l'utilisation d'une valeur positive et négative pour le paramètre `fin`.

```
texte = "Lexington";
ma_str = new String( texte );
trace(ma_str.slice( 1, 3 )); // "ex"
trace(ma_str.slice( 1, -6 )); // "ex"
```

Consultez également

`String.substr`, `String.substring`

String.split

Disponibilité

Flash Player 5.

Usage

```
ma_str.split("délimiteur", [limite])
```

Paramètres

délimiteur Le caractère ou la chaîne au niveau duquel *ma_str* est scindée.

limite Le nombre d'éléments à placer dans le tableau. Ce paramètre est facultatif.

Renvoie

Un tableau contenant les sous-chaînes de *ma_str*.

Description

Méthode : scinde un objet String en sous-chaînes à l'endroit où se trouve le paramètre *délimiteur* et renvoie les sous-chaînes dans un tableau. Si vous utilisez une chaîne vide (`""`) comme délimiteur, chaque caractère de la chaîne est placé en tant qu'élément dans le tableau, comme dans le code suivant.

```
ma_str = "Joe";
i = ma_str.split("");
trace (i);
```

Le panneau de sortie affiche le contenu suivant :

```
J, o, e
```

Si le paramètre *délimiteur* est `undefined`, la chaîne tout entière est placée dans le premier élément du tableau renvoyé.

Exemple

L'exemple suivant renvoie un tableau avec cinq éléments.

```
ma_str = "P, A, T, S, Y";
ma_str.split(",");
```

Cet exemple renvoie un tableau avec deux éléments, `"P"` et `"A"`.

```
ma_str.split(",", 2);
```

String.substr

Disponibilité

Flash Player 5.

Usage

ma_str.substr(*début*, [*longueur*])

Paramètres

début Un entier indiquant la position du premier caractère de *ma_str* à utiliser lors de la création de la sous-chaîne. Si *début* est un nombre négatif, la position de début est déterminée depuis la fin de la chaîne, -1 étant le dernier caractère.

longueur Le nombre de caractères dans la sous-chaîne créée. Si *longueur* n'est pas spécifié, la sous-chaîne inclut tous les caractères du début jusqu'à la fin de la chaîne.

Renvoie

Une sous-chaîne de la chaîne spécifiée.

Description

Méthode : renvoie les caractères d'une chaîne depuis l'index spécifié par le paramètre *début* jusqu'au nombre de caractères spécifié dans le paramètre *longueur*. La méthode substr ne modifie pas la chaîne spécifiée par *ma_str* ; elle renvoie une nouvelle chaîne.

String.substring

Disponibilité

Flash Player 5.

Usage

```
ma_str.substring(début, [fin])
```

Paramètres

début Un entier indiquant la position du premier caractère de *ma_str* utilisé pour créer la sous-chaîne. Les valeurs valides pour *début* sont comprises entre 0 et `String.length` - 1. Si *début* est une valeur négative, 0 est utilisé.

fin Un entier étant égal à 1+ l'index du dernier caractère de *ma_str* devant être extrait. Les valeurs valides pour *fin* sont comprises entre 1 et `string.length`. Le caractère indexé par le paramètre *fin* n'est pas inclus dans la chaîne extraite. Si ce paramètre est omis, `string.length` est utilisé. Si ce paramètre est une valeur négative, 0 est utilisé.

Renvoie

Une chaîne.

Description

Méthode : renvoie une chaîne constituée des caractères contenus entre les points spécifiés par les paramètres *début* et *fin*. Si le paramètre *fin* n'est pas spécifié, la fin de la sous-chaîne est la fin de la chaîne. Si la valeur de *début* est égale à la valeur de *fin*, la méthode renvoie une chaîne vide. Si la valeur de *début* est supérieure à la valeur de *fin*, les paramètres sont automatiquement permutés avant l'exécution de la fonction et la valeur d'origine reste inchangée.

String.toLowerCase

Disponibilité

Flash Player 5.

Usage

```
ma_str.toLowerCase()
```

Paramètres

Aucun.

Renvoie

Une chaîne.

Description

Méthode : renvoie une copie de l'objet String, les caractères majuscules étant convertis en minuscules. La valeur d'origine reste inchangée.

String.toUpperCase

Disponibilité

Flash Player 5.

Usage

```
ma_str.toUpperCase()
```

Paramètres

Aucun.

Renvoie

Une chaîne.

Description

Méthode : renvoie une copie de l'objet String, les caractères minuscules étant convertis en majuscules. La valeur d'origine reste inchangée.

String

Disponibilité

Flash Player 4 ; comportement modifié dans Flash Player 7.

Usage

`String(expression)`

Paramètres

`expression` Une expression à convertir en chaîne.

Renvoie

Une chaîne.

Description

Fonction : renvoie une représentation chaîne du paramètre spécifié, comme suit :

Si *`expression`* est un nombre, la chaîne renvoyée est une représentation texte du nombre.

Si *`expression`* est une chaîne, la chaîne renvoyée est *`expression`*.

Si *`expression`* est un objet, la valeur renvoyée est la représentation sous forme de chaîne de l'objet, générée par l'appel de la propriété String de l'objet ou par l'appel de `Object.toString()` si une telle propriété n'existe pas.

Si *`expression`* est `undefined`, les valeurs renvoyées sont les suivantes :

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

Si *`expression`* est une valeur booléenne, la chaîne renvoyée `"true"` ou `"false"`.

Si *`expression`* est un clip, la valeur renvoyée est le chemin cible du clip en notation à barre oblique (/).

Remarque : La notation à barre oblique n'est pas supportée par ActionScript 2.0.

Consultez également

`Number.toString()`, `Object.toString()`, Classe String, `" " (délimiteur de chaîne)`

substring

Disponibilité

Flash Player 4. Cette fonction est abandonnée et remplacée par `String.substr`.

Usage

```
substring("chaîne", index, nombre)
```

Paramètres

chaîne La chaîne à partir de laquelle s'effectue l'extraction de la nouvelle chaîne.

index Le numéro du premier caractère à extraire.

nombre Le nombre de caractères à inclure dans la chaîne extraite, en excluant le caractère d'index.

Renvoie

Rien.

Description

Fonction de chaîne : extrait une portion d'une chaîne. Cette fonction est basée sur un, tandis que les méthodes de l'objet String sont basées sur zéro.

Consultez également

`String.substr`

super

Disponibilité

Flash Player 6.

Usage

```
super.méthode([param1, ..., paramN])
super([param1, ..., paramN])
```

Paramètres

méthode La méthode à invoquer dans la superclasse.

param1 Paramètres facultatifs transmis à la version superclasse de la méthode (syntaxe 1) ou à la fonction constructeur de la superclasse (syntaxe 2).

Renvoie

Les deux formes invoquent une fonction. La fonction peut renvoyer n'importe quelle valeur.

Description

Opérateur : le premier style de syntaxe peut être utilisé dans le corps d'une méthode d'objet pour invoquer la version superclasse d'une méthode, et peut en option transmettre des paramètres (*param1* ... *paramN*) à la méthode de superclasse. Ceci est utile pour créer des méthodes de sous-classe qui ajoutent un comportement supplémentaire aux méthodes superclasse, mais qui invoquent également les méthodes superclasse pour remplir leur comportement d'origine.

Le deuxième style de syntaxe peut être utilisé dans le corps d'une fonction constructeur pour invoquer la version superclasse de la fonction constructeur, et peut en option lui transmettre des paramètres. Ceci est utile pour créer une sous-classe qui effectue une initialisation supplémentaire, mais qui invoque également le constructeur de superclasse pour effectuer une initialisation de superclasse.

switch

Disponibilité

Flash Player 4.

Usage

```
switch (expression){
  clauseDeCas:
  [clauseParDéfaut:]
}
```

Paramètres

expression Toute expression.

clauseDeCas Un mot-clé `case`, suivi d'une expression, de deux points et d'un groupe d'instructions à exécuter si l'expression correspond au paramètre *expression* avec une égalité stricte (===).

clauseParDéfaut Un mot-clé `default`, suivi d'instructions à exécuter si aucune des expressions de cas ne correspond au paramètre *expression* par égalité stricte (===).

Renvoie

Rien.

Description

Instruction : crée une structure de branchement pour des instructions ActionScript. Comme l'action `if`, l'action `switch` teste une condition et exécute des instructions si la condition renvoie une valeur de `true`.

Exemple

Dans l'exemple suivant, si le paramètre `nombre` est 1, l'action `trace()` qui suit `case 1` est exécutée ; si le paramètre `nombre` est 2, l'action `trace()` qui suit `case 2` est exécutée, et ainsi de suite. Si aucune expression `case` ne correspond au paramètre `nombre`, l'action `trace()` qui suit le mot-clé `default` est exécutée.

```
switch (nombre) {
  case 1:
    trace ("case 1 est true");
    break
  case 2:
    trace ("case 2 est true");
    break
  case 3:
    trace ("case 3 est true");
    break
  default
    trace ("aucune expression case n'est true")
}
```

Dans l'exemple suivant, il n'y a aucune rupture dans le premier groupe de cas ; donc, si le nombre est 1, A et B sont tous deux envoyés au panneau de sortie.

```
switch (nombre) {
  case 1:
    trace ("A");
  case 2:
    trace ("B");
    break
  default
    trace ("D")
}
```

Consultez également

`===` (égalité stricte), `break`, `case`, `default`, `if`

Classe System

Disponibilité

Flash Player 6.

Description

Il s'agit d'une classe de haut niveau qui contient l'objet capabilities (consultez Objet System.capabilities), l'objet de sécurité (consultez Objet System.security), ainsi que les méthodes, propriétés et gestionnaires d'événement répertoriés ci-dessous.

Méthodes de la classe System

Méthode	Description
`System.setClipboard()`	Remplace le contenu du presse-papiers du système par une chaîne de texte.
`System.showSettings()`	Affiche un panneau Paramètres Flash Player.

Propriétés de la classe System

Méthode	Description
`System.exactSettings`	Spécifie l'utilisation des règles de correspondance de superdomaines ou de domaines exacts lors de l'accès aux paramètres locaux.
`System.useCodepage`	Indique à Flash Player s'il doit utiliser Unicode ou la page de code classique du système d'exploitation exécutant le lecteur pour interpréter des fichiers texte externes.

Gestionnaires d'événement de la classe System

Méthode	Description
`System.onStatus`	Fournit un super gestionnaire d'événement pour un certain nombre d'objets.

System.exactSettings

Disponibilité

Programmation : Flash MX 2004.

Lecture : fichiers SWF publiés pour Flash Player 6 ou ultérieur, lus dans Flash Player 7 ou ultérieur.

Usage

```
System.exactSettings
```

Description

Propriété : spécifie l'utilisation des règles de correspondance de superdomaines ou de domaines exacts lors de l'accès aux paramètres locaux (par exemple, les autorisations d'accès aux caméras ou aux microphones) ou aux données persistantes localement (objets partagés). La valeur par défaut est `true` pour les fichiers publiés pour Flash Player 7 ou une version ultérieure et `false` pour les fichiers publiés pour Flash Player 6.

Si cette valeur est définie sur `true`, les paramètres et les données d'un fichier SWF hébergé sur ici.xyz.com sont stockés sur ici.xyz.com, les paramètres et les données d'un fichier SWF hébergé sur la.xyz.com sont stockés sur la.xyz.com et ainsi de suite. Si cette valeur est définie sur `false`, les paramètres et les données des fichiers SWF hébergés sur ici.xyz.com, la.xyz.com et xyz.com sont partagés et sont tous stockés sur xyz.com.

Si une partie de vos fichiers définit cette propriété sur `false` et d'autres sur `true`, les fichiers SWF dans différents sous-domaines peuvent partager des paramètres et des données. Par exemple, si cette propriété est `false` dans un fichier SWF hébergé sur ici.xyz.com et `true` dans un fichier SWF hébergé sur xyz.com, les deux fichiers utilisent les mêmes paramètres et données - c'est-à-dire, ceux situés sur xyz.com. Si ce n'est pas le comportement que vous souhaitez, vérifiez que vous définissez cette propriété dans chaque fichier pour représenter correctement l'emplacement où les paramètres et les données doivent être stockés.

Si vous souhaitez modifier cette propriété à partir de sa valeur par défaut, émettez la commande `System.exactSettings = false` dans la première image de votre document. Il est impossible de changer la propriété après une activité nécessitant un accès aux paramètres locaux, par exemple `System.ShowSettings()` ou `SharedObject.getLocal()`.

Si vous utilisez `loadMovie()`, `MovieClip.loadMovie` ou `MovieClipLoader.loadClip()` pour charger un fichier SWF dans un autre, tous les fichiers publiés pour Flash Player 7 partagent une valeur unique pour `System.exactSettings` et tous les fichiers publiés pour Flash Player 6 partagent une valeur unique pour `System.exactSettings`. Par conséquent, si vous spécifiez une valeur pour cette propriété dans un fichier publié pour une version particulière de Flash Player, vous devez le faire dans tous les fichiers que vous envisagez de charger. Si vous chargez plusieurs fichiers, le paramètre spécifié dans le dernier fichier chargé supplante tout paramètre précédemment spécifié.

Pour plus d'informations sur l'implémentation de la correspondance des domaines dans Flash, consultez *Fonctions de sécurité de Flash Player*, page 199.

Consultez également

`SharedObject.getLocal()`, `System.showSettings()`

System.onStatus

Disponibilité

Flash Player 6.

Description

Gestionnaire d'événement : fournit un « super » gestionnaire d'événement pour un certain nombre d'objets.

Les objets LocalConnection, NetStream et SharedObject fournissent un gestionnaire d'événement `onStatus` qui utilise un objet d'information pour délivrer des informations, un état ou des messages d'erreur. Pour répondre à ce gestionnaire d'événement, vous devez créer une fonction pour traiter l'objet d'information et vous devez connaître le format et le contenu de l'objet d'information renvoyé.

Outre les méthodes `onStatus` spécifiques fournies pour les objets répertoriés ci-dessus, Flash propose également une « super » fonction appelée `System.onStatus`. Si `onStatus` est invoqué pour un objet particulier avec une propriété `level` de `"erreur"` et qu'aucune fonction n'est affectée pour lui répondre, Flash traite une fonction affectée à `System.onStatus` s'il en existe une.

Remarque : Les classes Camera et Microphone ont aussi des gestionnaires `onStatus`, mais ne transmettent pas des objets d'information avec une propriété level de "erreur". Par conséquent, `System.onStatus` n'est pas appelé si vous ne spécifiez pas de fonction pour ces gestionnaires.

L'exemple suivant illustre la manière de créer des fonctions génériques et spécifiques pour traiter les objets d'information envoyés par la méthode `onStatus`.

```
// Créez une fonction générique
System.onStatus = function(genericError)
{
  // Votre script a plus de sens dans ce cas
  trace("Une erreur s'est produite. Veuillez essayer de nouveau.");
}

// Créez une fonction pour l'objet NetStream
// Si l'objet NetStream renvoie un objet d'information différent
// de celui répertorié ci-dessous, avec une propriété level de "erreur",
// System.onStatus sera invoqué

videoStream_ns.onStatus = function(objetInfo) {
  if (objetInfo.code == "NetStream.Play.StreamNotFound") {
    trace("Fichier vidéo introuvable.");
  }
}
```

Consultez également

`Camera.onStatus`, `LocalConnection.onStatus`, `Microphone.onStatus`, `NetStream.onStatus`, `SharedObject.onStatus`

System.setClipboard()

Disponibilité

Programmation : Flash MX 2004.

Lecture : fichiers SWF publiés pour Flash Player 6 ou ultérieur, lus dans Flash Player 7 ou ultérieur.

Usage

```
System.setClipboard(chaîne)
```

Paramètres

chaîne Une chaîne de caractères de texte ordinaire à placer sur le presse-papiers du système, remplaçant son contenu actuel (s'il y en a un). Si vous affectez une chaîne littérale, par opposition à une variable de type chaîne, mettez cette chaîne littérale entre guillemets.

Renvoie

Une valeur booléenne : `true` si le texte a été placé avec succès dans le presse-papiers, `false` dans les autres cas.

Description

Méthode : remplace le contenu du presse-papiers du système par une chaîne de texte spécifique.

System.showSettings()

Disponibilité

Flash Player 6.

Usage

```
System.showSettings([panneau])
```

Paramètres

panneau Un nombre facultatif spécifiant le panneau Paramètres Flash Player à afficher, conformément au tableau suivant :

Valeur transmise pour *panneau*	Panneau de paramètres affiché
Aucun (paramètre omis) ou valeur non supportée	N'importe quel panneau ouvert la dernière fois que l'utilisateur a fermé le panneau Paramètres Flash Player
0	Contrôle de l'accès
1	Enregistrement local
2	Microphone
3	Caméra

Renvoie

Rien.

Description

Méthode : affiche le panneau Paramètres Flash Player spécifié permettant aux utilisateurs de procéder aux actions suivantes :

- Autoriser ou refuser l'accès à la caméra et au microphone
- Spécifier l'espace disponible sur le disque local pour les objets partagés
- Sélectionner une caméra et un microphone par défaut
- Spécifier les paramètres du microphone et de la fonction de suppression de l'écho

Par exemple, si votre application nécessite l'utilisation d'une caméra, vous pouvez dire à l'utilisateur de sélectionner Autoriser dans le panneau des paramètres de contrôle de l'accès, pour ensuite donner la commande `System.showSettings(0)`. (Assurez-vous que la scène mesure au moins 215 x 138 pixels ; il s'agit de la taille minimale requise pour que Flash puisse afficher le panneau.)

Consultez également

`Camera.get()`, `Microphone.get()`, `SharedObject.getLocal()`

System.useCodepage

Disponibilité

Flash Player 6.

Usage

```
System.useCodepage
```

Description

Propriété : une valeur booléenne indiquant à Flash Player s'il faut utiliser Unicode ou la page de code classique du système d'exploitation exécutant le lecteur pour interpréter les fichiers texte externes. La valeur par défaut de `system.useCodepage` est `false`.

- Lorsque la propriété est définie sur `false`, Flash Player interprète les fichiers texte externes au format Unicode. (Ces fichiers doivent être encodés au format Unicode lorsque vous les enregistrez.)
- Lorsque la propriété est définie sur la valeur `true`, Flash Player interprète les fichiers texte externes à l'aide de la page de code classique du système d'exploitation exécutant le lecteur.

Le texte que vous incluez ou chargez en tant que fichier externe (grâce à la commande `#include`, aux actions `loadVariables()` ou `getURL` ou aux objets LoadVars ou XML) doit être encodé au format Unicode lorsque vous enregistrez le fichier texte afin que Flash Player puisse le reconnaître en tant qu'Unicode. Afin d'encoder les fichiers externes au format Unicode, enregistrez-les dans une application prenant en charge Unicode, comme Notepad sous Windows 2000.

Si vous incluez ou chargez des fichiers texte externes qui ne sont pas au format Unicode, vous devez définir `system.useCodepage` sur la valeur `true`. Ajoutez le code suivant en tant que première ligne de code dans la première image du fichier SWF qui charge les données :

```
system.useCodepage = true;
```

Avec ce code, Flash Player interprète le texte externe en utilisant la page de code classique du système d'exploitation qui exécute Flash Player. En règle générale, il s'agit du jeu de caractères CP1252 pour un système d'exploitation Windows anglais et du jeu de caractères Shift-JIS pour un système d'exploitation japonais. Si vous définissez `system.useCodepage` sur la valeur `true`, Flash Player 6 ou version ultérieure considère le texte de la même façon que Flash Player 5. (Flash Player 5 considérait tous les textes comme s'ils étaient au format de la page de code traditionnelle du système d'exploitation exécutant le lecteur.)

Si vous définissez `system.useCodepage` sur la valeur `true`, rappelez-vous que la page de code traditionnelle du système d'exploitation exécutant le lecteur doit inclure les caractères utilisés dans votre fichier texte externe pour pouvoir afficher le texte. Par exemple, si vous chargez un fichier texte externe contenant des caractères chinois, ceux-ci ne s'affichent pas sur un système qui utilise le jeu de caractères CP1252, cette page de code n'incluant pas les caractères chinois.

Afin de vous assurer que les utilisateurs de toutes les plates-formes peuvent visualiser les fichiers texte externes utilisés dans vos fichiers SWF, vous devez encoder tous les fichiers texte externes au format Unicode et laisser `System.useCodepage` sur la valeur `false` par défaut. De cette manière, Flash Player 6 et ultérieure interprète le texte en tant que Unicode.

Objet System.capabilities

Disponibilité

Flash Player 6.

Description

Vous pouvez utiliser l'objet System.capabilities pour déterminer les capacités du système et du lecteur hébergeant un fichier SWF. Ceci vous permet d'adapter le contenu à différents formats. Par exemple, l'écran d'un téléphone cellulaire (noir et blanc, 100 pixels carrés) est différent de l'écran couleur de 1000 pixels carrés d'un ordinateur. Pour fournir un contenu approprié au plus grand nombre possible d'utilisateurs, vous pouvez utiliser l'objet System.capabilities pour déterminer le type de périphérique dont dispose un utilisateur. En fonction des capacités du périphérique, vous pouvez indiquer au serveur d'envoyer des fichiers SWF différents ou indiquer au fichier SWF de modifier sa présentation.

Vous pouvez envoyer des informations sur les capacités en utilisant une méthode HTTP `GET` ou `POST`. L'exemple suivant est un exemple d'une chaîne de serveur pour un périphérique qui n'offre pas de support MP3 et dispose d'un écran de 400 x 200 pixels, 8 x 4 centimètres :

```
"A=t&SA=t&SV=t&EV=t&MP3=t&AE=t&VE=t&ACC=f&PR=t&SP=t&SB=f&DEB=t&V=WIN%207%2C0%2
  C0%2C226&M=Macromedia%20Windows&R=1152x864&DP=72&COL=color&AR=1.0&OS=Window
  s%20XP&L=en&PT=External&AVD=f&LFD=f"
```

Propriétés de l'objet System.capabilities

Propriété	Description	Chaîne du serveur
`System.capabilities.avHardwareDisable`	Lecture seule : indique si la caméra et le microphone de l'utilisateur sont activés ou désactivés.	`AVD`
`System.capabilities.hasAccessibility`	Indique si le système exécutant le lecteur supporte la communication entre Flash Player et les aides d'accessibilité.	`ACC`
`System.capabilities.hasAudio`	Indique si le système exécutant le lecteur possède des capacités audio.	`A`
`System.capabilities.hasAudioEncoder`	Indique si le lecteur s'exécute sur un système pouvant encoder un flux audio tel que celui provenant d'un microphone.	`AE`
`System.capabilities.hasEmbeddedVideo`	Indique si le système exécutant le lecteur supporte la vidéo intégrée.	`EV`
`System.capabilities.hasMP3`	Indique si le système exécutant le lecteur est équipé d'un décodeur MP3.	`MP3`
`System.capabilities.hasPrinting`	Indique si le lecteur fonctionne avec un système supportant l'impression.	`PR`

Propriété	Description	Chaîne du serveur
System.capabilities.hasScreenBroadcast	Indique si le lecteur supporte le développement d'applications de diffusion à l'écran à lancer via le serveur de communication Flash.	SB
System.capabilities.hasScreenPlayback	Indique si le lecteur supporte la lecture d'applications de diffusion à l'écran lancées via le serveur de communication Flash.	SP
System.capabilities.hasStreamingAudio	Indique si le lecteur est capable de lire des données audio en continu.	SA
System.capabilities.hasStreamingVideo	Indique si le lecteur est capable de lire des données vidéo en continu.	SV
System.capabilities.hasVideoEncoder	Indique si le lecteur peut encoder un flux vidéo tel que celui provenant d'une webcam.	VE
System.capabilities.isDebugger	Indique si le lecteur est une version officielle ou une version de débogage spéciale.	DEB
System.capabilities.language	Indique la langue du système exécutant le lecteur.	L
System.capabilities.localFileReadDisable	Lecture seule : indique si le lecteur doit tenter de lire les fichiers (y compris le premier fichier SWF ayant servi au lancement du lecteur) stockés sur le disque dur de l'utilisateur.	LFD
System.capabilities.manufacturer	Indique le fabricant de Flash Player.	M
System.capabilities.os	Indique le système d'exploitation hébergeant Flash Player.	OS
System.capabilities.pixelAspectRatio	Indique le rapport largeur/hauteur, en pixels, de l'écran.	AR
System.capabilities.playerType	Indique le type de lecteur : autonome, externe, plug-in ou ActiveX.	PT
System.capabilities.screenColor	Indique si l'écran est en couleur, noir et blanc ou à nuances de gris.	COL
System.capabilities.screenDPI	Indique le nombre de points par pouce de la résolution de l'écran, en pixels.	DP
System.capabilities.screenResolutionX	Indique la taille horizontale de l'écran.	R
System.capabilities.screenResolutionY	Indique la taille verticale de l'écran.	R

Propriété	Description	Chaîne du serveur
System.capabilities.serverString	Une chaîne d'URL encodée qui spécifie les valeurs pour chaque propriété System.capabilities.	N/A
System.capabilities.version	Une chaîne contenant la version de Flash Player et les informations de plate-forme.	V

System.capabilities.avHardwareDisable

Disponibilité

Flash Player 7.

Usage

```
System.capabilities.avHardwareDisable
```

Description

Propriété (lecture seule) : une valeur booléenne indiquant si la caméra et le microphone de l'utilisateur sont activés ou désactivés.

Consultez également

Camera.get(), Microphone.get(), System.showSettings()

System.capabilities.hasAccessibility

Disponibilité

Flash Player 6 version 65.

Usage

```
System.capabilities.hasAccessibility
```

Description

Propriété : une valeur booléenne qui indique si le lecteur fonctionne sur un environnement supportant la communication entre Flash Player et les aides d'accessibilité. La chaîne du serveur est ACC.

Consultez également

Accessibility.isActive(), Accessibility.updateProperties(), _accProps

System.capabilities.hasAudio

Disponibilité

Flash Player 6.

Usage

```
System.capabilities.hasAudio
```

Description

Propriété : une valeur booléenne indiquant si le système exécutant le lecteur possède des capacités audio. La chaîne du serveur est `A`.

System.capabilities.hasAudioEncoder

Disponibilité

Flash Player 6.

Usage

```
System.capabilities.hasAudioEncoder
```

Description

Propriété : une valeur booléenne indiquant si le lecteur peut encoder un flux audio, tel que celui provenant d'un microphone. La chaîne du serveur est `AE`.

System.capabilities.hasEmbeddedVideo

Disponibilité

Flash Player 6.

Usage

```
System.capabilities.hasEmbeddedVideo
```

Description

Propriété : une valeur booléenne indiquant si le système exécutant le lecteur supporte la vidéo intégrée. La chaîne du serveur est `EV`.

System.capabilities.hasMP3

Disponibilité

Flash Player 6.

Usage

```
System.capabilities.hasMP3
```

Description

Propriété : une valeur booléenne indiquant si le système exécutant le lecteur est équipé d'un décodeur MP3. La chaîne du serveur est `MP3`.

System.capabilities.hasPrinting

Disponibilité

Flash Player 6.

Usage

```
System.capabilities.hasPrinting
```

Description

Propriété : une valeur booléenne indiquant si le lecteur fonctionne avec un système prenant en charge l'impression. La chaîne du serveur est `PR`.

System.capabilities.hasScreenBroadcast

Disponibilité

Flash Player 6.

Usage

```
System.capabilities.hasScreenBroadcast
```

Description

Propriété : une valeur booléenne indiquant si le lecteur prend en charge le développement d'applications de diffusion à l'écran exécutées via Flash Communication Server. La chaîne du serveur est `SB`.

System.capabilities.hasScreenPlayback

Disponibilité

Flash Player 6.

Usage

```
System.capabilities.hasScreenPlayback
```

Description

Propriété : une valeur booléenne indiquant si le lecteur prend en charge la lecture d'applications de diffusion à l'écran exécutées via Flash Communication Server. La chaîne du serveur est `SP`.

System.capabilities.hasStreamingAudio

Disponibilité

Flash Player 6.

Usage

```
System.capabilities.hasStreamingAudio
```

Description

Propriété : une valeur booléenne indiquant si le lecteur est capable de lire des données audio en continu. La chaîne du serveur est `SA`.

System.capabilities.hasStreamingVideo

Disponibilité

Flash Player 6.

Usage

```
System.capabilities.hasStreamingVideo
```

Description

Propriété : une valeur booléenne indiquant si le lecteur est capable de lire des données vidéo en continu. La chaîne du serveur est `SV`.

System.capabilities.hasVideoEncoder

Disponibilité

Flash Player 6.

Usage

```
System.capabilities.hasVideoEncoder
```

Description

Propriété : une valeur booléenne indiquant si le lecteur peut encoder un flux vidéo, tel que celui provenant d'une webcam. La chaîne du serveur est `VE`.

System.capabilities.isDebugger

Disponibilité

Flash Player 6.

Usage

```
System.capabilities.isDebugger
```

Description

Propriété : une valeur booléenne indiquant si le lecteur est une version officielle (`false`) ou une version spéciale de débogage (`true`). La chaîne du serveur est `DEB`.

System.capabilities.language

Disponibilité

Flash Player 6.

Usage

```
System.capabilities.language
```

Description

Propriété : indique la langue du système exécutant le lecteur. Cette propriété se présente sous la forme d'un code de langue constitué de deux lettres minuscules, conformément à la norme ISO 639-1, et d'un code de pays facultatif constitué de deux lettres majuscules, conformément à la norme ISO 3166. Les codes représentent la langue du système exécutant le lecteur. Les langues sont désignées par les balises en anglais. Par exemple, « fr » signifie français.

Langue	Balise	Pays et balises supportés
Tchèque	`cs`	
Danois	`da`	
Hollandais	`nl`	
Anglais	`en`	
Finlandais	`fi`	
Français	`fr`	
Allemand	`de`	
Hongrois	`hu`	
Italien	`it`	
Japonais	`ja`	
Coréen	`ko`	
Norvégien	`no`	
Autre/inconnu	xu	
Polonais	`pl`	
Portugais	`pt`	
Russe	`ru`	
Chinois simplifié	`zh`	République populaire de Chine (chinois simplifié) `zh-CN`
Espagnol	`es`	
Suédois	`sv`	
Chinois traditionnel	`zh`	Taïwan (chinois traditionnel) `zh-TW`
Turc	`tr`	

System.capabilities.localFileReadDisable

Disponibilité

Flash Player 7.

Usage

```
System.capabilities.localFileReadDisable
```

Description

Propriété (lecture seule) : une valeur booléenne indiquant si Flash Player doit tenter de lire les fichiers (y compris le premier fichier SWF ayant servi au lancement de Flash Player) stockés sur le disque dur de l'utilisateur.

System.capabilities.manufacturer

Disponibilité

Flash Player 6.

Usage

```
System.capabilities.manufacturer
```

Description

Propriété : une chaîne indiquant le fabricant de Flash Player au format « `Macromedia`*`nomSE`* » (*`nomSE`* pouvant être « `Windows` », « `Macintosh` », « `Linux` » ou « `autre nomSE` »). La chaîne du serveur est `M`.

System.capabilities.os

Disponibilité

Flash Player 6.

Usage

```
System.capabilities.os
```

Description

Propriété : une chaîne indiquant le système d'exploitation en cours d'utilisation. La propriété `os` peut renvoyer les chaînes suivantes : « `Windows XP` », « `Windows 2000` », « `Windows NT` », « `Windows 98/ME` », « `Windows 95` », « `Windows CE` » (disponible uniquement dans le SDK Flash Player, pas dans la version de bureau), « `Linux` » et « `MacOS` ». La chaîne du serveur est `OS`.

System.capabilities.pixelAspectRatio

Disponibilité

Flash Player 6.

Usage

```
System.capabilities.pixelAspectRatio
```

Description

Propriété : un entier indiquant le rapport largeur/hauteur, en pixels, de l'écran. La chaîne du serveur est `AR`.

System.capabilities.playerType

Disponibilité

Flash Player 7.

Usage

```
System.capabilities.playerType
```

Description

Propriété : une chaîne indiquant le type de lecteur. Cette propriété peut avoir la valeur `"StandAlone"`, `"External"`, `"PlugIn"` ou `"ActiveX"`. La chaîne du serveur est `PT`.

System.capabilities.screenColor

Disponibilité

Flash Player 6.

Usage

```
System.capabilities.screenColor
```

Description

Propriété : indique si l'écran est en couleur (`color`), nuances de gris (`gray`) ou noir et blanc (`bw`). La chaîne du serveur est `COL`.

System.capabilities.screenDPI

Disponibilité

Flash Player 6.

Usage

```
System.capabilities.screenDPI
```

Description

Propriété : indique la résolution en points par pouce (ppp) de l'écran, en pixels. La chaîne du serveur est `DP`.

System.capabilities.screenResolutionX

Disponibilité

Flash Player 6.

Usage

```
System.capabilities.screenResolutionX
```

Description

Propriété : un entier indiquant la résolution horizontale maximum de l'écran. La chaîne du serveur est `R` (qui renvoie à la fois la largeur et la hauteur de l'écran).

System.capabilities.screenResolutionY

Disponibilité

Flash Player 6.

Usage

```
System.capabilities.screenResolutionY
```

Description

Propriété : un entier indiquant la résolution verticale maximum de l'écran. La chaîne du serveur est `R` (qui renvoie à la fois la largeur et la hauteur de l'écran).

System.capabilities.serverString

Disponibilité

Flash Player 6.

Usage

```
System.capabilities.serverString
```

Description

Propriété : une chaîne d'URL encodée spécifiant les valeurs pour chaque propriété `System.capabilities`, comme dans cet exemple :

```
A=t&SA=t&SV=t&EV=t&MP3=t&AE=t&VE=t&ACC=f&PR=t&SP=t&SB=f&DEB=t&V=WIN%207%2C0%2C
0%2C226&M=Macromedia%20Windows&R=1152x864&DP=72&COL=color&AR=1.0&OS=Windows%20
XP&L=en&PT=External&AVD=f&LFD=f
```

System.capabilities.version

Disponibilité

Flash Player 6.

Usage

```
System.capabilities.version
```

Description

Propriété : une chaîne contenant les informations sur la version et la plate-forme de Flash Player, par exemple `"WIN 7,0,0,231"`. La chaîne du serveur est `V`.

Objet System.security

Disponibilité

Flash Player 6.

Description

Cet objet contient des méthodes permettant de spécifier comment les fichiers SWF de domaines différents peuvent communiquer entre eux.

Méthodes de l'objet System.security

Méthode	Description
`System.security.allowDomain()`	Permet aux fichiers SWF des domaines identifiés d'accéder aux objets et aux variables du fichier SWF appelant ou de tout autre fichier SWF issu du même domaine que le fichier SWF appelant.
`System.security.allowInsecureDomain()`	Permet aux fichiers SWF des domaines identifiés d'accéder aux objets et aux variables du fichier SWF appelant, qui est hébergé à l'aide du protocole HTTPS.

System.security.allowDomain()

Disponibilité

Flash Player 6 ; comportement modifié dans Flash Player 7.

Usage

```
System.security.allowDomain("domaine1", "domaine2, ... domaineN")
```

Paramètres

domaine1, domaine2, ... domaineN Chaînes précisant les domaines qui ont accès aux objets et aux variables dans le fichier contenant l'appel `System.Security.allowDomain()`. Les domaines peuvent être aux formats suivants :

- « `domaine.com` »
- « `http://domaine.com` »
- « `http://adresseIP` »

Description

Méthode : permet aux fichiers SWF des domaines identifiés d'accéder aux objets et aux variables du fichier SWF appelant ou de tout autre fichier SWF issu du même domaine que le fichier SWF appelant.

Dans les fichiers lus dans Flash Player 7 ou version ultérieure, les paramètres affectés doivent suivre les règles d'appellation des domaines exacts. Par exemple, pour permettre l'accès des fichiers SWF hébergés sur www.domaine.com ou sur store.domaine.com, les deux noms de domaines doivent être transmis :

```
// Pour Flash Player 6
System.security.allowDomain("domaine.com");
// Commandes correspondantes pour autoriser l'accès à partir de fichiers SWF
// exécutés dans Flash Player 7 ou ultérieur
System.security.allowDomain("www.domaine.com". "store.domaine.com");
```

De plus, pour les fichiers exécutés dans Flash Player 7 ou version ultérieure, vous ne pouvez pas utiliser cette méthode pour permettre aux fichiers SWF hébergés à l'aide d'un protocole sécurisé (HTTPS) d'autoriser l'accès depuis des fichiers SWF hébergés dans des protocoles non sécurisés : vous devez plutôt utiliser `System.security.allowInsecureDomain()`.

Exemple

Le fichier SWF situé à l'adresse http://www.macromedia.com/MovieA.swf contient les lignes suivantes.

```
System.security.allowDomain("www.shockwave.com");
loadMovie("http://www.shockwave.com/MovieB.swf", _root.mon_mc);
```

Comme MovieA contient la commande `allowDomain()`, MovieB peut accéder aux objets et aux variables de MovieA. Si MovieA ne contenait pas cette commande, la sécurité Flash empêcherait MovieA d'accéder aux objets et aux variables de MovieB.

System.security.allowInsecureDomain()

Disponibilité

Flash Player 7.

Usage

```
System.Security.allowInsecureDomain("domaine")
```

Paramètres

domaine Un nom de domaine exact, par exemple, « www.monNomDeDomaine.com » ou « store.monNomDeDomaine.com ».

Renvoie

Rien.

Description

Méthode : permet aux fichiers SWF des domaines identifiés d'accéder aux objets et aux variables du fichier SWF appelant, qui est hébergé à l'aide du protocole HTTPS.

Par défaut, les fichiers SWF hébergés utilisant le protocole HTTPS ne sont accessibles que par d'autres fichiers SWF hébergés utilisant le protocole HTTPS. Cette implémentation garantit l'intégrité fournie par le protocole HTTPS.

Macromedia déconseille l'utilisation de cette méthode pour remplacer le comportement par défaut car elle compromet la sécurité HTTPS. Toutefois, vous pouvez le faire, par exemple, si vous devez autoriser l'accès aux fichiers HTTPS publiés pour Flash Player 7 ou version ultérieure à partir des fichiers HTTP publiés pour Flash Player 6.

Un fichier SWF publié pour Flash Player 6 peut utiliser `System.security.allowDomain()` pour autoriser l'accès de HTTP à HTTPS. Cependant, la sécurité étant implémentée de manière différente dans Flash Player 7, vous devez utiliser `System.Security.allowInsecureDomain()` pour autoriser un tel accès aux fichiers SWF publiés pour Flash Player 7 ou une version ultérieure.

Exemple

Dans cet exemple, vous hébergez un test de mathématiques sur un domaine sécurisé de telle sorte que seuls les étudiants enregistrés peuvent y accéder. Vous avez également développé un certain nombre de fichiers SWF qui illustrent certains concepts, que vous hébergez sur un domaine non sécurisé. Vous voulez que les étudiants accèdent au test à partir du fichier SWF contenant des informations sur un concept.

```
// Ce fichier SWF est situé à l'adresse https://
  monSiteEducatif.quelquepart.com/testMath.swf
// Les fichiers des concepts se trouvent sur http://mon
  SiteEducatif.quelquepart.com
System.Security.allowInsecureDomain("monSiteEducatif.quelquepart.com")
```

Consultez également

`System.security.allowDomain()`, `System.exactSettings`

targetPath

Disponibilité

Flash Player 5.

Usage

```
targetpath(objetClip)
```

Paramètres

objetClip Référence (par exemple, _root ou _parent) au clip dont le chemin cible est en cours de récupération.

Renvoie

Une chaîne contenant le chemin cible du clip spécifié.

Description

Fonction : renvoie une chaîne contenant le chemin cible de *objetClip*. Le chemin cible est renvoyé avec la notation point. Pour récupérer le chemin cible dans la notation à barre oblique, utilisez la propriété _target.

Exemple

Cet exemple affiche le chemin cible d'un clip dès son chargement.

```
onClipEvent(load) {
  trace(cheminCible(this));
}
```

Consultez également

eval()

tellTarget

Disponibilité

Flash Player 3. (Déconseillé dans Flash 5 ; l'utilisation de la notation à point et de l'action `with` est recommandée.)

Usage

```
tellTarget("cible") {
  instruction(s);
}
```

Paramètres

`cible` Une chaîne spécifiant le chemin cible du scénario à contrôler.

`instruction(s)` Les instructions à exécuter si la condition est évaluée comme `true`.

Renvoie

Rien.

Description

Action déconseillée : applique les instructions spécifiées dans le paramètre *`instructions`* au scénario spécifié dans le paramètre *`cible`*. L'action `tellTarget` est utile pour les contrôles de navigation. Affectez `tellTarget` aux boutons qui arrêtent ou démarrent les clips n'importe où sur la scène. Vous pouvez également envoyer les clips à une image particulière de ce clip. Par exemple, vous pouvez affecter `tellTarget` aux boutons qui arrêtent ou démarrent les clips sur la scène ou qui envoient les clips dans une image particulière.

Dans Flash 5 ou version ultérieure, vous pouvez utiliser la notation à point au lieu de l'action `tellTarget`. Vous pouvez utiliser l'action `with` pour communiquer plusieurs actions au même scénario. Vous pouvez utiliser l'action `with` pour cibler n'importe quel objet, alors que l'action `tellTarget` ne peut cibler que des clips.

Exemple

Cette instruction `tellTarget` contrôle l'occurrence de clip `balle` du scénario principal. L'image 1 de l'occurrence `balle` est vide et possède une action `stop()` invisible sur la scène. Lorsque l'utilisateur clique sur le bouton avec l'action suivante, `tellTarget` indique à la tête de lecture du clip `balle` d'atteindre l'image 2 où commence l'animation.

```
on(release) {
  tellTarget("balle") {
    gotoAndPlay(2);
  }
}
```

L'exemple suivant utilise la notation à point pour obtenir les mêmes résultats.

```
on(release) {
  balle.gotoAndPlay(2);
}
```

Si vous avez besoin de communiquer plusieurs commandes à l'occurrence `balle`, vous pouvez utiliser l'action `with`, comme dans l'instruction suivante.

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

Consultez également

`with`

Classe TextField

Disponibilité

Flash Player 6.

Description

Tous les champs de texte dynamique et de saisie d'un fichier SWF sont des occurrences de la classe TextField. Vous pouvez donner un nom d'occurrence à un champ de texte dans l'inspecteur des propriétés et utiliser les méthodes et les propriétés de la classe TextField pour le manipuler avec ActionScript. Les noms d'occurrence TextField sont affichés dans l'explorateur d'animations et dans la boîte de dialogue Insérer un chemin cible du panneau Actions.

La classe TextField hérite de la Classe Object.

Pour créer un champ de texte de manière dynamique, vous pouvez utiliser `MovieClip.createTextField()`.

Méthodes de la classe TextField

Méthode	Description
`TextField.addListener()`	Enregistre un objet pour la réception de la notification lorsque les événements `onChanged` et `onScroller` sont invoqués.
`TextField.getFontList`	Renvoie les noms des polices sur le système hôte du lecteur sous forme de tableau.
`TextField.getDepth()`	Renvoie la profondeur d'un champ de texte.
`TextField.getNewTextFormat()`	Lit le format de texte par défaut affecté au texte nouvellement inséré.
`TextField.getTextFormat()`	Renvoie un objet TextFormat contenant des informations de format sur une partie ou sur l'ensemble du texte contenu dans un champ de texte.
`TextField.removeListener`	Supprime un objet d'écoute.
`TextField.removeTextField()`	Supprime un champ de texte créé avec `MovieClip.createTextField()`.
`TextField.replaceSel()`	Remplace la sélection actuelle.

Méthode	Description
TextField.setNewTextFormat	Définit un objet TextFormat pour le texte inséré par un utilisateur ou par une méthode.
TextField.setTextFormat	Définit un objet TextFormat pour une plage de texte spécifiée dans un champ de texte.

Propriétés de la classe TextField

Propriété	Description
TextField._alpha	La valeur de transparence d'une occurrence de champ de texte.
TextField.autoSize	Contrôle le dimensionnement et l'alignement automatique des champs de texte.
TextField.background	Indique si le champ de texte a un remplissage d'arrière-plan.
TextField.backgroundColor	Indique la couleur du remplissage d'arrière-plan.
TextField.border	Indique si le champ de texte a une bordure.
TextField.borderColor	Indique la couleur de la bordure.
TextField.bottomScroll	La dernière ligne visible dans un champ de texte. Lecture seule.
TextField.embedFonts	Indique si le champ de texte utilise des polices vectorielles intégrées ou des polices de périphérique.
TextField._height	La hauteur d'une occurrence de champ de texte, en pixels. Cela n'affecte que le cadre de délimitation du champ de texte, pas l'épaisseur de la bordure ou la taille du texte.
TextField._highquality	Indique la qualité de rendu du fichier SWF.
TextField.hscroll	Indique la valeur de défilement horizontal d'un champ de texte.
TextField.html	Indique la position de défilement maximum actuelle d'un champ de texte.
TextField.htmlText	Contient la représentation HTML du contenu d'un champ de texte.
TextField.length	Le nombre de caractères d'un champ de texte. Lecture seule.
TextField.maxChars	La quantité maximum de caractères qu'un champ de texte peut contenir.
TextField.maxhscroll	La valeur maximum de TextField.hscroll. Lecture seule.
TextField.maxscroll	La valeur maximum de TextField.scroll. Lecture seule.
TextField.menu	Associe un objet ContextMenu à un champ de texte.
TextField.mouseWheelEnabled	Indique si Flash Player doit automatiquement faire défiler les champs de texte de plusieurs lignes lorsque le pointeur de la souris se trouve sur un champ de texte et que l'utilisateur actionne la molette de la souris.
TextField.multiline	Indique si le champ de texte contient plusieurs lignes.
TextField._name	Le nom d'occurrence d'une occurrence de champ de texte.

Propriété	Description
TextField._parent	Une référence à l'occurrence parent de cette occurrence ; de type Button ou MovieClip.
TextField.password	Indique si un champ de texte masque les caractères saisis.
TextField._quality	Indique la qualité du rendu d'un fichier SWF.
TextField.restrict	Le jeu de caractères qu'un utilisateur peut rentrer dans un champ de texte.
TextField._rotation	Le degré de rotation d'une occurrence de champ de texte.
TextField.scroll	Indique la position de défilement actuelle d'un champ de texte.
TextField.selectable	Indique si un champ de texte est sélectionnable.
TextField._soundbuftime	La durée nécessaire de mise en tampon d'un son avant sa lecture en flux continu.
TextField.tabEnabled	Indique si un clip est inclus dans l'ordre de tabulation automatique.
TextField.tabIndex	Indique l'ordre de tabulation d'un objet.
TextField._target	Le chemin cible de l'occurrence de champ de texte spécifiée. Lecture seule.
TextField.text	Le texte du champ de texte.
TextField.textColor	La couleur du texte d'un champ de texte.
TextField.textHeight	La hauteur du cadre de délimitation du champ de texte.
TextField.textWidth	La largeur du cadre de délimitation du champ de texte.
TextField.type	Indique si un champ de texte est un champ de saisie ou un champ de texte dynamique.
TextField._url	L'URL du fichier SWF créateur de l'occurrence de champ de texte. Lecture seule.
TextField.variable	Le nom de variable associé au champ de texte.
TextField._visible	Une valeur booléenne déterminant si l'occurrence d'un champ de texte est masquée ou visible.
TextField._width	La largeur d'une occurrence de champ de texte, en pixels. Cela n'affecte que le cadre de délimitation du champ de texte, pas l'épaisseur de la bordure ou la taille du texte.
TextField.wordWrap	Indique si le texte du champ de texte passe automatiquement à la ligne.
TextField._x	La coordonnée *x* d'une occurrence de champ de texte.
TextField._xmouse	La coordonnée *x* du pointeur par rapport à une occurrence de champ de texte. Lecture seule.
TextField._xscale	La valeur spécifiant le pourcentage de redimensionnement horizontal d'une occurrence de champ de texte.
TextField._y	La coordonnée *y* d'une occurrence de champ de texte.

Propriété	Description
TextField._ymouse	La coordonnée *y* du pointeur par rapport à une occurrence de champ de texte. Lecture seule.
TextField._yscale	La valeur spécifiant le pourcentage de redimensionnement vertical d'une occurrence de champ de texte.

Gestionnaires d'événement de la classe TextField

Gestionnaire d'événement	Description
TextField.onChanged	Invoqué lorsque le contenu du champ de texte change.
TextField.onKillFocus	Invoqué lorsqu'un champ de texte perd le focus.
TextField.onScroller	Invoqué lorsque l'une des propriétés de défilement du champ de texte change.
TextField.onSetFocus	Invoqué lorsqu'un champ de texte reçoit le focus.

Ecouteurs de la classe TextField

Méthode	Description
TextField.onChanged	Notifié lorsque le contenu du champ de texte change.
TextField.onScroller	Notifié lorsque la propriété `scroll` ou `maxscroll` d'un champ de texte change.

TextField.addListener()

Disponibilité

Flash Player 6.

Usage

mon_txt.addListener(*écouteur*)

Paramètres

écouteur Un objet avec un gestionnaire d'événement onChanged ou onScroller.

Renvoie

Rien.

Description

Méthode : enregistre un objet pour la réception de la notification lorsque les gestionnaires d'événements onChanged et onScroller sont invoqués. Lorsqu'un champ de texte change ou défile, les gestionnaires d'événement TextField.onChanged et TextField.onScroller sont invoqués, suivis des gestionnaires d'événement onChanged et onScroller de tous les objets enregistrés en tant qu'écouteurs. Les objets multiples peuvent être enregistrés en tant qu'écouteurs.

Pour supprimer un objet d'écoute d'un champ de texte, appelez TextField.removeListener.

Une référence à l'occurrence du champ de texte est affectée en tant que paramètre aux gestionnaires d'événement onScroller et onChanged par la source de l'événement. Vous pouvez capturer ces données en plaçant un paramètre dans la méthode du gestionnaire d'événement. Par exemple, le code suivant utilise txt en tant que paramètre affecté au gestionnaire d'événement onScroller. Puis, le paramètre est utilisé dans une instruction trace afin d'envoyer le nom de l'occurrence du champ de texte vers le panneau de sortie.

```
monChampDeTexte.onScroller = function (txt) {
  trace (txt._nom + "a changé");
};
```

Exemple

L'exemple suivant définit un gestionnaire onChange pour le champ de texte de saisie monTexte. Il définit un nouvel objet d'écoute, monEcouteur, et un gestionnaire onChanged pour ce même objet. Ce gestionnaire est invoqué lorsque le champ de texte monTexte est changé. La dernière ligne de code appelle TextField.addListener afin d'enregistrer l'objet d'écoute monEcouteur avec le champ de texte monTexte de manière à notifier les changements de monTexte.

```
monTexte.onChanged = function (txt) {
  trace (txt._nom + "a changé");
};
monEcouteur = new Object();
monEcouteur.onChanged = function (txt) {
  trace(txt._nom + " a changé et notifié monEcouteur");
};

monTexte.addListener(monEcouteur);
```

Consultez également

TextField.onChanged, TextField.onScroller, TextField.removeListener

TextField._alpha

Disponibilité

Flash Player 6.

Usage

mon_txt._alpha

Description

Propriété : définit et récupère la transparence alpha (valeur) du champ de texte spécifié par *mon_txt*. Les valeurs valides vont de 0 (transparence complète) à 100 (opacité complète). La valeur par défaut est 100.

Exemple

Le code suivant définit la propriété _alpha d'un champ de texte nommé text1_txt sur 30 % lorsque l'utilisateur clique sur le bouton :

```
on(release) {
  texte1_txt._alpha = 30;
}
```

Consultez également

Button._alpha, MovieClip._alpha

TextField.autoSize

Disponibilité

Flash Player 6.

Usage

mon_txt.autoSize

Description

Propriété : contrôle le dimensionnement et l'alignement automatique des champs de texte. Les valeurs acceptables pour autoSize sont "none" (valeur par défaut), "left", "right" et "center". Lorsque vous paramétrez la propriété autoSize, true est synonyme de "left" et false est synonyme de "none".

Les valeurs d'autoSize, multiline et wordWrap déterminent l'agrandissement ou la réduction d'un champ de texte vers la gauche, la droite ou le bas. Vous pouvez utiliser le code suivant et entrer des valeurs différentes pour autoSize, multiline et wordWrap pour voir comment le champ est redimensionné lorsque ces valeurs changent.

```
createTextField("mon_txt", 1, 0, 0, 200, 20);
with (mon_txt) {
  border = true;
  borderColor = 0x000000;
  multiline = false
  wordWrap = false;
  autoSize = "none";
  text = "Une grande partie du texte ne rentre pas dans le champ ";
}
```

Exemple

Les instructions suivantes définissent la propriété autosize du champ de texte mon_txt sur "center".

```
mon_txt.autosize = "center";
```

TextField.background

Disponibilité

Flash Player 6.

Usage

mon_txt.background

Description

Propriété : si true, le champ de texte a un remplissage d'arrière-plan. Si false, le champ de texte n'a pas de remplissage d'arrière-plan.

TextField.backgroundColor

Disponibilité

Flash Player 6.

Usage

mon_txt.backgroundColor

Description

Propriété : la couleur d'arrière-plan du champ de 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 est seulement visible si le champ de texte a une bordure.

Consultez également

TextField.background

TextField.border

Disponibilité

Flash Player 6.

Usage

mon_txt.border

Description

Propriété : si true, le champ de texte a une bordure. Si false, le champ de texte n'a pas de bordure.

TextField.borderColor

Disponibilité

Flash Player 6.

Usage

mon_txt.borderColor

Description

Propriété : la couleur de bordure du champ de texte (la valeur par défaut est 0x000000 – le noir). Cette propriété peut être récupérée ou définie, même s'il n'y a actuellement aucune bordure.

Consultez également

TextField.border

TextField.bottomScroll

Disponibilité

Flash Player 6.

Usage

mon_txt.bottomScroll

Description

Propriété (lecture seule) : un entier (index basé sur 1) qui indique la dernière ligne actuellement visible dans *mon_txt*. Considérez le champ de texte comme une « fenêtre » sur un bloc de texte. La propriété TextField.scroll est l'index basé sur 1 de la première ligne visible dans la fenêtre.

Tout le texte entre les lignes TextField.scroll et TextField.bottomScroll est actuellement visible dans le champ de texte.

TextField.condenseWhite

Disponibilité

Flash Player 6.

Usage

mon_txt.condenseWhite

Description

Propriété : une valeur booléenne indiquant si les espaces blancs ajoutés (espaces, sauts de ligne, etc.) à un champ de texte HTML doivent être supprimés lors du rendu de ce dernier dans un navigateur. La valeur par défaut est false.

Si vous définissez cette valeur sur true, vous devez utiliser des commandes HTML standard (par exemple,
 et>P<) pour insérer des sauts de ligne dans le champ de texte.

Si *mon_txt*.html est définie sur false, cette propriété est ignorée.

Consultez également

TextField.html

TextField.embedFonts

Disponibilité

Flash Player 6.

Usage

mon_txt.embedFonts

Description

Propriété : une valeur booléenne qui, lorsque true, rend le champ de texte avec des polices vectorielles intégrées. Si false, le champ de texte est rendu avec les polices de périphériques.

TextField.getDepth()

Disponibilité

Flash Player 6.

Usage

mon_txt.getDepth()

Paramètres

Aucun.

Renvoie

Un entier.

Description

Méthode : renvoie la profondeur d'un champ de texte.

TextField.getFontList

Disponibilité

Flash Player 6.

Usage

TextField.getFontList()

Paramètres

Aucun.

Renvoie

Un tableau.

Description

Méthode : une méthode statique de la Classe TextField globale. Lorsque vous appelez cette méthode, vous ne spécifiez pas un champ de texte précis (tel que mon_txt). Cette méthode renvoie les noms des polices sur le système hôte du lecteur sous forme de tableau. (Elle ne renvoie pas les noms de toutes les polices dans les fichiers SWF actuellement chargés.) Les noms sont de type chaîne.

Exemple

Le code suivant affiche une liste des polices renvoyée par getFontList().

```
polices_array = TextField.getFontList();
for( i in polices_array){
     trace(polices_array[i]);
}
```

TextField.getNewTextFormat()

Disponibilité

Flash Player 6.

Usage

mon_txt.getNewTextFormat()

Paramètres

Aucun.

Renvoie

Un objet TextFormat.

Description

Méthode : renvoie un objet TextFormat contenant une copie de l'objet de format de texte du champ de texte. L'objet de format de texte est le format que reçoit le texte nouvellement inséré, tel que le texte inséré par la méthode `replaceSel()` ou le texte entré par un utilisateur. Lorsque `getNewTextFormat()` est invoqué, l'objet TextFormat est retourné avec toutes ses propriétés définies. Aucune propriété n'est `null`.

TextField.getTextFormat()

Disponibilité

Flash Player 6.

Usage

```
mon_txt.getTextFormat()
mon_txt.getTextFormat(index)
mon_txt.getTextFormat(indexDébut, indexFin)
```

Paramètres

indexUn entier spécifiant un caractère d'une chaîne.

indexDébut, indexFin Entiers indiquant les emplacements de début et de fin d'une plage de texte dans *mon_txt*.

Renvoie

Un objet.

Description

Méthode : Usage 1 : renvoie un objet TextFormat contenant des informations de format sur l'ensemble du texte contenu dans un champ de texte. Seules les propriétés qui sont communes à tout le texte du champ de texte sont définies dans l'objet TextFormat résultant. Toute propriété qui est *mixte*, ce qui signifie qu'elle a différentes valeurs à différents points du texte, a sa valeur définie sur `null`.

Usage 2 : Renvoie un objet TextFormat contenant une copie du format de texte du champ de texte à *index*.

Usage 3 : Renvoie un objet TextFormat contenant des informations de format pour la plage de texte allant de *indexDébut* à *indexFin*.

Consultez également

`TextField.getNewTextFormat()`, `TextField.setNewTextFormat`, `TextField.setTextFormat`

TextField._height

Disponibilité

Flash Player 6.

Usage

mon_txt._height

Description

Propriété : la hauteur du champ de texte, en pixels.

Exemple

L'exemple de code suivant définit la hauteur et la largeur d'un champ de texte.

```
mon_txt._width = 200;
mon_txt._height = 200;
```

TextField._highquality

Disponibilité

Flash Player 6.

Usage

mon_txt._highquality

Description

Propriété (globale) : spécifie le niveau d'anti-aliasing appliqué au fichier SWF en cours. Spécifiez 2 (qualité maximum) pour appliquer une qualité élevée avec le lissage bitmap toujours actif. Spécifiez 1 (qualité élevée) pour appliquer l'anti-aliasing ; cela permettra de lisser les bitmaps si le fichier SWF ne contient pas d'animation. Spécifiez 0 (faible qualité) pour empêcher l'anti-aliasing.

Consultez également

_quality

TextField.hscroll

Disponibilité

Flash Player 6.

Usage

```
mon_txt.hscroll
```

Renvoie

Un entier.

Description

Propriété : indique la position de défilement horizontal actuelle. Si la propriété `hscroll` a pour valeur 0, le texte ne défile pas horizontalement.

Pour plus d'informations sur le texte défilant, consultez *Création de texte défilant*, page 161.

Exemple

L'exemple suivant fait défiler le texte horizontalement.

```
on (release) {
  mon_txt.hscroll += 1;
}
```

Consultez également

`TextField.maxhscroll`, `TextField.scroll`

TextField.html

Disponibilité

Flash Player 6.

Usage

```
mon_txt.html
```

Description

Propriété : une valeur indiquant si le champ de texte contient une représentation HTML. Si la propriété `html` est `true`, le champ de texte est un champ de texte html. Si `html` est `false`, le champ de texte n'est pas un champ de texte html.

Consultez également

`TextField.htmlText`

TextField.htmlText

Disponibilité

Flash Player 6.

Usage

mon_txt.htmlText

Description

Propriété : si le champ de texte est un champ de texte HTML, cette propriété contient la représentation HTML du contenu du champ de texte. Si le champ de texte n'est pas un champ de texte HTML, elle se comporte de façon identique à la propriété `text`. Vous pouvez indiquer qu'un champ de texte est un champ de texte HTML dans l'inspecteur des propriétés, ou en définissant la propriété `html` du champ de texte sur `true`.

Exemple

Dans l'exemple suivant, le texte du champ de texte `texte2` est rendu en gras.

```
texte2.html = true;
texte2.htmlText = "<b> ceci est du texte en gras </b>";
```

Consultez également

TextField.html

TextField.length

Disponibilité

Flash Player 6.

Usage

mon_txt.length

Renvoie

Un nombre.

Description

Propriété (lecture seule) : indique le nombre de caractères d'un champ de texte. Cette propriété renvoie la même valeur que `text.length`, mais elle est plus rapide. Un caractère tel que « tab » (« \t ») compte comme un seul caractère.

TextField.maxChars

Disponibilité

Flash Player 6.

Usage

mon_txt.maxChars

Description

Propriété : indique la quantité maximum de caractères qu'un champ de texte peut contenir. Un script peut insérer plus de texte que `maxChars` ne le permet : la propriété `maxChars` n'indique que la quantité de texte qu'un utilisateur peut entrer. Si la valeur de cette propriété est `null`, il n'y a pas de limite sur la quantité de texte qu'un utilisateur peut entrer.

TextField.maxhscroll

Disponibilité

Flash Player 6.

Usage

mon_txt.maxhscroll

Description

Propriété (lecture seule) : indique la valeur maximum de `TextField.hscroll`.

TextField.maxscroll

Disponibilité

Flash Player 6.

Usage

monChampDeTexte.maxscroll

Description

Propriété (lecture seule) : indique la valeur maximale de `TextField.scroll`.

Pour plus d'informations sur le texte défilant, consultez *Création de texte défilant*, page 161.

TextField.menu

Disponibilité

Flash Player 7.

Utilisation

```
mon_txt.menu = contextMenu
```

Paramètres

contextMenu Un objet ContextMenu.

Description

Propriété : associe l'objet *contextMenu* au champ de texte *mon_txt*. La classe ContextMenu vous permet de modifier le menu contextuel qui apparaît quand l'utilisateur clique avec le bouton droit de la souris (Windows) ou enfonce la touche Contrôle (Macintosh) dans Flash Player.

Cette propriété fonctionne uniquement avec les champs de texte sélectionnables (modifiables) ; elle n'a aucun effet sur les champs de texte non sélectionnables.

Exemple

L'exemple suivant attribue l'objet ContextMenu `menu_cm` au champ de texte `news_txt`. L'objet ContextMenu contient une option de menu personnalisée intitulée « Imprimer » avec un gestionnaire de rappel associé nommé `doPrint()`, permettant d'effectuer les opérations d'impression (masqué) :

```
var menu_cm = new ContextMenu();
menu_cm.customItems.push(nouveau ContextMenuItem("Imprimer...", doPrint));
function doPrint(menu, obj) {
  // Code "Imprimer" ici
}
news_txt.menu = menu_cm;
```

Voir aussi

`Button.menu`, `Classe ContextMenu`, `Classe ContextMenuItem`, `MovieClip.menu`

TextField.mouseWheelEnabled

Disponibilité

Flash Player 7.

Usage

mon_txt.mouseWheelEnabled

Description

Propriété : une valeur booléenne indiquant si Flash Player doit automatiquement faire défiler les champs de texte de plusieurs lignes lorsque le pointeur de la souris se trouve sur un champ de texte et que l'utilisateur actionne la molette de la souris. Par défaut, cette valeur est `true`. Cette propriété est utile si vous voulez éviter de faire défiler les champs de texte avec la molette ou mettre en œuvre votre propre défilement des champs de texte.

Consultez également

Mouse.onMouseWheel

TextField.multiline

Disponibilité

Flash Player 6.

Usage

mon_txt.multiline

Description

Propriété : indique si le champ de texte est un champ de 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 à ligne unique.

TextField._name

Disponibilité

Flash Player 6.

Usage

mon_txt._name

Description

Propriété : le nom de l'occurrence du champ de texte spécifié par *mon_txt*.

TextField.onChanged

Disponibilité

Flash Player 6.

Usage

```
mon_txt.onChanged = function(){
  // vos instructions
}
```

Paramètres

Aucun.

Renvoie

Le nom de l'occurrence du champ de texte.

Description

Gestionnaire d'événement : invoqué lorsque le contenu d'un champ de texte change. Par défaut, il est undefined ; vous pouvez le définir dans un script.

Une référence à l'occurrence du champ de texte est affectée en tant que paramètre au gestionnaire `onChanged`. Vous pouvez capturer ces données en plaçant un paramètre dans la méthode du gestionnaire d'événement. Par exemple, le code suivant utilise `txt` en tant que paramètre affecté au gestionnaire d'événement `onChanged`. Puis, le paramètre est utilisé dans une instruction `trace()` afin d'envoyer le nom de l'occurrence du champ de texte vers le panneau de sortie.

```
monChampDeTexte.onChanged = function (txt) {
  trace (txt._nom + " changé");
};
```

TextField.onKillFocus

Disponibilité

Flash Player 6.

Usage

```
mon_txt.onKillFocus = function(nouveauFocus){
  // vos instructions
}
```

Paramètres

`nouveauFocus` L'objet recevant le focus.

Renvoie

Rien.

Description

Gestionnaire d'événement : invoqué lorsqu'un champ de texte perd le focus clavier. La méthode `onKillFocus` reçoit un paramètre, *`nouveauFocus`*, qui est un objet représentant le nouvel objet recevant le focus. Si aucun objet ne reçoit le focus, *`nouveauFocus`* contient la valeur `null`.

TextField.onScroller

Disponibilité

Flash Player 6.

Usage

```
mon_txt.onScroller = function(occurrenceDeChampDeTexte){
  // vos instructions
}
```

Paramètres

occurrenceDeChampDeTexte Une référence à l'objet TextField dont la position de défilement a été modifiée.

Renvoie

Rien.

Description

Gestionnaire d'événement : invoqué lorsque l'une des propriétés de défilement du champ de texte change.

Une référence à l'occurrence du champ de texte est affectée en tant que paramètre au gestionnaire `onScroller`. Vous pouvez capturer ces données en plaçant un paramètre dans la méthode du gestionnaire d'événement. Par exemple, le code suivant utilise `txt` en tant que paramètre affecté au gestionnaire d'événement `onScroller`. Puis, le paramètre est utilisé dans une instruction `trace()` afin d'envoyer le nom de l'occurrence du champ de texte vers le panneau de sortie.

```
monChampDeTexte.onScroller = function (txt) {
  trace (txt._nom + " défilé");
};
```

Consultez également

`TextField.hscroll`, `TextField.maxhscroll`, `TextField.maxscroll`, `TextField.scroll`

TextField.onSetFocus

Disponibilité

Flash Player 6.

Usage

```
mon_txt.onSetFocus = function(ancienFocus){
  // vos instructions
}
```

Paramètres

ancienFocus L'objet devant perdre le focus.

Renvoie

Rien.

Description

Gestionnaire d'événement : invoqué lorsqu'un champ de texte reçoit le focus clavier. Le paramètre *ancienFocus* est l'objet perdant le focus. Par exemple, si l'utilisateur appuie sur la touche Tab pour faire passer le focus de saisie d'un bouton à un champ de texte, *ancienFocus* contient l'occurrence de champ de texte.

Si aucun objet n'a précédemment reçu le focus, *ancienFocus* contient une valeur `null`.

TextField._parent

Disponibilité

Flash Player 6.

Usage

```
mon_txt._parent.propriété
_parent.propriété
```

Description

Propriété : une référence au clip ou à l'objet contenant le champ de texte ou l'objet actuel. L'objet courant est celui contenant le code ActionScript faisant référence à `_parent`.

`_parent` permet de spécifier un chemin relatif aux clips ou objets se trouvant au-dessus du champ de texte actuel. Vous pouvez utiliser `_parent` pour monter de plusieurs niveaux dans la liste d'affichage, comme dans l'exemple suivant :

```
_parent._parent._alpha = 20;
```

Consultez également

`Button._parent`, `MovieClip._parent`, `_root`, `targetPath`

TextField.password

Disponibilité

Flash Player 6.

Usage

mon_txt.password

Description

Propriété : si la valeur password est true, le champ de texte est un champ de texte de mot de passe dont le contenu est masqué. Si false, le champ de texte n'est pas un champ de mode passe.

TextField._quality

Disponibilité

Flash Player 6.

Usage

mon_txt._quality

Description

Propriété (globale) : définit ou récupère la qualité de rendu utilisée pour un fichier SWF. Les polices de périphérique sont toujours aliasées et ne sont donc pas affectées par la propriété _quality.

Remarque : Même si vous pouvez spécifier cette propriété pour un objet TextField, il s'agit en fait d'une propriété globale et vous pouvez en spécifier la valeur simplement en tant que _quality. Pour plus d'informations, consultez _quality.

TextField.removeListener

Disponibilité

Flash Player 6.

Usage

mon_txt.removeListener(*écouteur*)

Paramètres

écouteur L'objet qui ne recevra plus de notifications de TextField.onChanged ou de TextField.onScroller.

Renvoie

Si *écouteur* a été correctement supprimé, la méthode renvoie une valeur true. Si *écouteur* n'a pas été correctement supprimé, par exemple, si *écouteur* n'apparaissait pas dans la liste des écouteurs de l'objet TextField, la méthode renvoie une valeur false.

Description

Méthode : retire un objet d'écoute précédemment enregistré pour une occurrence de champ de texte avec TextField.addListener().

TextField.removeTextField()

Disponibilité

Flash Player 6.

Usage

mon_txt.removeTextField()

Description

Méthode : supprime le champ de texte spécifié par *mon_txt*. Cette opération peut uniquement être réalisée sur un champ de texte créé avec MovieClip.createTextField(). Lorsque vous appelez cette méthode, le champ de texte est supprimé. Cette méthode est similaire à MovieClip.removeMovieClip().

TextField.replaceSel()

Disponibilité

Flash Player 6.

Usage

mon_txt.replaceSel(*texte*)

Paramètres

texte Une chaîne.

Renvoie

Rien.

Description

Méthode : remplace la sélection actuelle par le contenu du paramètre *texte*. Le texte est inséré à la place de la sélection actuelle, en utilisant le format de caractère par défaut et le format de paragraphe par défaut actuellement en vigueur. Le texte n'est pas considéré comme HTML, même si le champ de texte est un champ de texte HTML.

Vous pouvez utiliser la méthode replaceSel() pour insérer et supprimer du texte sans perturber le format des caractères et des paragraphes du reste du texte.

Vous devez utiliser Selection.setFocus pour cibler le champ avant d'émettre cette commande.

Consultez également

Selection.setFocus

TextField.replaceText()

Disponibilité

Flash Player 7.

Usage

mon_txt.replaceText(*indexDébut*, *indexFin*, *texte*)

Description

Méthode : remplace une série de caractères, spécifié par les paramètres *indexDébut* et *indexFin*, dans le champ de texte spécifié avec le contenu du paramètre *texte*.

TextField.restrict

Disponibilité

Flash Player 6.

Usage

mon_txt.restrict

Description

Propriété : indique le jeu de caractères qu'un utilisateur peut rentrer dans le champ de texte. Si la valeur de la propriété `restrict` est `null`, vous pouvez entrer n'importe quel caractère. Si la valeur de la propriété `restrict` est une chaîne vide, vous ne pouvez entrer aucun caractère. Si la valeur de la propriété `restrict` est une chaîne de caractères, vous ne pouvez entrer que les caractères de la chaîne dans le champ de texte. La chaîne est lue de gauche à droite. Une plage peut être spécifiée en utilisant un tiret (-). Ceci limite seulement l'interaction avec l'utilisateur, un script pouvant mettre n'importe quel texte dans le champ de texte. Cette propriété ne se synchronise pas avec les cases à cocher de polices vectorielles intégrées de l'inspecteur des propriétés.

Si la chaîne commence par un caret (^), tous les caractères sont initialement acceptés et les caractères suivants de la chaîne sont exclus du jeu de caractères acceptés. Si la chaîne ne commence pas par un caret (^), aucun caractère n'est initialement accepté et les caractères suivants de la chaîne sont inclus dans le jeu de caractères acceptés.

Exemple

L'exemple suivant permet seulement que des caractères majuscules, des espaces et des nombres soient entrés dans un champ de texte :

```
mon_txt.restrict = "A-Z 0-9";
```

L'exemple suivant comprend tous les caractères, mais exclut les lettres minuscules :

```
mon_txt.restrict = "^a-z";
```

Vous pouvez utiliser une barre oblique inverse pour entrer un ^ ou - textuellement. Les séquences de barre oblique inverse acceptées sont \-, \^ ou \\. La barre oblique inverse doit être effectivement un caractère de la chaîne ; aussi, pour le spécifier dans ActionScript, une double barre oblique inverse doit être utilisée. Par exemple, le code suivant inclut uniquement le tiret (-) et le caret (^) :

```
mon_txt.restrict = "\\-\\^";
```

Le caret (^) peut être utilisé n'importe où dans la chaîne pour alterner entre l'inclusion de caractères et l'exclusion de caractères. Le code suivant inclut seulement les lettres majuscules, mais exclut la lettre majuscule Q :

```
mon_txt.restrict = "A-Z^Q";
```

Vous pouvez utiliser la séquence d'échappement `\u` pour construire des chaînes `restrict`. Le code suivant inclut seulement les caractères compris entre le code ASCII 32 (espace) et le code ASCII 126 (tilde).

```
mon_txt.restrict = "\u0020-\u007E";
```

TextField._rotation

Disponibilité

Flash Player 6.

Usage

mon_txt.`_rotation`

Description

Propriété : la rotation du champ de texte, en degrés, à partir de son orientation d'origine. Les valeurs de 0 à 180 représentent une rotation dans le sens horaire ; les valeurs de 0 à -180 représentent une rotation dans le sens antihoraire. Les valeurs en dehors de cette plage sont ajoutées à ou soustraites de 360 pour obtenir une valeur comprise dans la plage. Par exemple, l'instruction `mon_txt._rotation = 450` est la même que `mon_txt._rotation = 90`.

Consultez également

`Button._rotation`, `MovieClip._rotation`

TextField.scroll

Disponibilité

Flash Player 6.

Usage

mon_txt.scroll

Description

Propriété : définit la position verticale du texte dans un champ de texte. La propriété `scroll` est utile pour diriger les utilisateurs vers un paragraphe spécifique dans un long passage, ou pour créer des champs de texte défilants. Cette propriété peut être récupérée et modifiée.

Pour plus d'informations sur le texte défilant, consultez *Création de texte défilant*, page 161.

Exemple

Le code suivant est associé à un bouton Haut qui fait défiler le champ de texte `mon_txt`.

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

Consultez également

`TextField.hscroll`, `TextField.maxscroll`

TextField.selectable

Disponibilité

Flash Player 6.

Usage

mon_txt.selectable

Description

Propriété : une valeur booléenne indiquant si le champ de texte est sélectionnable (modifiable). La valeur `true` indique que le texte est sélectionnable.

TextField.setNewTextFormat

Disponibilité

Flash Player 6.

Usage

mon_txt.setNewTextFormat(*formatDeTexte*)

Paramètres

formatDeTexte Une occurrence de l'objet TextFormat.

Renvoie

Rien.

Description

Méthode : définit un objet TextFormat pour le texte nouvellement inséré, tel que le texte inséré par la méthode `replaceSel()` ou le texte entré par un utilisateur dans un champ de texte. Chaque champ de texte a un nouveau format de texte. Lorsque du texte est inséré, le nouveau format est appliqué au nouveau texte.

Le format de texte est défini dans un nouvel objet TextFormat. Contient des informations de format de caractères et de paragraphes. Les informations de format des caractères décrivent l'apparence des différents caractères, telles que le nom de la police, la taille de la police, la couleur du texte et l'URL associée. Les informations de format des paragraphes décrivent l'apparence des paragraphes, telles que la marge gauche, la marge droite, l'indentation de la première ligne et l'alignement à gauche, à droite, au centre ou justifié.

Consultez également

`TextField.getNewTextFormat()`, `TextField.getTextFormat()`, `TextField.setTextFormat`

TextField.setTextFormat

Disponibilité

Flash Player 6.

Usage

```
mon_txt.setTextFormat (formatDeTexte)
mon_txt.setTextFormat (index, formatDeTexte)
mon_txt.setTextFormat (indexDébut, indexFin, formatDeTexte)
```

Paramètres

formatDeTexte Un objet TextFormat contenant des informations sur le format des caractères et des paragraphes.

index Un entier spécifiant un caractère dans `mon_txt`.

indexDébut Un entier.

indexFin Un entier indiquant le premier caractère suivant la plage de texte souhaitée.

Renvoie

Rien.

Description

Méthode : définit un objet TextFormat pour une plage de texte spécifiée dans un champ de texte. Vous pouvez affecter un format de texte à chaque caractère d'un champ de texte. Le format du premier caractère d'un paragraphe est examiné pour effectuer le formatage de paragraphe sur tout le paragraphe. La méthode `setTextFormat()` modifie le format de texte appliqué à chaque caractère, à des groupes de caractères ou à l'ensemble du corps de texte d'un champ de texte.

Le format de texte est défini dans un nouvel objet TextFormat. Contient des informations de format de caractères et de paragraphes. Les informations de format des caractères décrivent l'apparence des différents caractères, telles que le nom de la police, la taille de la police, la couleur du texte et l'URL associée. Les informations de formatage des paragraphes décrivent l'aspect des paragraphes, par exemple la marge gauche, la marge droite, l'indentation de la première ligne et l'alignement à gauche, à droite, au centre ou justifié.

Usage 1 : Applique les propriétés de *formatDeTexte* à tout le texte du champ de texte.

Usage 2 : Applique les propriétés de *formatDeTexte* au caractère à la position *index*.

Usage 3 : Applique les propriétés du paramètre *formatDeTexte* à la plage de texte allant du paramètre *indexDébut* au paramètre *indexFin*.

Notez que tout texte inséré manuellement par l'utilisateur, ou remplacé au moyen de `TextField.replaceSel()`, ne prend pas le format spécifié dans un appel de `setTextFormat()`. Pour définir le format par défaut d'un objet champ de texte, utilisez `TextField.setNewTextFormat`.

Exemple

Cet exemple crée un objet TextFormat appelé `monFormatDeTexte` et définit sa propriété `bold` sur `true`. Il appelle ensuite la méthode `setTextFormat()` et applique le nouveau format de texte au champ de texte `mon_txt`.

```
monFormatDeTexte = new TextFormat();
monFormatDeTexte.bold = true;
mon_txt.setTextFormat(monFormatDeTexte);
```

Consultez également

`TextField.setNewTextFormat`, `Classe TextFormat`

TextField._soundbuftime

Disponibilité

Flash Player 6.

Usage

`mon_txt``._soundbuftime`

Description

Propriété (globale) : un entier spécifiant le nombre de secondes de mise en tampon d'un son avant sa lecture en flux continu.

Classe TextField.StyleSheet

Disponibilité

Flash Player 7.

Description

La classe TextField.StyleSheet permet de créer un objet feuille de style contenant des règles de format de texte, telles que la taille, la couleur et autres styles de format. Vous pouvez ensuite appliquer les styles définis par une feuille à un objet TextField contenant du texte au format HTML ou XML. Le texte contenu dans l'objet TextField est ensuite automatiquement formaté selon le style des balises défini dans l'objet feuille de style. Les styles de texte permettent de définir de nouvelles balises de formatage, de redéfinir les balises HTML intégrées ou de créer des classes de style pouvant être appliquées à certaines balises HTML.

Pour appliquer des styles à un objet TextField, affectez l'objet feuille de style à la propriété `styleSheet` d'un objet TextField.

Pour plus d'informations, consultez *Formatage de texte avec les feuilles de style en cascade*, page 145.

Méthodes de la classe TextField.StyleSheet

Méthode	Description
`TextField.StyleSheet.getStyle()`	Renvoie une copie de l'objet feuille de style associé au nom de style spécifié.
`TextField.StyleSheet.getStyleNames()`	Renvoie un tableau contenant les noms de tous les styles enregistrés dans l'objet feuille de style.
`TextField.StyleSheet.load()`	Commence à charger un fichier CSS dans l'objet feuille de style.
`TextField.StyleSheet.parseCSS()`	Analyse une chaîne de texte CSS et crée le style spécifié.
`TextField.StyleSheet.setStyle()`	Ajoute un nouveau style à l'objet feuille de style.

Gestionnaires d'événement de la classe TextField.StyleSheet

Méthode	Description
`TextField.StyleSheet.onLoad`	Gestionnaire de rappel invoqué lorsqu'une opération `TextField.StyleSheet.load()` est terminée.

Constructeur de la classe TextField.StyleSheet

Disponibilité

Flash Player 7.

Usage

```
new TextField.StyleSheet()
```

Renvoie

Rien.

Description

Constructeur : crée un objet TextField.StyleSheet.

TextField.StyleSheet.getStyle()

Disponibilité

Flash Player 7.

Usage

feuilleDeStyle.getStyle(*nomDeStyle*)

Paramètres

nomDeStyle Une chaîne qui spécifie le nom du style à récupérer.

Renvoie

Un objet.

Description

Méthode : renvoie une copie de l'objet style associé au style nommé *nomDeStyle*. S'il n'existe pas d'objet style associé au *nomDeStyle*, *null* est renvoyé.

Exemple

Supposez qu'un objet feuille de style nommé `stylesDeTexte` charge un fichier de feuille de style externe nommé styles.css contenant un seul style nommé `heading`, définissant les propriétés `font-family`, `font-size` et `font-weight`, comme indiqué ci-dessous.

```
// Dans styles.css
heading {
  font-family: Arial;
  font-size: 24px;
  font-weight: bold;
}
```

Le code suivant charge les styles à partir du fichier CSS et affiche ensuite chaque nom de propriété et sa valeur dans le panneau de sortie.

```
var feuilleDeStyle = new TextField.styleSheet();
feuilleDeStyle.load("styles.css");
var styleDeSection = styleSheet.getStyle("heading");
for(propriété dans sectionStyle) {
  var nomProp = property;
  var ValeurProp = sectionStyle[propriété];
  trace(nomProp + " : " + ValeurProp);
}
```

Le panneau de sortie affiche alors les informations suivantes :

```
fontfamily : Arial
fontsize : 24px
fontweight : gras
```

Consultez également

TextField.StyleSheet.setStyle()

TextField.StyleSheet.getStyleNames()

Disponibilité

Flash Player 7.

Usage

feuilleDeStyle.getStyleNames()

Paramètres

Aucun.

Renvoie

Un tableau.

Description

Méthode : renvoie un tableau contenant les noms (sous forme de chaînes) de tous les styles enregistrés dans la feuille de style.

Exemple

Cet exemple crée un objet feuille de style nommé `feuilleDeStyle` contenant deux styles, `heading` et `bodyText`. Il invoque ensuite la méthode `getStyleNames()` de l'objet feuille de style, affecte le résultat au tableau `noms_array` et affiche son contenu dans le panneau de sortie.

```
var feuilleDeStyle= new TextField.StyleSheet();
feuilleDeStyle.setStyle("heading", {
  fontsize: '24px'
});
styleSheet.setStyle("bodyText", {
  fontsize: '12px'
});
var noms_tableau = feuilleDeStyle.getStyleNames();
trace(names.join("\n"));
```

Le résultat suivant apparaît dans le panneau de sortie :

```
bodyText
heading
```

Consultez également

TextField.StyleSheet.getStyle()

TextField.StyleSheet.load()

Disponibilité

Flash Player 7.

Usage

feuilleDeStyle.load(*url*)

Paramètres

url L'URL d'un fichier CSS à charger. L'URL doit se trouver dans le même domaine que l'URL sur laquelle réside le fichier SWF.

Renvoie

Rien.

Description

Méthode : commence à charger le fichier CSS dans *feuilleDeStyle*. L'opération de chargement est asynchrone ; utilisez le gestionnaire de rappel TextField.StyleSheet.onLoad afin de déterminer à quel moment le chargement du fichier se termine.

Le fichier CSS doit résider exactement dans le même domaine que le fichier SWF qui le charge. Pour plus d'informations sur les restrictions de chargement de données dans tous les domaines, consultez *Fonctions de sécurité de Flash Player*, page 199.

Exemple

L'exemple suivant charge le fichier CSS nommé styles.css (masqué) dans l'objet la feuille de style `objStyle`. Lorsque le fichier est correctement chargé, l'objet feuille de style s'applique à un objet TextField nommé `news_txt`.

```
var objStyle = new TextField.StyleSheet();
objStyle.load("styles.css");
objStyle.onLoad = function (success) {
  if (success) {
    news_txt.styleSheet = objStyle;
  }
}
```

Consultez également

TextField.StyleSheet.onLoad

TextField.StyleSheet.onLoad

Disponibilité

Flash Player 7.

Usage

```
feuilleDeStyle.onLoad = function (success) {}
```

Paramètres

success Une valeur booléenne indiquant si le fichier CSS a été correctement chargé.

Renvoie

Rien.

Description

Gestionnaire de rappel : invoqué lorsqu'une opération `TextField.StyleSheet.load()` est achevée. Si la feuille de style est correctement chargée, le paramètre *success* est `true`. Si le document n'a pas été reçu, ou si une erreur est survenue lors de la réception de la réponse du serveur, le paramètre *success* est `false`.

Exemple

L'exemple suivant charge le fichier CSS nommé styles.css (masqué) dans l'objet la feuille de style `objStyle`. Lorsque le fichier est correctement chargé, l'objet feuille de style s'applique à un objet TextField nommé `news_txt`.

```
var objStyle = new TextField.StyleSheet();
objStyle.load("styles.css");
objStyle.onLoad = function (success) {
  if (success) {
    news_txt.styleSheet = objStyle;
  }
}
```

Consultez également

`TextField.StyleSheet.load()`

TextField.StyleSheet.parseCSS()

Disponibilité

Flash Player 7.

Usage

```
feuilleDeStyle.parseCSS(TexteCss)
```

Paramètres

texteCss Le texte CSS à analyser (une chaîne).

Renvoie

Une valeur booléenne indiquant si le texte a été analysé correctement (`true`) ou non (`false`).

Description

Méthode : analyse le CSS dans *texteCss* et charge la feuille de style. Si un style dans *texteCss* existe déjà dans *feuilleDeStyle*, les propriétés de *feuilleDeStyle* restent les mêmes et seules celles de *texteCss* sont ajoutées/modifiées dans *feuilleDeStyle*.

Pour étendre la capacité d'analyse CSS native, vous pouvez annuler cette méthode en créant une sous-classe de la classe TextField.StyleSheet. Pour plus d'informations, consultez *Création de sous-classes*, page 171.

TextField.StyleSheet.setStyle()

Disponibilité

Flash Player 7.

Usage

feuilleDeStyle.setStyle(*nom*, *style*)

Paramètres

nom Une chaîne spécifiant le nom du style à ajouter dans la feuille de style.

style Un objet décrivant le style, ou `null`.

Renvoie

Rien.

Description

Méthode : ajoute un nouveau style avec le nom spécifié à l'objet feuille de style. Si le style nommé n'existe pas déjà dans la feuille de style, il est ajouté. S'il existe déjà dans la feuille de style, il est remplacé. Si le paramètre *style* est `null`, le style nommé est supprimé.

Flash Player crée une copie de l'objet style que vous transmettez à cette méthode.

Exemple

Le code suivant ajoute un style nommé `emphasized` à la feuille de style `maFeuilleDeStyle`. Le style inclut deux propriétés de style : `color` et `fontWeight`. L'objet style est défini à l'aide de l'opérateur `{}`.

```
maFeuilleDeStyle.setStyle("emphasized", {color:'#000000',fontWeight:'bold'});
```

Vous pouvez également créer un objet style à l'aide d'une occurrence de la classe Object, puis transmettre cet objet en tant que paramètre *style*, comme indiqué dans l'exemple suivant.

```
var objStyle = new Object();
objStyle.color = '#000000';
objStyle.fontWeight = 'bold';
maFeuilleDeStyle.setStyle("emphasized", styleObj);
delete styleObj;
```

Remarque : La dernière ligne de code (`delete styleObj`) supprime l'objet style d'origine transmis à `setStyle()`. Cette étape est facultative, mais permet de libérer de la mémoire, car Flash Player crée une copie de l'objet style transmis à `setStyle()`.

Consultez également

`{} (initialisateur d'objet)`

TextField.styleSheet

Disponibilité

Flash Player 7.

Usage

```
mon_txt.styleSheet = TextField StyleSheet
```

Description

Propriété : lie une feuille de style à un champ de texte spécifié par *mon_txt*. Pour plus d'informations sur la création des feuilles de style, consultez l'entrée Classe TextField.StyleSheet et *Formatage de texte avec les feuilles de style en cascade*, page 145.

TextField.tabEnabled

Disponibilité

Flash Player 6.

Usage

```
mon_txt.tabEnabled
```

Description

Propriété : spécifie si *mon_txt* est inclus dans l'ordre de tabulation automatique. Valeur `undefined` par défaut.

Si la propriété `tabEnabled` est `undefined` ou `true`, l'objet est inclus dans l'ordre de tabulation automatique. Si la propriété `tabIndex` est également définie avec une valeur, l'objet est également inclus dans l'ordre de tabulation automatique. Si `tabEnabled` est `false`, l'objet n'est pas inclus dans l'ordre de tabulation automatique ou personnalisé, même si la propriété `tabIndex` est définie.

Consultez également

`Button.tabEnabled`, `mon_mc.tabEnabled`

TextField.tabIndex

Disponibilité

Flash Player 6.

Usage

mon_txt.tabIndex

Paramètres

Aucun.

Renvoie

Rien.

Description

Propriété : permet de personnaliser l'ordre de tabulation des objets d'un fichier SWF. Vous pouvez définir la propriété `tabIndex`, qui est `undefined` par défaut, pour une occurrence de bouton, clip ou champ de texte.

Si l'un des objets affichés dans le fichier SWF contient une propriété `tabIndex`, l'ordre de tabulation automatique est désactivé, et l'ordre de tabulation est alors calculé en fonction des propriétés `tabIndex` des objets du fichier SWF. L'ordre de tabulation personnalisé n'inclut que les objets possédant des propriétés `tabIndex`.

La propriété `tabIndex` doit être un entier positif. Les objets sont placés dans l'ordre correspondant à leurs propriétés `tabIndex`, dans un ordre croissant. Un objet dont la valeur de propriété `tabIndex` est `1` précède un objet dont la valeur de propriété `tabIndex` est 2. Si deux objets ont la même valeur `tabIndex`, celui qui précède l'autre dans l'ordre de tabulation est `undefined`.

L'ordre de tabulation personnalisé défini par la propriété `tabIndex` est *flat*. Cela signifie que la relation hiérarchique des objets du fichier SWF n'a pas importance. Tous les objets du fichier SWF possédant des 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 précède l'autre dans l'ordre de tabulation est `undefined`. Vous ne devriez pas utiliser la même valeur `tabIndex` pour plusieurs objets.

Consultez également

`Button.tabIndex`, `MovieClip.tabIndex`

TextField._target

Disponibilité

Flash Player 6.

Usage

mon_txt._target

Description

Propriété (lecture seule) : le chemin cible d'une occurrence de champ de texte spécifiée par *mon_txt*.

TextField.text

Disponibilité

Flash Player 6.

Usage

mon_txt.text

Description

Propriété : indique le texte d'un champ de texte. Les lignes sont séparées par le caractère de retour chariot (« \r », ASCII 13). Cette propriété contient le texte normal, non formaté, sans balises HTML, même si le champ de texte est en HTML.

Consultez également

TextField.htmlText

TextField.textColor

Disponibilité

Flash Player 6.

Usage

mon_txt.textColor

Description

Propriété : indique la couleur du texte d'un champ de texte.

TextField.textHeight

Disponibilité

Flash Player 6.

Usage

mon_txt.textHeight

Description

Propriété : indique la hauteur du texte.

TextField.textWidth

Disponibilité

Flash Player 6.

Usage

mon_txt.textWidth

Description

Propriété : indique la largeur du texte.

TextField.type

Disponibilité

Flash Player 6.

Usage

mon_txt.type

Description

Propriété : spécifie le type du champ de texte. Il y a deux valeurs : `"dynamic"`, qui spécifie un champ de texte dynamique ne pouvant pas être modifié par l'utilisateur et `"input"`, qui spécifie un champ de texte de saisie.

Exemple

```
mon_txt.type = "dynamic";
```

TextField._url

Disponibilité

Flash Player 6.

Usage

mon_txt._url

Description

Propriété (lecture seule) : récupère l'URL du fichier SWF créateur de ce champ de texte.

TextField.variable

Disponibilité

Flash Player 6.

Usage

mon_txt.variable

Description

Propriété : le nom de la variable à laquelle le champ de texte est associé. Le type de cette propriété est String.

TextField._visible

Disponibilité

Flash Player 6.

Usage

mon_txt._visible

Description

Propriété : une valeur booléenne indiquant si le champ de texte *mon_txt* est visible. Les champs de texte qui ne sont pas visibles (propriété _visible définie sur false) sont désactivés.

Consultez également

Button._visible, MovieClip._visible

TextField._width

Disponibilité

Flash Player 6.

Usage

mon_txt._width

Description

Propriété : la largeur du champ de texte, en pixels.

Exemple

L'exemple suivant définit les propriétés de hauteur et de largeur d'un champ de texte :

```
mon_txt._width=200;
mon_txt._height=200;
```

Consultez également

MovieClip._height

TextField.wordWrap

Disponibilité

Flash Player 6.

Usage

mon_txt.wordWrap

Description

Propriété : une valeur booléenne indiquant si le contenu du champ de texte passe automatiquement à la ligne. Si la valeur de wordWrap est true, le champ de texte passe automatiquement à la ligne ; si la valeur est false, le champ de texte ne passe pas automatiquement à la ligne.

TextField._x

Disponibilité

Flash Player 6.

Usage

mon_txt.x

Description

Propriété : un entier définissant la coordonnée *x* du champ de texte par rapport aux coordonnées locales du clip parent. Si un champ de texte se trouve dans le scénario principal, son système de coordonnées fait référence au coin supérieur gauche de la scène, comme (0, 0). Si le champ de texte se trouve dans un clip qui a subi des transformations, le champ de texte est dans le système de coordonnées local du clip le contenant. Donc, pour un clip ayant pivoté de 90 dans le sens inverse des aiguilles d'une montre, le champ de texte inclus hérite d'un système de coordonnées qui a pivoté de 90 dans le sens inverse des aiguilles d'une montre. Les coordonnées du champ de texte font référence à la position du point d'alignement.

Consultez également

TextField._xscale, TextField._y, TextField._yscale

TextField._xmouse

Disponibilité

Flash Player 6.

Usage

mon_txt._xmouse

Description

Propriété (lecture seule) : renvoie la coordonnée *x* de la position de la souris par rapport au champ de texte.

Consultez également

TextField._ymouse

TextField._xscale

Disponibilité

Flash Player 6.

Usage

mon_txt._xscale

Description

Propriété : détermine l'échelle horizontale (exprimée en pourcentage) du champ de texte telle qu'elle est appliquée à partir du point d'alignement du champ de texte. Le point d'alignement par défaut est (0,0).

Consultez également

TextField._x, TextField._y, TextField._yscale

TextField._y

Disponibilité

Flash Player 6.

Usage

mon_txt._y

Description

Propriété : la coordonnée *y* du champ de texte par rapport aux coordonnées locales du clip parent. Si un champ de texte se trouve dans le scénario principal, son système de coordonnées fait référence au coin supérieur gauche de la scène, comme (0, 0). Si le champ de texte se trouve dans un autre clip qui a subi des transformations, le champ de texte est dans le système de coordonnées local du clip le contenant. Donc, pour un clip ayant pivoté de 90 dans le sens inverse des aiguilles d'une montre, le champ de texte inclus hérite d'un système de coordonnées qui a pivoté de 90 dans le sens inverse des aiguilles d'une montre. Les coordonnées du champ de texte font référence à la position du point d'alignement.

Consultez également

TextField._x, TextField._xscale, TextField._yscale

TextField._ymouse

Disponibilité

Flash Player 6.

Usage

mon_txt.`_ymouse`

Description

Propriété (lecture seule) : indique la coordonnée *y* de la position de la souris par rapport au champ de texte.

Consultez également

`TextField._xmouse`

TextField._yscale

Disponibilité

Flash Player 6.

Usage

mon_txt.`_yscale`

Description

Propriété : échelle verticale (exprimée en pourcentage) du champ de texte telle qu'elle est appliquée à partir du point d'alignement du champ de texte. Le point d'alignement par défaut est (0,0).

Consultez également

`TextField._x`, `TextField._xscale`, `TextField._y`

Classe TextFormat

Disponibilité

Flash Player 6.

Description

La classe TextFormat donne les informations sur les format des caractères.

Vous devez utiliser le constructeur `new TextFormat()` pour créer un objet TextFormat avant d'en appeler les méthodes.

Vous pouvez définir les paramètres de TextFormat sur `null` pour indiquer qu'ils ne sont pas définis. Lorsque vous appliquez un objet TextFormat à un champ de texte à l'aide de `TextField.setTextFormat`, seules ses propriétés définies sont appliquées, comme dans l'exemple suivant :

```
mon_fmt = new TextFormat();
mon_fmt.bold = true;
mon_txt.setTextFormat(mon_fmt);
```

Ce code crée d'abord un objet TextFormat vide, avec toutes ses propriétés non définies, puis applique la propriété `bold` à une valeur définie.

Le code `mon_txt.setTextFormat(mon_fmt)` ne modifie que la propriété `bold` du format de texte par défaut du champ de texte, étant donné que la propriété `bold` est la seule qui soit définie dans `mon_fmt`. Tous les autres aspects du format de texte par défaut du champ de texte restent inchangés.

Lorsque `TextField.getTextFormat()` est invoqué, un objet TextFormat est renvoyé avec toutes ses propriétés définies, aucune propriété ne restant `null`.

Méthodes de la classe TextFormat

Méthode	Description
`TextFormat.getTextExtent()`	Renvoie les informations relatives aux mesures du texte dans une chaîne de texte.

Propriétés de la classe TextFormat

Propriété	Description
`TextFormat.align`	Indique l'alignement d'un paragraphe.
`TextFormat.blockIndent`	Indique l'indentation d'un bloc, en points.
`TextFormat.bold`	Indique si le texte apparaît en gras.
`TextFormat.bullet`	Indique si le texte fait partie d'une liste à puces.
`TextFormat.color`	Indique la couleur du texte.
`TextFormat.font`	Indique le nom de la police du texte avec ce format.
`TextFormat.indent`	Indique l'indentation, de la marge gauche au premier caractère du paragraphe.
`TextFormat.italic`	Indique si le texte apparaît en italique.
`TextFormat.leading`	Indique le nombre d'espaces verticaux (*espacement*) entre les lignes.
`TextFormat.leftMargin`	Indique la marge gauche du paragraphe, en points.
`TextFormat.rightMargin`	Indique la marge droite du paragraphe, en points.
`TextFormat.size`	Indique la taille du texte, en points.
`TextFormat.tabStops`	Spécifie des taquets de tabulation personnalisés.
`TextFormat.target`	Indique la fenêtre de navigateur dans laquelle un hyperlien est affiché.
`TextFormat.underline`	Indique si le texte est souligné.
`TextFormat.url`	Indique l'URL à laquelle le texte est lié.

Constructeur de la classe TextFormat

Disponibilité

Flash Player 6.

Usage

```
new TextFormat([police, [taille, [couleur, [gras, [italique, [souligné, [url,
  [cible, [alignement, [margeGauche, [margeDroite, [indentation,
  [espacement]]]]]]]]]]]])
```

Paramètres

police Le nom d'une police pour le texte, sous forme de chaîne.

taille Un entier indiquant la taille en points.

couleur La couleur du texte utilisant ce format de texte. Un nombre contenant trois composants RVB 8 bits ; par exemple, 0xFF0000 est le rouge, 0x00FF00 est le vert.

gras Une valeur booléenne indiquant si le texte apparaît en gras.

italique Une valeur booléenne indiquant si le texte apparaît en italique.

souligné Une valeur booléenne indiquant si le texte apparaît souligné.

url L'URL à laquelle le texte de ce format est lié. Si *url* est une chaîne vide, le texte n'a pas d'hyperlien.

cible La fenêtre cible dans laquelle est affiché l'hyperlien. Si la fenêtre ciblée est une chaîne vide, le texte est affiché dans la fenêtre cible par défaut, `_self`. Si le paramètre *url* est défini sur une chaîne vide ou sur la valeur `null`, vous pouvez récupérer ou définir cette propriété, mais elle n'aura aucun effet.

alignement L'alignement du paragraphe, représenté sous forme de chaîne. Si `"left"`, le paragraphe est aligné à gauche. Si `"center"`, le paragraphe est centré. Si `"right"`, le paragraphe est aligné à droite.

margeGauche Indique la marge gauche du paragraphe, en points.

margeDroite Indique la marge droite du paragraphe, en points.

indentation Un entier indiquant l'indentation de la marge gauche au premier caractère du paragraphe.

espacement Un nombre indiquant l'espace séparant les lignes.

Renvoie

Rien.

Description

Constructeur : crée un objet TextFormat avec les propriétés spécifiées. Vous pouvez ensuite changer les propriétés de l'objet TextFormat pour changer le formatage des champs de texte.

N'importe quel paramètre peut être défini sur la valeur `null` pour indiquer qu'il n'est pas défini. Tous les paramètres sont facultatifs, tous les paramètres omis étant considérés comme `null`.

TextFormat.align

Disponibilité

Flash Player 6.

Usage

mon_fmt.align

Description

Propriété : indique l'alignement du paragraphe, représenté sous forme de chaîne. L'alignement du paragraphe, représenté sous forme de chaîne. Si "left", le paragraphe est aligné à gauche. Si "center", le paragraphe est centré. Si "right", le paragraphe est aligné à droite. La valeur par défaut est null, qui indique que la propriété n'est pas définie.

TextFormat.blockIndent

Disponibilité

Flash Player 6.

Usage

mon_fmt.blockIndent

Description

Propriété : un nombre qui indique l'indentation d'un bloc, en points. L'indentation de bloc est appliquée à un bloc de texte entier ; c'est-à-dire, à toutes les lignes du texte. A l'inverse, l'indentation normale (TextFormat.indent) affecte seulement la première ligne de chaque paragraphe. Si cette propriété est null, l'objet TextFormat ne spécifie pas d'indentation de bloc.

TextFormat.bold

Disponibilité

Flash Player 6.

Usage

mon_fmt.bold

Description

Propriété : une valeur booléenne indiquant si le texte apparaît en gras. La valeur par défaut est null, qui indique que la propriété n'est pas définie.

TextFormat.bullet

Disponibilité

Flash Player 6.

Usage

mon_fmt.bullet

Description

Propriété : une valeur booléenne indiquant si le texte fait partie d'une liste à puces. Dans une liste à puces, chaque paragraphe de texte est indenté. Un symbole de puce est affiché à gauche de la première ligne de chaque paragraphe. La valeur par défaut est `null`.

TextFormat.color

Disponibilité

Flash Player 6.

Usage

mon_fmt.color

Description

Propriété : indique la couleur du texte. Un nombre contenant trois composants RVB 8 bits ; par exemple, 0xFF0000 est le rouge, 0x00FF00 est le vert.

TextFormat.font

Disponibilité

Flash Player 6.

Usage

mon_fmt.font

Description

Propriété : le nom de la police pour le texte dans ce format de texte, sous forme de chaîne. La valeur par défaut est `null`, qui indique que la propriété n'est pas définie.

TextFormat.getTextExtent()

Disponibilité

Flash Player 6. Le paramètre facultatif *width* est supporté dans Flash Player 7.

Usage

```
mon_fmt.getTextExtent(texte, [largeur])
```

Paramètres

texte Une chaîne.

largeur Un nombre facultatif représentant la largeur, en pixels, à partir de laquelle le texte spécifié doit faire un retour à la ligne.

Renvoie

Un objet avec les propriétés `width`, `height`, `ascent`, `descent`, `textFieldHeight` et `textFieldWidth`.

Description

Méthode : renvoie les informations concernant les mesures du texte de la chaîne de texte *texte* au format spécifié par *mon_fmt*. La chaîne texte est considéré comme un texte ordinaire (et non HTML).

La méthode renvoie un objet avec six propriétés : `ascent`, `descent`, `width`, `height`, `textFieldHeight` et `textFieldWidth`. Toutes les mesures sont indiquées en pixels.

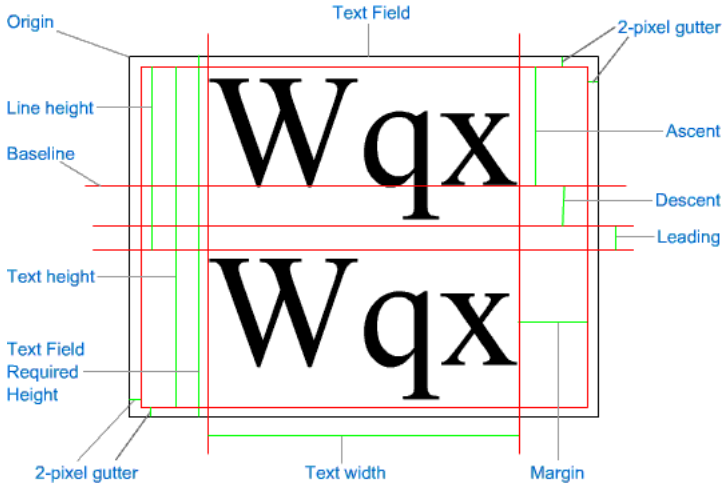
Si un paramètre *width* est spécifié, l'habillage du texte est appliqué au texte spécifié. Cela permet de déterminer la hauteur à laquelle le champ de texte affiche la totalité du texte spécifié.

Les mesures `ascent` et `descent` fournissent respectivement la distance au-dessus et au-dessous de la ligne de base d'une ligne de texte. La ligne de base de la première ligne de texte est positionnée à l'origine du champ de texte à laquelle est additionnée sa mesure `ascent`.

Les mesures `width` et `height` fournissent la largeur et la hauteur de la chaîne de texte. Les mesures `textFieldHeight` et `textFieldWidth` indiquent la hauteur et la largeur requises d'un objet champ de texte pour afficher la totalité de la chaîne de texte. Les champs de texte sont entourés d'une bordure large de deux pixels. Ainsi, la valeur `textFieldHeight` est égale à la valeur `height` + 4 ; de même, la valeur `textFieldWidth` est toujours égale à la valeur `width` + 4.

Si vous créez un champ de texte sur la base métrique, utilisez `textFieldHeight` plutôt que `height` et `textFieldWidth` plutôt que `width`.

L'illustration ci-dessous répertorie ces différentes mesures.



Lorsque vous définissez un objet TextFormat, veillez à ce que l'ensemble des attributs soient identiques à ceux du champ de texte, notamment la police, la taille de police et l'espacement. La valeur par défaut de l'espacement est 2.

Exemple

Cet exemple crée un champ de texte d'une ligne juste assez grand pour afficher une chaîne de texte au format spécifié.

```
var texte = "Petite chaîne";

// Créer un objet TextFormat
// et appliquer ses propriétés.
var txt_fmt = new TextFormat();
with(txt_fmt) {
  font = "Arial";
  bold = true;
}

// Obtenir des informations métriques sur la chaîne de texte
// avec le format spécifié.
var métrique = txt_fmt.getTextExtent(text);

// Créer un champ de texte juste assez grand pour afficher le texte.
this.createTextField ("champDeTexte", 0, 100, 100, métrique.textFieldWidth,
  métrique.textFieldHeight);
champDeTexte.border = true;
champDeTexte.wordWrap = true;
// Affecter la même chaîne de texte et
// l'objet TextFormat à l'objet TextField.
champDeTexte.text = text;
champDeTexte.setTextFormat(txt_fmt);
```

L'exemple suivant crée un champ de texte de plusieurs lignes, large de 100 pixels, dont la taille permet d'afficher une chaîne au format spécifié.

```
// Créer un objet TextFormat.
var txt_fmt:TextFormat= new TextFormat();
```

```
// Spécifier les propriétés du format pour l'objet TextFormat :
txt_fmt.font = "Arial";
txt_fmt.bold = true;
txt_fmt.leading = 4;

// La chaîne de texte à afficher
var texteAAfficher:String = "Macromedia Flash 7, à présent disponible avec des
  mesures métriques de texte améliorées.";

// Obtenir les informations relatives aux mesures du texte pour la chaîne,
// retour à la ligne à 100 pixels.
var métrique:Object = txt_fmt.getTextExtent(texteAAfficher, 100);

// Créer un nouvel objet TextField à l'aide des
// informations métriques obtenues.
this.createTextField ("champDeTexte", 0, 50, 50-metrics.ascent, 100,
  métrique.textFieldHeight)
champDeTexte.wordWrap = true;
champDeTexte.border = true;
// Affecter le texte et l'objet TextFormat à TextObject :
champDeTexte.text = texteAAfficher;
champDeTexte.setTextFormat(unFormat);
```

TextFormat.indent

Disponibilité

Flash Player 6.

Usage

mon_fmt.indent

Description

Propriété : un entier indiquant l'indentation de la marge gauche au premier caractère du paragraphe. La valeur par défaut est `null`, qui indique que la propriété n'est pas définie.

Consultez également

TextFormat.blockIndent

TextFormat.italic

Disponibilité

Flash Player 6.

Usage

mon_fmt.italic

Description

Propriété : une valeur booléenne indiquant si le texte dans ce format de texte apparaît en italique. La valeur par défaut est `null`, qui indique que la propriété n'est pas définie.

TextFormat.leading

Disponibilité

Flash Player 6.

Usage

mon_fmt.leading

Description

Propriété : nombre d'espaces verticaux (*espacement*) entre les lignes. La valeur par défaut est null, qui indique que la propriété n'est pas définie.

TextFormat.leftMargin

Disponibilité

Flash Player 6.

Usage

mon_fmt.leftMargin

Description

Propriété : la marge gauche du paragraphe, en points. La valeur par défaut est null, qui indique que la propriété n'est pas définie.

TextFormat.rightMargin

Disponibilité

Flash Player 6.

Usage

mon_fmt.rightMargin

Description

Propriété : la marge droite du paragraphe, en points. La valeur par défaut est null, qui indique que la propriété n'est pas définie.

TextFormat.size

Disponibilité

Flash Player 6.

Usage

mon_fmt.size

Description

Propriété : la taille, en points, du texte de ce format. La valeur par défaut est null, qui indique que la propriété n'est pas définie.

TextFormat.tabStops

Disponibilité

Flash Player 6.

Usage

mon_fmt.tabStops

Description

Propriété : spécifie des arrêts de tabulation personnalisés, sous forme d'un tableau d'entiers non négatifs. Chaque arrêt de tabulation est spécifié en points. Si des arrêts de tabulation personnalisés ne sont pas spécifiés (null), l'arrêt de tabulation par défaut est 4 (largeur de caractère moyenne).

TextFormat.target

Disponibilité

Flash Player 6.

Usage

mon_fmt.target

Description

Propriété : indique la fenêtre cible dans laquelle est affiché l'hyperlien. Si la fenêtre ciblée est une chaîne vide, le texte est affiché dans la fenêtre cible par défaut, _self. Si la propriété TextFormat.url est une chaîne vide ou null, vous pouvez obtenir ou définir cette propriété, mais elle n'aura aucun effet.

TextFormat.underline

Disponibilité

Flash Player 6.

Usage

mon_fmt.underline

Description

Propriété : une valeur booléenne qui indique si le texte utilisant le format est souligné (true) ou non (false). Ce soulignement est semblable à celui appliqué par la balise <U>, qui n'est pas un « véritable » soulignement étant donné qu'il ne saute pas correctement les jambages. La valeur par défaut est null, qui indique que la propriété n'est pas définie.

TextFormat.url

Disponibilité

Flash Player 6.

Usage

mon_fmt.url

Description

Propriété : indique l'URL à laquelle le texte de ce format est lié. Si la propriété `url` est une chaîne vide, le texte n'a pas d'hyperlien. La valeur par défaut est `null`, qui indique que la propriété n'est pas définie.

Objet TextSnapshot

Disponibilité

Programmation : Flash MX 2004.

Lecture : Fichiers SWF publiés pour Flash Player 6 ou ultérieur, lus dans Flash Player 7 ou ultérieur.

Description

Les objets TextSnapshot vous permettent d'utiliser du texte statique dans un clip. Vous pouvez les utiliser, par exemple, pour positionner le texte avec une plus grande précision que celle autorisée par le texte dynamique, mais le texte n'est toujours accessible qu'en mode lecture seule.

N'utilisez pas un constructeur pour créer un objet TextSnapshot ; il est renvoyé par `MovieClip.getTextSnapshot()`.

Méthodes de l'objet TextSnapshot

Méthode	Description
`TextSnapshot.findText()`	Renvoie la position de la première occurrence de texte spécifié.
`TextSnapshot.getCount()`	Renvoie le nombre de caractères.
`TextSnapshot.getSelected()`	Spécifie si une partie du texte de la plage spécifiée a été sélectionnée par `TextSnapshot.setSelected()`.
`TextSnapshot.getSelectedText()`	Renvoie une chaîne qui contient tous les caractères spécifiés par `TextSnapshot.setSelected()`.
`TextSnapshot.getText()`	Renvoie une chaîne contenant les caractères de la plage spécifiée.
`TextSnapshot.hitTestTextNearPos()`	Vous permet de déterminer quel caractère de l'objet se trouve sur ou près des coordonnées spécifiées.

Méthode	Description
TextSnapshot.setSelectColor()	Spécifie la couleur à utiliser lors de la mise en surbrillance des caractères qui ont été sélectionnés avec la commande TextSnapshot.setSelected().
TextSnapshot.setSelected()	Spécifie une plage de caractères à sélectionner ou à désélectionner.

TextSnapshot.findText()

Disponibilité

Programmation : Flash MX 2004.

Lecture : fichiers SWF publiés pour Flash Player 6 ou ultérieur, lus dans Flash Player 7 ou ultérieur.

Usage

```
maCapture.findText( indexDebut, texteArechercher, sensibleAlaCasse )
```

Paramètres

indexDebut Un entier spécifiant le point de départ dans *maCapture* à partir duquel effectuer la recherche du texte spécifié.

texteArechercher Une chaîne spécifiant le texte à rechercher. Si vous spécifiez une chaîne littérale plutôt qu'une variable de type String, mettez la chaîne entre guillemets.

sensibleAlaCasse Une valeur booléenne spécifiant si le texte dans *maCapture* doit correspondre à la casse de la chaîne dans *texteArechercher*.

Renvoie

La position de l'index basé sur zéro de la première occurrence du texte spécifié ou -1.

Description

Méthode : recherche l'objet TextSnapshot spécifié et renvoie la position de la première occurrence de *texteArechercher* trouvée au niveau de ou après *indexDébut*. Si *texteArechercher* n'est pas trouvé, la méthode renvoie -1.

Consultez également

TextSnapshot.getText()

TextSnapshot.getCount()

Disponibilité

Programmation : Flash MX 2004.

Lecture : fichiers SWF publiés pour Flash Player 6 ou ultérieur, lus dans Flash Player 7 ou ultérieur.

Usage

```
maCapture.getCount()
```

Paramètres

Aucun.

Renvoie

Un entier représentant le nombre de caractères dans l'objet TextSnapshot spécifié.

Description

Méthode : renvoie le nombre de caractères dans un objet TextSnapshot.

Consultez également

`TextSnapshot.getText()`

TextSnapshot.getSelected()

Disponibilité

Programmation : Flash MX 2004.

Lecture : fichiers SWF publiés pour Flash Player 6 ou ultérieur, lus dans Flash Player 7 ou ultérieur.

Usage

```
maCapture.getSelected(de, a)
```

Paramètres

de Un entier indiquant la position du premier caractère de *maCapture* à examiner. Les valeurs valides pour *de* sont comprises entre 0 et `TextSnapshot.getCount()`- 1. Si *de* est une valeur négative, 0 est utilisé.

a Un entier étant égal à 1+ l'index du dernier caractère de *maCapture* à examiner. Les valeurs valides pour *à* sont comprises entre 0 et `TextSnapshot.getCount()`. Le caractère indexé par le paramètre *a* n'est pas inclus dans la chaîne extraite. Si ce paramètre est omis, `TextSnapshot.getCount()` est utilisé. Si cette valeur est inférieure ou égale à la valeur de `de`, `de+1` est utilisé.

Renvoie

Une valeur booléenne `true`, si au moins un caractère de la plage donnée a été sélectionné par la commande `TextSnapshot.setSelected()` correspondante, `false` dans les autres cas.

Description

Méthode : renvoie une valeur booléenne spécifiant si un objet TextSnapshot contient le texte sélectionné dans la plage spécifiée.

Pour rechercher tous les caractères, affectez une valeur de 0 à *de* et une valeur `TextSnapshot.getCount()` (ou tout nombre très grand) à *a*. Pour rechercher un caractère unique, affectez une valeur de `de+1` à `a`.

Consultez également

`TextSnapshot.getSelectedText()`, `TextSnapshot.getText()`

TextSnapshot.getSelectedText()

Disponibilité

Programmation : Flash MX 2004.

Lecture : fichiers SWF publiés pour Flash Player 6 ou ultérieur, lus dans Flash Player 7 ou ultérieur.

Usage

```
maCapture.getSelectedText( [ includeLineEndings ] )
```

Paramètres

includeLineEndings Une valeur booléenne facultative spécifiant si les caractères newline sont insérés dans la chaîne renvoyée, à l'emplacement approprié. La valeur par défaut est *false*.

Renvoie

Une chaîne contenant tous les caractères spécifiés par la commande `TextSnapshot.setSelected()` correspondante.

Description

Méthode : renvoie une chaîne contenant tous les caractères spécifiés par la commande `TextSnapshot.setSelected()` correspondante. Si aucun caractère n'est sélectionné, une chaîne vide est renvoyée.

Si vous affectez une valeur `true` à *includeLineEndings*, les caractères newline sont insérés dans la chaîne renvoyée à l'emplacement jugé approprié. Dans ce cas, la chaîne renvoyée peut être plus longue que la plage saisie. Si *includeLineEndings* est défini sur `false` ou omis, le texte sélectionné est renvoyé sans que des caractères soient ajoutés.

Consultez également

`TextSnapshot.getSelected()`

TextSnapshot.getText()

Disponibilité

Programmation : Flash MX 2004.

Lecture : fichiers SWF publiés pour Flash Player 6 ou ultérieur, lus dans Flash Player 7 ou ultérieur.

Usage

```
maCapture.getText(de, a [, includeLineEndings ] )
```

Paramètres

de Un entier indiquant la position du premier caractère de *maCapture* à inclure dans la chaîne renvoyée. Les valeurs valides pour *de* sont comprises entre 0 et `TextSnapshot.getCount()`- 1. Si *de* est une valeur négative, 0 est utilisé.

a Un entier étant égal à 1+ l'index du dernier caractère de *maCapture* à examiner. Les valeurs valides pour *à* sont comprises entre 0 et `TextSnapshot.getCount()`. Le caractère indexé par le paramètre *a* n'est pas inclus dans la chaîne extraite. Si ce paramètre est omis, `TextSnapshot.getCount()` est utilisé. Si cette valeur est inférieure ou égale à la valeur de `de`, `de+1` est utilisé.

includeLineEndings Une valeur booléenne facultative spécifiant si les caractères newline sont insérés dans la chaîne renvoyée, à l'emplacement approprié. La valeur par défaut est *false*.

Renvoie

Une chaîne contenant les caractères de la plage spécifiée ou une chaîne vide si aucun caractère n'a été trouvé dans cette plage.

Description

Méthode : renvoie une chaîne contenant tous les caractères spécifiés par les paramètres *de* et *a*. Si aucun caractère n'est sélectionné, une chaîne vide est renvoyée.

Pour renvoyer tous les caractères, affectez une valeur de 0 pour *de* et `TextSnapshot.getCount()` (ou tout nombre très grand) pour *a*. Pour renvoyer un caractère unique, affectez une valeur `de+1` à `a`.

Si vous affectez une valeur `true` à *includeLineEndings*, les caractères newline sont insérés dans la chaîne renvoyée à l'emplacement jugé approprié. Dans ce cas, la chaîne renvoyée peut être plus longue que la plage saisie. Si *includeLineEndings* est défini sur `false` ou omis, le texte sélectionné est renvoyé sans que des caractères soient ajoutés.

Consultez également

`TextSnapshot.getSelectedText()`

TextSnapshot.hitTestTextNearPos()

Disponibilité

Programmation : Flash MX 2004.

Lecture : fichiers SWF publiés pour Flash Player 6 ou ultérieur, lus dans Flash Player 7 ou ultérieur.

Usage

```
maCapture.hitTestTextNearPos(x, y [, distanceMax] )
```

Paramètres

x Un nombre qui représente la coordonnée x du clip contenant le texte de *maCapture*.

y Un nombre qui représente la coordonnée y du clip contenant le texte de *maCapture*.

distanceMax Un nombre facultatif qui représente la distance maximale depuis *x*, *y* pour laquelle le texte est recherché. La distance est mesurée à partir du point central de chaque caractère. La valeur par défaut est 0.

Renvoie

Un entier représentant la valeur d'index du caractère de *maCapture* qui est la plus proche des coordonnées *x*, *y* spécifiées, ou -1 si aucun caractère n'est trouvé.

Description

Méthode : vous permet de déterminer quel caractère d'un objet TextSnapshot se trouve sur ou près des coordonnées *x*, *y* spécifiées pour le clip contenant le texte de *maCapture*.

Si vous omettez la valeur ou définissez *distanceMax* sur 0, l'emplacement spécifié par les coordonnées *x*, *y* doit se trouver à l'intérieur du cadre de délimitation de *maCapture*.

Consultez également

MovieClip.getTextSnapshot(), MovieClip._x, MovieClip._y

TextSnapshot.setSelectColor()

Disponibilité

Programmation : Flash MX 2004.

Lecture : fichiers SWF publiés pour Flash Player 6 ou ultérieur, lus dans Flash Player 7 ou ultérieur.

Usage

```
maCapture.setSelectColor(couleurHexa);
```

Paramètres

couleurHexa La couleur utilisée pour la bordure placée autour des caractères sélectionnés par la commande `TextSnapshot.setSelected()` correspondante, exprimée au format `0xRRGGBB`.

Renvoie

Rien.

Description

Méthode : spécifie la couleur à utiliser lors de la mise en surbrillance des caractères qui ont été sélectionnés avec la commande `TextSnapshot.setSelected()`. La couleur est toujours opaque ; vous ne pouvez pas spécifier une valeur de transparence.

TextSnapshot.setSelected()

Disponibilité

Programmation : Flash MX 2004.

Lecture : fichiers SWF publiés pour Flash Player 6 ou ultérieur, lus dans Flash Player 7 ou ultérieur.

Usage

```
maCapture.setSelected(de, a, selection)
```

Paramètres

de Un entier indiquant la position du premier caractère de *maCapture* à sélectionner. Les valeurs valides pour *de* sont comprises entre 0 et `TextSnapshot.getCount()`- 1. Si *de* est une valeur négative, 0 est utilisé.

a Un entier étant égal à 1+ l'index du dernier caractère de *maCapture* à examiner. Les valeurs valides pour *à* sont comprises entre 0 et `TextSnapshot.getCount()`. Le caractère indexé par le paramètre *a* n'est pas inclus dans la chaîne extraite. Si ce paramètre est omis, `TextSnapshot.getCount()` est utilisé. Si cette valeur est inférieure ou égale à la valeur de `de`, `de+1` est utilisé.

selection Une valeur booléenne indiquant si le texte doit être sélectionné (`true`) ou désélectionné (`false`).

Renvoie

Rien.

Description

Méthode : spécifie une plage de caractères d'un objet TextSnapshot à sélectionner ou à désélectionner. Un rectangle de couleur correspondant au cadre de délimitation du caractère est tracé derrière chacun des caractères sélectionnés. La couleur du cadre de délimitation est défini par `TextSnapshot.setSelectColor()`.

Pour sélectionner ou désélectionner tous les caractères, définissez une valeur de 0 pour *de* et `TextSnapshot.getCount()` (ou tout nombre très grand) pour *a*. Pour spécifier un caractère unique, affectez une valeur de `de+1` pour `a`.

Les caractères étant individuellement marqués comme sélectionnés, vous pouvez émettre plusieurs fois cette commande pour sélectionner plusieurs caractères ; cette opération n'annule pas la sélection des autres caractères définis par cette commande.

this

Disponibilité

Flash Player 5.

Usage

```
this
```

Description

Mot-clé : fait référence à un objet ou à une occurrence de clip. Lorsqu'un script est exécuté, `this` fait référence à l'occurrence de clip qui contient le script. Lorsqu'une méthode est appelée, `this` contient une référence à l'objet qui contient la méthode appelée.

Dans une action de gestionnaire d'événement `on` associée à un bouton, `this` fait référence au scénario contenant le bouton. Dans une action de gestionnaire d'événement `onClipEvent()` associée à un clip, `this` fait référence au scénario du clip.

Comme `this` est évalué dans le contexte du script qui le contient, vous ne pouvez pas utiliser `this` dans un script pour faire référence à une variable définie dans un fichier de classe :

```
// dans le fichier applyThis.as
class applyThis{
  var str:String = "Défini dans applyThis.as";
  function conctStr(x:String):String{
    return x+x;
  }

  function addStr():String{
    return str;
  }
}

// Utilisez le code suivant dans FLA pour tester l'animation
import applyThis;

var obj:applyThis = new applyThis();
var abj:applyThis = new applyThis();
abj.str = "défini dans FLA";

trace(obj.addStr.call(abj,null));    // défini dans FLA
trace(obj.addStr.call(this,null));   // non défini
trace(obj.addStr.call(obj,null));    // Défini dans applyThis.as
```

De même, pour appeler une fonction définie dans une classe dynamique, vous devez utiliser `this` pour l'étendue de la fonction :

```
// version incorrecte de simple.as
dynamic class simple{
  function callfunc(){
    trace(func());
  }

}
```

```
// version correcte de simple.as
dynamic class simple{
  function callfunc(){
    trace(this.func());
  }
}
// instructions du fichier FLA
import simple;
var obj:simple = new simple();
obj.num = 0;
obj.func = function():Boolean{
  renvoie true;
}
obj.callfunc(); // erreur de syntaxe avec version incorrecte de simple.as
```

Exemple

Dans l'exemple suivant, le mot-clé `this` fait référence à l'objet Cercle.

```
function Cercle(rayon) {
  this.rayon = rayon;
  this.aire = Math.PI * rayon * rayon;
}
```

Dans l'instruction suivante affectée à une image, le mot-clé `this` fait référence au clip courant.

```
// définit la propriété alpha du clip actuel à 20
this._alpha = 20;
```

Dans l'instruction suivante au sein d'un gestionnaire `onClipEvent()`, le mot-clé `this` fait référence au clip courant.

```
// pendant le chargement du clip, une opération startDrag()
// du clip courant.

onClipEvent(load) {
  startDrag (this, true);
}
```

Consultez également

`on()`, `onClipEvent()`

throw

Disponibilité

Flash Player 7.

Usage

```
throw expression
```

Description

Instruction : génère (« émet ») une erreur pouvant être gérée (« saisie ») par un bloc de code `catch{}` ou `finally{}`. Si une exception n'est pas saisie par un bloc `catch` ou `finally`, la représentation de chaîne de la valeur émise est envoyée vers le panneau de sortie.

Généralement, vous émettez des occurrences de la classe Error ou de ses sous-classes (consultez les exemples suivants).

Paramètres

`expression` Une expression ou un objet ActionScript.

Exemple

Dans cet exemple, une fonction nommée `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 émet une erreur.

```
function checkEmail(email:String) {
  if (email.indexOf("@") == -1) {
    throw new Error("L'adresse mail est invalide");
  }
}
```

Le code suivant appelle ensuite la fonction `checkEmail()` à l'intérieur d'un bloc de code `try`, transmettant le texte dans un champ de texte (`email_txt`) en tant que paramètre. Si le paramètre de la chaîne ne contient pas d'adresse électronique valide, le message d'erreur s'affiche dans le champ de texte (`erreur_txt`).

```
try
  checkEmail("Joe Smith");
} catch (e) {
  erreur_txt.text = e.toString();
}
```

Dans cet exemple, une sous-classe de la classe Error est émise. La fonction `checkEmail()` est modifiée de manière à émettre l'occurrence de cette sous-classe. Pour plus d'informations, consultez *Création de sous-classes*, page 171.

```
// Définir la sous-classe Error InvalidEmailError
// Dans InvalidEmailError.as:
class InvalidEmailAddress extends Error {
  var message = "L'adresse mail est invalide.";
}
function checkEmail(email:String) {
  if (email.indexOf("@") == -1) {
    throw new InvalidEmailAddress();
  }
}
```

Consultez également

Classe Error, `try..catch..finally`

toggleHighQuality()

Disponibilité

Flash 2 ; déconseillé et remplacé par `_quality`.

Usage

```
toggleHighQuality()
```

Paramètres

Aucun.

Renvoie

Rien.

Description

Fonction déconseillée : active ou désactive l'anti-aliasing dans Flash Player. L'anti-aliasing permet de lisser les bords des objets et ralentit la lecture du fichier SWF. Cette action affecte tous les fichiers SWF de Flash Player.

Exemple

Le code suivant peut être appliqué à un bouton qui, lorsque cliqué, active ou désactive l'anti-aliasing :

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

Consultez également

`_highquality`, `_quality`

trace()

Disponibilité

Flash Player 4.

Usage

`trace(`*`expression`*`)`

Paramètres

`expression` Une expression à évaluer. Lorsqu'un fichier SWF est ouvert dans l'outil de programmation Flash (via la commande Tester l'animation), la valeur du paramètre *`expression`* est affichée dans le panneau de sortie.

Renvoie

Rien.

Description

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

Utilisez cette action pour enregistrer des remarques de programmation ou pour afficher des messages dans le panneau de sortie pendant le test d'une animation. Utilisez le paramètre *`expression`* pour vérifier si une condition existe ou pour afficher les valeurs dans le panneau de sortie. L'action `trace()` est similaire à la fonction `alert` de JavaScript.

Vous pouvez utiliser la commande Omettre les actions Trace des paramètres de publication pour supprimer les actions `trace()` du fichier SWF exporté.

Exemple

Cet exemple provient d'un jeu dans lequel une occurrence de clip déplaçable nommée `mon_mc` doit être relâchée sur une cible spécifique. Une instruction conditionnelle évalue la propriété `_droptarget` et exécute différentes actions en fonction de l'endroit où `mon_mc` est relâché. L'action `trace()` est utilisée à la fin du script pour évaluer l'emplacement du clip `mon_mc` et affiche le résultat dans le panneau de sortie. Si `mon_mc` ne se comporte pas de la manière prévue (par exemple, s'il atteint la mauvaise cible), les valeurs envoyées vers le panneau de sortie par l'action `trace()` vous aident à déterminer le problème dans le script.

```
on(press) {
  mon_mc.startDrag();
}

on(release) {
  if(eval(_droptarget) != target) {
    mon_mc._x = mon_mc_xValue;
    mon_mc._y = mon_mc_yValue;
  } else {
    var mon_mc_xValue = mon_mc._x;
    var mon_mc_yValue = mon_mc._y;
    target = "_root.champ";
  }
  trace("mon_mc_xValue = " + mon_mc_xValue);
  trace("mon_mc_xValue = " + mon_mc_xValue);
  stopDrag();
}
```

true

Disponibilité

Flash Player 5.

Usage

```
true
```

Description

Constante : une valeur booléenne unique représentant l'opposé de `false`.

Consultez également

`false`

try..catch..finally

Disponibilité

Flash Player 7.

Usage

```
try
  // ... bloc try ...
} finally {
  // ... bloc finally ...
}
try
  // ... bloc try ...
} catch(erreur[:TypeDerreur1]) {
  // ... bloc catch ...
} [catch(erreur[:TypeDerreurN]) {
  // ... bloc catch ...
}] [finally {
  // ... bloc finally ...
}]
```

Paramètres

erreur L'expression émise à partir d'une instruction *throw*, généralement une occurrence de la classe Error ou de sa sous-classe.

TypeDerreur Un type facultatif spécifiant l'identifiant *error*. La clause `catch` n'intercepte que les erreurs du type spécifié.

Description

Mots-clés : renferment un bloc de code dans lequel une erreur peut se produire, puis répondent à l'erreur. Si un code contenu dans le bloc de code `try` émet une erreur (à l'aide de l'action `throw`), contrôlez la transmission au bloc `catch`, s'il existe, puis au bloc de code `finally`, s'il existe. Le bloc `finally` s'exécute toujours, qu'une erreur ait été émise ou non. Si le code à l'intérieur du bloc `try` n'émet pas d'erreur (c'est à dire si le bloc `try` se termine normalement) alors le code du bloc `finally` s'exécute. Le bloc `finally` s'exécute même si le bloc `try` existe, à l'aide d'une instruction `return`.

Un bloc `try` doit être suivi d'un bloc `catch`, un bloc `finally` ou les deux. Un bloc unique `try` peut avoir plusieurs blocs `catch` mais seulement un bloc `finally`. Vous pouvez imbriquer les blocs `try` d'autant de profondeurs de niveau requises.

Le paramètre *erreur* spécifié dans un gestionnaire `catch` doit être un identifiant simple tel que `e`, `theException` ou `x`. La variable dans un gestionnaire `catch` peu également être *typed*. Lorsqu'elles sont utilisées avec plusieurs blocs `catch`, les erreurs tapées permettent de saisir plusieurs types d'erreur émises d'un bloc `try` unique.

Si l'exception émise est un objet, le type correspond si l'objet émis est une sous-classe du type spécifié. Si une erreur d'un type spécifique est émise, le bloc `catch` qui gère l'erreur correspondante est exécuté. Si une exception émise n'est pas du type spécifié, le bloc `catch` ne s'exécute pas et l'exception est automatiquement émise du bloc `try` vers un gestionnaire `catch` qui lui correspond.

Si une erreur est émise au sein d'une fonction et que la fonction n'inclut pas de gestionnaire `catch`, l'interprète d'ActionScript interrompt cette fonction, de même que les fonctions d'appel, jusqu'à ce qu'un bloc `catch` soit repéré. Durant ce processus, les gestionnaires `finally` sont appelés à tous les niveaux.

Exemple

L'exemple suivant montre comment créer une instruction `try..finally`. Etant donné que le code dans le bloc `finally` est forcément exécuté, il est généralement utilisé pour réaliser un code de « nettoyage » après l'exécution d'un bloc `try`. Dans cet exemple, le bloc `finally` est utilisé pour supprimer un objet ActionScript, qu'une erreur se produise ou non.

```
var compte = new Account()
try
  var valRetour = compte.getAccountInfo();
  if(valRetour != 0) {
    throw new Error("Erreur d'obtention des informations sur le compte.");
  }
}
finally {
  // Supprimer l'objet "compte", quelle que soit la situation.
  if(compte != null) {
    delete compte;
  }
}
```

L'exemple suivant illustre une instruction `try..catch`. Le code dans le bloc `try` est exécuté. Si une exception est émise par un code dans le bloc `try`, le contrôle transmet au bloc `catch`, qui affiche le message d'erreur dans un champ de texte à l'aide de la méthode `Error.toString()`.

```
var compte = new Account()
try
  var valRetour = compte.getAccountInfo();
  if(valRetour != 0) {
    throw new Error("Erreur d'obtention des informations sur le compte.");
  }
} catch (e) {
  état_txt.text = e.toString();
}
```

L'exemple suivant illustre un bloc de code `try` avec plusieurs blocs de code `catch` tapés. En fonction du type d'erreur survenue, le bloc de code `try` émet un type d'objet différent. Dans ce cas, `monEnsembleDEnregistrements` est une occurrence d'une classe (hypothétique) nommée EnsembleDEnregistrements dont la méthode `sortRows()` peut émettre deux types d'erreur différents : RecordSetException et MalformedRecord.

Dans cet exemple, les objets RecordSetException et MalformedRecord sont des sous-classes de la classe Error. Chacune est définie dans son propre fichier de classe AS. Pour plus d'informations, consultez le Chapitre 9, *Création de classes avec ActionScript 2.0*, page 163.

```
// Dans RecordSetException.as:
class RecordSetException extends Error {
  var message = "Une exception de définition d'enregistrement s'est produite."
}
// Dans MalformedRecord.as:
class MalformedRecord extends Error {
  var message = "Une exception d'enregistrement mal formulé s'est produite."
}
```

Dans la méthode `sortRows()` de la classe RecordSet, l'un des objets error préalablement défini est émis en fonction du type d'exception survenue. Le fragment de code suivant montre à quoi le code peut ressembler.

```
// Dans le fichier de classe RecordSet.as...
function sortRows() {
  ...
  if(recordSetErrorCondition) {
    throw new RecordSetException();
  }
  if(malFormedRecordCondition) {
    throw new MalformedRecord();
  }
  ...
}
```

Enfin, dans un autre fichier AS ou script FLA, le code suivant invoque la méthode `sortRows()` dans une occurrence de la classe RecordSet. Elle définit les blocs `catch` pour chaque type d'erreur émise par `sortRows()`.

```
try
  monEnsembleDEnregistrements.sortRows();
} catch (e:RecordSetException) {
  trace("Une exception de définition d'enregistrement a été saisie");
} catch (e:MalformedRecord) {
  trace("Une exception d'enregistrement mal formulé a été saisie");
}
```

Consultez également

Classe Error, `throw`, `class`, `extends`

typeof

Disponibilité

Flash Player 5.

Usage

`typeof(`*`expression`*`)`

Paramètres

`expression` Chaîne, clip, bouton, objet ou fonction.

Description

Opérateur : un opérateur unaire placé avant un paramètre unique. L'opérateur `typeof` oblige l'interprète de Flash à évaluer *`expression`* ; le résultat est une chaîne spécifiant si l'expression est une chaîne, un clip, un objet, une fonction, un nombre ou une valeur booléenne. Le tableau suivant indique les résultats de l'opérateur `typeof` sur chaque type d'expression.

Paramètre	Résultat
String	`chaîne`
Clip	`clip`
Button	`objet`
TextField	`objet`
Number	`nombre`
Boolean	`valeur booléenne`
Object	`objet`
Function	`fonction`

undefined

Disponibilité

Flash Player 5.

Usage

```
undefined
```

Paramètres

Aucun.

Renvoie

Rien.

Description

Une valeur spéciale, généralement utilisée pour indiquer qu'aucune valeur n'a encore été affectée à une variable. Une référence à une valeur non définie renvoie la valeur spéciale `undefined`. Le code ActionScript `typeof(undefined)` renvoie la chaîne `"undefined"`. La seule valeur de type `undefined` est `undefined`.

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

La valeur `undefined` est similaire à la valeur spéciale `null`. Lorsque `null` et `undefined` sont comparés avec l'opérateur d'égalité, ils apparaissent comme égaux.

Exemple

Dans cet exemple, la variable `x` n'a pas été déclarée et, par conséquent, a la valeur `undefined`. Dans la première section de code, l'opérateur d'égalité (==) compare la valeur `x` à la valeur `undefined` et le résultat approprié est envoyé vers le panneau de sortie. Dans la deuxième section de code, l'opérateur d'égalité compare les valeurs `null` et `undefined`.

```
// x n'a pas été déclaré
trace ("La valeur de x est " + x);
if (x == undefined) {
  trace ("x est undefined");
} else {
  trace ("x n'est pas undefined");
}

trace ("typeof (x) est " + typeof (x));
if (null == undefined) {
  trace ("null et undefined sont égaux");
} else {
  trace ("null et undefined ne sont pas égaux");
}
```

Le résultat suivant est affiché dans le panneau de sortie.

```
La valeur de x est undefined
x est undefined
typeof (x) est undefined
null et undefined sont égaux
```

unescape

Disponibilité

Flash Player 5.

Usage

```
unescape(x)
```

Paramètres

x Une chaîne avec des séquences hexadécimales à échapper.

Renvoie

Une chaîne décodée à partir d'un paramètre URL encodé.

Description

Fonction : évalue le paramètre *x* comme une chaîne, décode la chaîne d'un format de code URL (convertissant toutes les séquences hexadécimales en caractères ASCII) et renvoie la chaîne.

Exemple

L'exemple suivant illustre le processus de conversion d'échappement en non-échappement.

```
escape("Bonjour{[Monde]}");
```

Le résultat d'échappement est le suivant :

```
("Bonjour%7B%5BMonde%5D%7D');
```

Utilisez `unescape` pour retourner au format original :

```
unescape("Bonjour%7B%5BMonde%5D%7D")
```

Le résultat est le suivant :

```
Bonjour{[Monde]}
```

unloadMovie()

Disponibilité

Flash Player 3.

Usage

```
unloadMovie(cible)
```

Paramètres

cible Le chemin cible d'un clip.

Renvoie

Rien.

Description

Fonction : supprime de Flash Player un clip chargé au moyen de `loadMovie()`. Pour purger une animation chargée au moyen de `loadMovieNum()`, utilisez `unloadMovieNum()` au lieu de `unloadMovie()`.

Exemple

L'exemple suivant purge le clip `draggable_mc` du scénario principal et charge l'animation `animation.swf` dans le niveau 4.

```
on(press) {
    unloadMovie ("_root.draggable_mc");
    loadMovieNum ("animation.swf", 4);
}
```

L'exemple suivant purge l'animation chargée dans le niveau 4.

```
on(press) {
    unloadMovieNum (4);
}
```

Consultez également

`loadMovie()`, `MovieClipLoader.unloadClip()`

unloadMovieNum()

Disponibilité

Flash Player 3.

Usage

```
unloadMovieNum(niveau)
```

Paramètres

niveau Le niveau (_levelN) d'une animation chargée.

Renvoie

Rien.

Description

Fonction : supprime de Flash Player une animation chargée au moyen de `loadMovieNum()`. Pour purger une animation chargée au moyen de `loadMovie()`, utilisez `unloadMovie()` au lieu de `unloadMovieNum()`.

Consultez également

`loadMovie()`, `loadMovieNum()`, `unloadMovie()`

updateAfterEvent()

Disponibilité

Flash Player 5.

Usage

```
updateAfterEvent()
```

Paramètres

Aucun.

Renvoie

Rien.

Description

Fonction : met à jour les informations affichées (indépendamment de la valeur des images par seconde définie pour l'animation) lorsque vous l'appelez dans un gestionnaire `onClipEvent()` ou dans une fonction ou méthode transmise à `setInterval()`. Flash ignore les appels à `updateAfterEvent` qui ne sont pas dans un gestionnaire `onClipEvent()` ou qui ne font pas partie d'une fonction ou d'une méthode transmise à `setInterval()`.

Consultez également

`onClipEvent()`, `setInterval()`

var

Disponibilité

Flash Player 5.

Usage

```
var nomDeVariable1 [= valeur1] [...,nomDeVariableN [=valeurN]]
```

Paramètres

nomDeVariable Un identifiant.

valeur La valeur affectée à la variable.

Renvoie

Rien.

Description

Instruction : utilisée pour déclarer les variables locales ou du scénario.

- Si vous déclarez les variables dans une fonction, les variables sont locales. Elles sont définies pour la fonction et expirent à la fin de l'appel de la fonction.
- Si les variables ne sont pas déclarées au sein d'un bloc ({}), mais que la liste d'actions a été exécutée avec une action `call()`, les variables sont locales et expirent à la fin de la liste courante.
- Si les variables ne sont pas déclarées au sein d'un bloc et que la liste des actions courantes n'a pas été exécutée avec l'action `call()`, ces variables sont interprétées en tant que variables de scénario. Vous n'avez pas à utiliser `var` pour déclarer les variables de scénario.

Vous ne pouvez pas déclarer une variable dont le domaine inclut un autre objet en tant que variable locale :

```
mon_array.length = 25;     // ok
var mon_array.length = 25; // erreur de syntaxe
```

Si vous utilisez `var`, vous pouvez taper la variable de manière stricte ; consultez *Typage strict des données*, page 40.

Remarque : Les classes définies dans les scripts externes prennent aussi en charge les étendues des variables publiques, privées et statiques. Consultez le Chapitre 9, *Création de classes avec ActionScript 2.0*, page 163, `private`, `public` et `static`.

Classe Video

Disponibilité

Flash Player 6 ; la fonction de lecture des fichiers Flash Video (FLV) a été rajoutée dans Flash Player 7.

Description

La classe Video vous permet d'afficher une vidéo en continu en direct sur la scène sans l'intégrer dans votre fichier SWF. Vous capturez la vidéo en utilisant `Camera.get()`. Dans les fichiers publiés pour Flash Player 7 et les versions ultérieures, vous pouvez également utiliser la classe Video pour lire les fichiers Flash Video (FLV) à partir d'un serveur HTTP ou du système de fichiers local. Pour plus d'informations, consultez *Lecture dynamique des fichiers FLV externes*, page 209, Classe NetConnection et Classe NetStream.

Vous pouvez utiliser un objet Video comme un clip. Comme pour les autres objets placés dans la scène, vous pouvez contrôler les différentes propriétés des objets Video. Par exemple, vous pouvez déplacer l'objet Video autour de la scène à l'aide de ses propriétés `_x` et `_y` ; vous pouvez également le redimensionner à l'aide de ses propriétés `_height` et `_width`.

Pour afficher le flux vidéo, placez dans un premier temps un objet Video sur la scène. Ensuite, utilisez `Video.attachVideo()` pour rattacher le flux vidéo à l'objet Video.

Pour placer un objet Video sur la scène :

1. Si le panneau Bibliothèque n'est pas visible, sélectionnez Fenêtre > Bibliothèque pour l'afficher.
2. Ajoutez un objet Video intégrée à la bibliothèque en cliquant sur le menu d'options situé à droite de la barre de titre du panneau Bibliothèque et en sélectionnant Nouvelle vidéo.
3. Déplacez l'objet Video sur la scène et servez-vous de l'inspecteur des propriétés pour lui attribuer un nom d'occurrence unique tel que `ma_video`. (Ne l'appelez pas Video.)

Méthodes de la classe Video

Méthode	Description
`Video.attachVideo()`	Spécifie un flux vidéo à afficher dans les limites de l'objet Video sur la scène.
`Video.clear()`	Efface l'image courante affichée dans l'objet Video.

Propriétés de la classe Video

Propriété	Description
`Video.deblocking`	Spécifie le comportement du filtre de déblocage que le compresseur vidéo applique, lorsque cela est nécessaire, à la lecture de la vidéo.
`Video.height`	Lecture seule ; la hauteur du flux vidéo, en pixels.
`Video.smoothing`	Spécifie si la vidéo doit être lissée (interpolée) lorsqu'elle est redimensionnée.
`Video.width`	Lecture seule ; la largeur du flux vidéo, en pixels.

Video.attachVideo()

Disponibilité

Flash Player 6 ; la possibilité de travailler avec les fichiers Flash Video (FLV) a été rajoutée dans Flash Player 7.

Usage

```
ma_video.attachVideo(source)
```

Paramètres

source Un objet Camera qui capture les données vidéo ou un objet NetStream. Pour interrompre la connexion à l'objet Video, choisissez `null` pour *source*.

Renvoie

Rien.

Description

Méthode : spécifie un flux vidéo (*source*) à afficher dans les limites de l'objet Video sur la scène. Le flux vidéo est un fichier FLV en cours d'affichage via la commande `NetStream.play()`, un objet Camera ou `null`. Si *source* est `null`, la vidéo n'est plus lue dans l'objet Video.

Vous n'avez pas besoin d'utiliser cette méthode si le fichier FLV contient uniquement des données audio ; la partie audio d'un fichier FLV est lue automatiquement lorsque la commande `NetStream.play()` est activée.

Pour contrôler les données audio associées à un fichier FLV, vous pouvez utiliser la méthode `MovieClip.attachAudio()` pour acheminer ces données vers le clip ; vous pouvez ensuite créer un objet Sound pour contrôler certains aspects de ces données audio. Pour plus d'informations, consultez `MovieClip.attachAudio()`.

Exemple

L'exemple suivant lit localement une vidéo en direct.

```
ma_cam = Camera.get();
ma_video.attachVideo(ma_cam); // ma_video est un objet Video sur la scène
```

L'exemple suivant montre la lecture d'un fichier précédemment enregistré nommé maVidéo.flv, stocké dans le même dossier que le fichier SWF.

```
var nc:NetConnection = new NetConnection();
nc.connect(null);
var ns:NetStream = new NetStream(mon_nc);
ma_video.attachVideo(ns); // ma_video est un objet Video sur la scène
ns.play("myVideo.flv");
```

Consultez également

Classe Camera, Classe NetStream

Video.clear()

Disponibilité

Flash Player 6.

Usage

```
ma_video.clear()
```

Paramètres

Aucun.

Renvoie

Rien.

Description

Méthode : efface l'image courante affichée dans l'objet Video. Cela est utile, par exemple, si vous voulez afficher des informations en attente sans masquer l'objet Video.

Consultez également

`Video.attachVideo()`

Video.deblocking

Disponibilité

Flash Player 6.

Usage

```
ma_video.deblocking
ma_video.deblocking = paramètre
```

Description

Propriété : spécifie le comportement du filtre de déblocage que le compresseur vidéo applique, lorsque cela est nécessaire, à la lecture de la vidéo. Les valeurs suivantes sont acceptables pour *paramètre* :

- 0 (valeur par défaut) : Laisse le compresseur vidéo appliquer le filtre de déblocage lorsque cela est nécessaire.
- 1 : N'utilise jamais le filtre de déblocage.
- 2 : Utilise toujours le filtre de déblocage.

Le filtre de déblocage a un effet sur l'ensemble de la performance de la lecture et il n'est généralement pas nécessaire avec la vidéo à large bande. Si votre système n'est pas assez puissant, vous pourriez rencontrer des difficultés lors de la lecture de la vidéo lorsque le filtre est activé.

Video.height

Disponibilité

Flash Player 6.

Usage

```
ma_video.height
```

Description

Propriété (lecture seule) : un entier spécifiant la hauteur du flux vidéo, en pixels. Pour les flux vidéo en direct, cette valeur est identique à la propriété `Camera.height` de l'objet Camera qui capture le flux vidéo. Pour les fichiers FLV, cette valeur correspond à la hauteur du fichier exporté en tant que fichier FLV.

Cette propriété peut vous servir, par exemple, à vous assurer que l'utilisateur regarde une vidéo dont la taille correspond à la taille de la vidéo au moment de sa capture, quelle que soit la taille réelle de l'objet Video sur la scène.

Exemple

Usage 1 : L'exemple suivant montre la définition des valeurs de hauteur et de largeur de l'objet Video, en fonction des valeurs d'un fichier FLV. Vous devez appeler ce code après l'invocation de `NetStream.onStatus`, lorsque la propriété `code` est définie sur `NetStream.Buffer.Full`. Si vous appelez ce code alors que valeur de la propriété `code` est `NetStream.Play.Start`, les valeurs de hauteur et de largeur sont 0, car l'objet Video n'a pas encore la hauteur et la largeur du fichier FLV chargé.

```
// Clip est le nom d'occurrence du clip
// qui contient l'objet vidéo "ma_vidéo".
_root.Clip._width = _root.Clip.ma_video.width;
_root.Clip._height = _root.Clip.ma_video.height;
```

Usage 2 : L'exemple suivant permet à l'utilisateur d'appuyer sur un bouton pour ajuster la hauteur et la largeur d'un flux vidéo affiché dans Flash Player avec une hauteur et une largeur égales à celles de la vidéo lorsqu'elle a été capturée.

```
on (release) {
  _root.ma_video._width = _root.ma_video.width
  _root.ma_video._height = _root.ma_video.height
}
```

Consultez également

`MovieClip._height`, `Video.width`

Video.smoothing

Disponibilité

Flash Player 6.

Usage

```
ma_video.smoothing
```

Description

Propriété : une valeur booléenne spécifiant si la vidéo doit être lissée (interpolée) lorsqu'elle est redimensionnée. Pour que le lissage fonctionne, le lecteur doit être en mode haute qualité. La valeur par défaut est `false` (aucun lissage).

Video.width

Disponibilité

Flash Player 6.

Usage

```
ma_video.width
```

Description

Propriété (lecture seule) : un entier spécifiant la largeur du flux vidéo, en pixels. Pour les flux vidéo en direct, cette valeur est identique à la propriété `Camera.width` de l'objet Camera qui capture le flux vidéo. Pour les fichiers FLV, cette valeur correspond à la largeur du fichier exporté en tant que fichier FLV.

Cette propriété peut vous servir, par exemple, à vous assurer que l'utilisateur regarde une vidéo dont la taille correspond à la taille de la vidéo au moment de sa capture, quelle que soit la taille réelle de l'objet Video sur la scène.

Exemple

Consultez les exemples pour `Video.height`.

void

Disponibilité

Flash Player 5.

Usage

```
void (expression)
```

Description

Opérateur : un opérateur unaire qui supprime la valeur *expression* et renvoie une valeur indéfinie. L'opérateur `void` est généralement utilisé dans les comparaisons avec l'opérateur == pour tester des valeurs indéfinies.

while

Disponibilité

Flash Player 4.

Usage

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

Paramètres

condition Expression réévaluée chaque fois que l'action `while` est exécutée.

instruction(s) Les instructions à exécuter si la condition est évaluée comme `true`.

Renvoie

Rien.

Description

Instruction : teste une expression et exécute une instruction ou une série d'instructions de manière répétée dans une boucle aussi longtemps que l'expression est `true`.

La *condition* est testée avant l'exécution du bloc d'instructions : si le test renvoie `true`, le bloc d'instructions est exécuté. Si la condition est `false`, le bloc d'instructions est ignoré et la première instruction se trouvant après le bloc d'instructions de l'action `while` est exécutée.

La boucle est souvent utilisée pour répéter une action tant qu'une variable de compteur est inférieure à une valeur spécifiée. Le compteur est incrémenté à la fin de chaque boucle, jusqu'à ce que la valeur spécifiée soit atteinte. A ce moment-là, la *condition* n'est plus `true` et la boucle se termine.

L'instruction `while` effectue la série d'étapes suivante. Chaque répétition des étapes 1–4 est appelée une ***itération*** de la boucle. La *condition* est retestée au début de chaque itération, comme dans les étapes suivantes :

1 L'expression *condition* est évaluée.

2 Si *condition* est `true` ou une valeur qui donne lieu par conversion à la valeur booléenne `true`, comme un nombre non nul, passez à l'étape 3.

Sinon, l'instruction `while` est terminée et l'exécution continue à partir de l'instruction qui suit la boucle `while`.

3 Exécutez le bloc d'instructions *instruction(s)*.

4 Passez à l'étape 1.

Consultez également

`do while`, `continue`, `for`, `for..in`

with

Disponibilité

Flash Player 5.

Usage

```
with (objet) {
  instruction(s);
}
```

Paramètres

objet Une occurrence d'un objet ActionScript ou d'un clip.

instruction(s) Une action ou un groupe d'actions renfermées dans des accolades.

Renvoie

Rien.

Description

Instruction : permet de spécifier un objet (un clip, par exemple) à l'aide du paramètre *object* et d'évaluer des expressions et des actions à l'intérieur de cet objet avec le paramètre *instruction(s)*. Cela vous évite d'avoir à rédiger de façon répétée le nom de l'objet ou le chemin de l'objet.

Le paramètre *object* devient le contexte dans lequel les propriétés, les variables et les fonctions de *instruction(s)* sont lues. Par exemple, si *objet* est `mon_array` et si deux des propriétés spécifiées sont `length` et `concat`, ces propriétés sont automatiquement lues comme `mon_array.length` et `mon_array.concat`. Dans un autre exemple, si *objet* est `state.california`, les actions ou instructions au sein de l'action `with` sont appelées depuis l'occurrence `california`.

Pour trouver la valeur d'un identifiant dans le paramètre *instruction(s)*, ActionScript démarre au début de la chaîne de portée spécifiée par *objet* et recherche l'identifiant à chaque niveau de la chaîne de portée, dans un ordre spécifique.

La chaîne de portée utilisée par l'action `with` pour traduire les identifiants commence par le premier élément de la liste suivante et continue jusqu'au dernier :

- L'objet spécifié dans le paramètre *objet* est l'action `with` la plus intérieure.
- L'objet spécifié dans le paramètre *objet* est l'action `with` la plus extérieure.
- L'objet Activation. Un objet temporaire créé automatiquement lorsqu'une fonction est appelée et qui contient les variables locales appelées dans la fonction.
- Le clip contenant le script en cours d'exécution
- L'objet Global (des objets intégrés tels que Math et String).

Pour définir une variable dans une action `with`, la variable doit avoir été déclarée à l'extérieur de l'action `with` ou vous devez entrer le chemin complet du scénario dans lequel vous souhaitez que la variable existe. Si vous définissez une variable dans une action `with` sans l'avoir déclarée, l'action `with` cherche la valeur selon la chaîne de plage. Si la variable n'existe pas déjà, la nouvelle valeur sera définie dans le scénario à partir duquel l'action `with` a été appelée.

Dans Flash 5 ou ultérieur, l'action `with` remplace l'action `tellTarget`. Nous vous recommandons d'utiliser `with` au lieu de `tellTarget`, étant donné qu'il s'agit d'une extension ActionScript standard de la norme ECMA-262. La principale différence entre les actions `with` et `tellTarget` réside dans le fait que `with` utilise comme paramètre une référence à un clip ou à un autre objet, tandis que `tellTarget` utilise une chaîne de chemin cible qui identifie un clip et ne peut pas être utilisée pour cibler des objets.

Exemple

L'exemple suivant définit les propriétés `_x` et `_y` de l'occurrence `autre_mc` et indique ensuite à `autre_mc` d'atteindre l'image 3 et de s'arrêter.

```
with (autre_mc) {
  _x = 50;
  _y = 100;
  gotoAndStop(3);
}
```

Le fragment de code suivant montre comment vous auriez pu rédiger le code précédent sans utiliser l'action `with`.

```
autre_mc._x = 50;
autre_mc._y = 100;
autre_mc.gotoAndStop(3);
```

Vous pourriez également rédiger ce code en utilisant l'action `tellTarget`. Cependant, si `autre_mc` n'était pas un clip, mais un objet, vous ne pouvez pas utiliser l'action `with`.

```
tellTarget ("autre_mc") {
  _x = 50;
  _y = 100;
  gotoAndStop(3);
}
```

L'action `with` est utile pour accéder simultanément à plusieurs éléments dans une liste de chaînes de portée. Dans l'exemple suivant, l'objet intégré `Math` est placé au début de la chaîne de plage. La définition de `Math` comme objet par défaut traduit les identifiants `cos`, `sin` et `PI` en `Math.cos`, `Math.sin` et `Math.PI`, respectivement. Les identifiants `a`, `x`, `y` et `r` ne sont pas des méthodes ou des propriétés de l'objet `Math`, mais puisqu'elles existent dans la plage d'activation d'objet de la fonction `polaire()`, elles se traduisent en variables locales correspondantes.

```
function polaire(r) {
  var a, x, y;
  with (Math) {
    a = PI * r * r;
    x = r * cos(PI);
    y = r * sin(PI/2);
}
trace("aire  = " +a);
trace("x = " + x);
trace("y = " + y);
}
```

Vous pouvez utiliser des actions `with` imbriquées pour accéder aux informations de plusieurs portées. Dans l'exemple suivant, l'occurrence `fresno` et l'occurrence `salinas` sont les enfants de l'occurrence `california`. L'instruction définit les valeurs `_alpha` de `fresno` et `salinas` sans changer la valeur `_alpha` de `california`.

```
with (california){
  with (fresno){
    _alpha = 20;
  }
  with (salinas){
    _alpha = 40;
  }
}
```

Consultez également

`tellTarget`

Classe XML

Disponibilité

Flash Player 5 (est devenu un objet natif dans Flash Player 6, améliorant ainsi les performances de manière significative).

Description

Utilisez les méthodes et les propriétés de la classe XML pour charger, analyser, envoyer, construire et manipuler des arborescences de documents XML.

Vous devez utiliser le constructeur `new XML()` pour créer un objet XML avant d'appeler les méthodes de la classe XML.

Méthodes de la classe XML

Méthode	Description
`XML.addRequestHeader()`	Ajoute ou modifie les en-têtes HTTP pour les opérations `POST`.
`XML.appendChild()`	Ajoute un nœud à la fin de la liste des enfants de l'objet spécifiée.
`XML.cloneNode()`	Clone le nœud spécifié et, optionnellement, clone récursivement tous les enfants.
`XML.createElement`	Crée un élément XML.
`XML.createTextNode`	Crée un nœud texte XML.
`XML.getBytesLoaded()`	Renvoie le nombre d'octets chargés pour le document XML spécifié.
`XML.getBytesTotal()`	Renvoie la taille du document XML, en octets.
`XML.hasChildNodes()`	Renvoie `true` si le nœud spécifié a des nœuds enfants ; sinon, renvoie `false`.
`XML.insertBefore`	Insère un nœud devant un nœud existant dans la liste des enfants du nœud spécifiée.
`XML.load()`	Charge un document (spécifié par l'objet XML) depuis une URL.

Méthode	Description
XML.parseXML	Analyse un document XML dans l'arborescence de l'objet XML spécifiée.
XML.removeNode	Supprime le nœud spécifié de son parent.
XML.send	Envoie l'objet XML spécifié à une URL.
XML.sendAndLoad	Envoie l'objet XML spécifié à une URL et charge la réponse du serveur dans un autre objet XML.
XML.toString	Convertit le nœud spécifié et ses enfants en texte XML.

Propriétés de la classe XML

Propriété	Description
XML.contentType	Indique le type MIME des données transmises au serveur.
XML.docTypeDecl	Définit et renvoie les informations relatives à la déclaration DOCTYPE d'un document XML.
XML.firstChild	Lecture seule ; fait référence au premier enfant de la liste pour le nœud spécifié.
XML.ignoreWhite	Lorsque true, les nœuds texte qui ne contiennent que des espaces vierges sont supprimés au cours de l'analyse.
XML.lastChild	Fait référence au dernier enfant de la liste pour le nœud spécifié.
XML.loaded	Lecture seule ; vérifie si l'objet XML spécifié a été chargé.
XML.nextSibling	Lecture seule ; fait référence au frère suivant dans la liste pour le nœud parent spécifié.
XML.nodeName	Le nom de nœud d'un objet XML.
XML.nodeType	Le type du nœud spécifié (élément XML ou nœud texte).
XML.nodeValue	Le texte du nœud spécifié si le nœud est un nœud texte.
XML.parentNode	Lecture seule ; fait référence au nœud parent du nœud spécifié.
XML.previousSibling	Lecture seule ; fait référence au frère précédent dans la liste pour le nœud parent spécifié.
XML.status	Un code d'état numérique indiquant le succès ou l'échec d'une opération d'analyse d'un document XML.
XML.xmlDecl	Spécifie les informations concernant la déclaration XML d'un document.

Collections de la classe XML

Méthode	Description
XML.attributes	Renvoie un tableau associatif contenant tous les attributs du nœud spécifié.
XML.childNodes	Lecture seule ; renvoie un tableau contenant les références aux nœuds enfants du nœud spécifié.

Gestionnaires d'événement de la classe XML

Gestionnaire d'événement	Description
`XML.onData`	Un gestionnaire d'événement invoqué lorsque du texte XML a été complètement téléchargé du serveur ou lorsqu'une erreur se produit au cours du téléchargement de texte XML depuis le serveur.
`XML.onLoad`	Un gestionnaire d'événements qui renvoie une valeur booléenne indiquant si l'objet XML a été chargé avec `XML.load()` ou `XML.sendAndLoad()`.

Constructeur de la classe XML

Disponibilité

Flash Player 5.

Usage

```
new XML([source])
```

Paramètres

source Texte XML analysé pour créer le nouvel objet XML.

Renvoie

Rien.

Description

Constructeur : crée un nouvel objet XML. Vous devez utiliser le constructeur pour créer un objet XML avant d'appeler les méthodes de la classe XML.

Remarque : Les méthodes `createElement()` et `createTextNode()` sont les méthodes « constructeur » pour la création d'éléments et de nœuds texte dans une arborescence de documents XML.

Exemple

Usage 1 : L'exemple suivant crée un nouvel objet XML vide.

```
mon_xml = new XML();
```

Usage 2 : L'exemple suivant crée construit un objet XML en analysant le texte XML spécifié dans le paramètre *source* et remplit l'objet XML nouvellement créé avec l'arborescence XML résultante.

```
unAutreXML = new XML("<état>Californie<ville>san francisco</ville></état>");
```

Consultez également

`XML.createElement`, `XML.createTextNode`

XML.addRequestHeader()

Disponibilité

Flash Player 6.

Usage

```
xml.addRequestHeader(nomDentete, valeurDentete)
xml.addRequestHeader(["nomDentete_1", "valeurDentete_1" ... "nomDentete_n",
  "valeurDentete_n"])
```

Paramètres

nomDentete Un nom d'en-tête de requête HTTP.

valeurDentete La valeur associée à *nomDentête*.

Renvoie

Rien.

Description

Méthode : ajoute ou modifie les en-têtes de requête HTTP (telles que `Content-type` ou `SOAPAction`) envoyés avec les actions `POST`. Dans la première utilisation, vous transmettez deux chaînes à la méthode : *nomDentete* et *valeurDentete*. Dans la seconde utilisation, vous transmettez un tableau de chaînes, alternant noms et valeurs d'en-têtes.

Si des appels multiples sont définis pour un seul et même nom d'en-tête, chaque valeur successive remplace la valeur définie dans le précédent appel.

Cette méthode ne permet pas d'ajouter ou de modifier les en-têtes standard HTTP suivants : `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`.

Exemple

Cet exemple montre l'ajout d'un en-tête HTTP personnalisé nommé `SOAPAction` et ayant la valeur `Foo` à un objet XML nommé `mon_xml`.

```
mon_xml.addRequestHeader("SOAPAction", "'Foo'");
```

L'exemple suivant crée un tableau nommé `en-têtes` contenant deux en-têtes secondaires et leurs valeurs associées. Le tableau est transmis en tant que paramètre à la méthode `addRequestHeader()`.

```
var en-têtes = ["Content-type", "texte/normal", "X-ClientAppVersion", "2.0"];
mon_xml.addRequestHeader(en-têtes);
```

Voir aussi

LoadVars.addRequestHeader()

XML.appendChild()

Disponibilité

Flash Player 5.

Usage

mon_xml.appendChild(*nœudEnfant*)

Paramètres

nœudEnfant Nœud enfant à ajouter à la liste des enfants de l'objet XML spécifiée.

Renvoie

Rien.

Description

Méthode : ajoute le nœud enfant spécifié à la fin de la liste des enfants de l'objet XML. Le nœud enfant ajouté est placé dans l'arborescence une fois supprimé de son nœud parent existant (s'il y en a un).

Exemple

L'exemple suivant clone le dernier nœud de `doc1` et l'ajoute à la fin de `doc2`.

```
doc1 = new XML(src1);
doc2 = new XML();
node = doc1.lastChild.cloneNode(true);
doc2.appendChild(node);
```

XML.attributes

Disponibilité

Flash Player 5.

Usage

mon_xml.attributes

Paramètres

Aucun.

Renvoie

Un tableau.

Description

Propriété : un tableau associé contenant tous les attributs de l'objet XML spécifié.

Exemple

L'exemple suivant enregistre les noms des attributs XML dans le panneau de sortie.

```
str = "<nom maBalise=\"Val\"> Elément </maBalise>";
doc = new XML(str);
y = doc.firstChild.attributes.name;
  trace (y);
doc.firstChild.attributes.order = "first";
z = doc.firstChild.attributes.order
  trace(z);
```

Le résultat suivant est affiché dans le panneau de sortie :

```
Val
first
```

XML.childNodes

Disponibilité

Flash Player 5.

Usage

mon_xml.childNodes

Paramètres

Aucun.

Renvoie

Un tableau.

Description

Propriété (lecture seule) : un tableau contenant les enfants de l'objet XML spécifié. Chaque élément du tableau est une référence à un objet XML qui représente un nœud enfant. Cette propriété est en lecture seule et ne peut pas être utilisée pour manipuler des nœuds enfants. Utilisez XML.appendChild(), XML.insertBefore et XML.removeNode pour manipuler les nœuds enfants.

Cette propriété n'est pas définie pour les nœuds de texte (nodeType == 3).

Consultez également

XML.nodeType

XML.cloneNode()

Disponibilité

Flash Player 5.

Usage

mon_xml.cloneNode(*récursivement*)

Paramètres

récursivement Une valeur booléenne spécifiant si les enfants de l'objet XML spécifié sont clonés récursivement.

Renvoie

Un nœud XML.

Description

Méthode : construit et renvoie un nouveau nœud XML possédant les mêmes type, nom, valeur et attributs que l'objet XML spécifié. Si *récursivement* est défini sur true, tous les nœuds enfants sont récursivement clonés, obtenant ainsi une copie exacte de l'arborescence de documents de l'objet original.

Le clone du nœud qui est renvoyé n'est plus associé à l'arborescence de l'élément cloné. Par conséquent, frèreSuivant, nœudParent et frèrePrécédent ont tous une valeur null. Si aucune copie de clip n'est réalisée, premierEnfant et dernierEnfant sont également null.

XML.contentType

Disponibilité

Flash Player 6.

Usage

mon_xml.contentType

Description

Propriété : le type MIME envoyé au serveur lorsque vous appelez la méthode XML.send XML.send ou XML.sendAndLoad XML.sendAndLoad. La valeur par défaut est *application/x-www-urlform-encoded*.

Consultez également

XML.send, XML.sendAndLoad

XML.createElement

Disponibilité

Flash Player 5.

Usage

mon_xml.createElement(*nom*)

Paramètres

nom Nom de balise de l'élément XML créé.

Renvoie

Un élément XML.

Description

Méthode : crée un élément XML portant le nom spécifié dans le paramètre. Le nouvel élément n'a, au début, ni parent, ni enfants, ni frères. La méthode renvoie une référence à l'objet XML nouvellement créé représentant l'élément. Cette méthode et createTextNode sont les méthodes constructeur permettant de créer les nœuds d'un objet XML.

XML.createTextNode

Disponibilité

Flash Player 5.

Usage

mon_xml.createTextNode(*texte*)

Paramètres

texte Texte utilisé pour créer le nouveau nœud texte.

Renvoie

Rien.

Description

Méthode : crée un nœud texte XML avec le texte spécifié. Le nouveau nœud n'a, au début, aucun parent et les nœuds texte ne peuvent pas avoir d'enfants ou frères. Cette méthode renvoie une référence à l'objet XML représentant le nouveau nœud texte. Cette méthode et `createElement()` sont les méthodes constructeur permettant de créer les nœuds d'un objet XML.

XML.docTypeDecl

Disponibilité

Flash Player 5.

Usage

mon_xml.XMLdocTypeDecl

Description

Propriété : spécifie les informations concernant la déclaration DOCTYPE du document XML. Une fois le texte XML analysé 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, <!DOCTYPE greeting SYSTEM>"Bonjour.dtd">. Cette propriété est définie avec une représentation chaîne de la déclaration DOCTYPE et non un objet nœud XML.

Le programme d'analyse XML d'ActionScript n'est pas un programme d'analyse de validation. La déclaration DOCTYPE est lue par le programme d'analyse et stockée dans la propriété docTypeDecl, mais aucune validation DTD n'est effectuée.

Si aucune déclaration DOCTYPE n'a été rencontrée pendant l'analyse, XML.docTypeDecl est défini sur undefined. XML.toString produit le contenu de XML.docTypeDecl immédiatement après la déclaration XML stockée dans XML.xmlDecl et avant tout autre texte dans l'objet XML. Si XML.docTypeDecl est indéfini, aucune déclaration DOCTYPE n'est sortie.

Exemple

L'exemple suivant utilise XML.docTypeDecl pour définir la déclaration DOCTYPE d'un objet XML :

```
mon_xml.docTypeDecl = "<!DOCTYPE greeting SYSTEM \"Bonjour.dtd\">";
```

Consultez également

XML.toString, XML.xmlDecl

XML.firstChild

Disponibilité

Flash Player 5.

Usage

mon_xml.firstChild

Description

Propriété (lecture seule) : évalue l'objet XML spécifié et fait référence au premier enfant de la liste des enfants du nœud parent. Cette propriété est null si le nœud n'a pas d'enfant. Cette propriété est undefined si le nœud est un nœud texte. Cette propriété est en lecture seule et ne peut pas être utilisée pour manipuler des nœuds enfants. Utilisez les méthodes appendChild(), insertBefore() et removeNode() pour manipuler les nœuds enfants.

Consultez également

XML.appendChild(), XML.insertBefore, XML.removeNode

XML.getBytesLoaded()

Disponibilité

Flash Player 6.

Usage

mon_xml.getBytesLoaded()

Paramètres

Aucun.

Renvoie

Un entier indiquant le nombre d'octets chargés.

Description

Méthode : renvoie le nombre d'octets chargés (en flux continu) pour le document XML. Vous pouvez comparer la valeur de `getBytesLoaded()` avec celle de `getBytesTotal()` pour déterminer le pourcentage d'un document XML qui a été chargé.

Consultez également

XML.getBytesTotal()

XML.getBytesTotal()

Disponibilité

Flash Player 6.

Usage

mon_xml.getBytesTotal()

Paramètres

Aucun.

Renvoie

Un entier.

Description

Méthode : renvoie la taille, en octets, du document XML spécifié.

Consultez également

XML.getBytesLoaded()

XML.hasChildNodes()

Disponibilité

Flash Player 5.

Usage

```
mon_xml.hasChildNodes()
```

Paramètres

Aucun.

Renvoie

Une valeur booléenne.

Description

Méthode : renvoie `true` si l'objet XML spécifié a des nœuds enfants ; sinon, renvoie `false`.

Exemple

L'exemple suivant utilise les informations de l'objet XML dans une fonction définie par l'utilisateur.

```
if (rootNode.hasChildNodes()) {
  maFonc (rootNode.firstChild);
}
```

XML.ignoreWhite

Disponibilité

Flash Player 5.

Usage

```
mon_xml.ignoreWhite = boolean
XML.prototype.ignoreWhite = boolean
```

Paramètres

boolean Une valeur booléenne (`true` ou `false`).

Description

Propriété : le paramètre par défaut est `false`. Lorsque `true`, les nœuds texte qui ne contiennent que des espaces vierges sont supprimés au cours de l'analyse. Les nœuds de texte qui contiennent un espace vierge avant ou après leur nom ne sont pas affectés.

Usage 1 : Vous pouvez définir la propriété `ignoreWhite` pour des objets XML individuels, comme dans l'exemple suivant :

```
mon_xml.ignoreWhite = true
```

Usage 2 : Vous pouvez définir la propriété `ignoreWhite` pour des objets XML individuels, comme dans l'exemple suivant :

```
XML.prototype.ignoreWhite = true
```

XML.insertBefore

Disponibilité

Flash Player 5.

Usage

```
mon_xml.insertBefore(nœudEnfant, nœudAvant)
```

Paramètres

nœudEnfant Le nœud à insérer.

nœudAvant Le nœud avant le point d'insertion de *nœudEnfant*.

Renvoie

Rien.

Description

Méthode : insère un nouveau nœud enfant dans la liste des enfants de l'objet XML, avant le nœud *nœudAvant*. Si le paramètre *nœudAvant* est indéfini ou `null`, le nœud est ajouté à l'aide de `appendChild()`. Si *nœudAvant* n'est pas un enfant de *mon_xml*, l'insertion échoue.

XML.lastChild

Disponibilité

Flash Player 5.

Usage

```
mon_xml.lastChild
```

Description

Propriété (lecture seule) : évalue l'objet XML et fait référence au dernier enfant de la liste des enfants du nœud parent. Cette méthode renvoie `null` si le nœud ne possède pas d'enfants. Cette propriété est en lecture seule et ne peut pas être utilisée pour manipuler des nœuds enfants. Utilisez les méthodes `appendChild()`, `insertBefore()` et `removeNode()` pour manipuler les nœuds enfants.

Consultez également

`XML.appendChild()`, `XML.insertBefore`, `XML.removeNode`

XML.load()

Disponibilité

Flash Player 5 ; comportement mis à jour dans Flash Player 7.

Usage

mon_xml.load(*url*)

Paramètres

url L'URL où se trouve le document XML à charger. Si le fichier SWF résultant de cet appel est ouvert dans un navigateur web, l'*url* doit être du même domaine que le fichier SWF ; pour plus de détails, consultez la description ci-dessous.

Renvoie

Rien.

Description

Méthode : charge un document XML depuis l'URL spécifiée et remplace le contenu de l'objet XML spécifié par les données XML téléchargées. L'URL est relative et est appelée via HTTP. Le processus de chargement est asynchrone et ne se termine pas immédiatement après l'exécution de la méthode `load()`.

Dans des fichiers SWF exécutés dans une version du lecteur antérieure à Flash Player 7, l'*url* doit être du même super-domaine que le fichier SWF résultant de cet appel. Par exemple, un fichier SWF à l'adresse www.Domaine.com peut charger des variables d'un fichier SWF à l'adresse store.Domaine.com car les deux fichiers sont du même super-domaine de Domaine.com.

Dans des fichiers SWF d'une quelconque version exécutée dans Flash Player 7 ou ultérieur, l'*url* doit être exactement du même domaine (consultez *Fonctions de sécurité de Flash Player*, page 199). Par exemple, un fichier SWF à l'adresse www.Domaine.com peut uniquement charger des variables de fichiers SWF également à l'adresse www.Domaine.com. Si vous souhaitez charger des variables d'un autre domaine, vous pouvez placer un *fichier de régulation inter-domaines* sur le serveur hébergeant le fichier SWF auquel vous accédez actuellement. Pour plus d'informations, consultez *A propos de l'autorisation de chargement de données inter-domaines*, page 201.

Lorsque la méthode `load()` est exécutée, la propriété `loaded` de l'objet XML est définie sur la valeur `false`. Lorsque le téléchargement des données XML est terminé, la propriété `loaded` est définie sur la valeur `true` et la méthode `onLoad()` est invoquée. Les données XML ne sont pas analysées avant le téléchargement complet. Si l'objet XML contenait auparavant des arborescences XML, elles sont supprimées.

Vous pouvez spécifier votre propre gestionnaire d'événements à la place de la méthode `onLoad()`.

Exemple

L'exemple suivant illustre l'utilisation de `XML.load()` :

```
doc = new XML();
doc.load ("leFichier.xml");
```

Consultez également

`XML.loaded`, `XML.onLoad`

XML.loaded

Disponibilité

Flash Player 5.

Usage

mon_xml.loaded

Description

Propriété (lecture seule) : détermine si le processus de chargement du document initié par l'appel XML.load() est achevé. Si le processus s'est achevé avec succès, la méthode renvoie true ; sinon, elle renvoie false.

Exemple

L'exemple suivant utilise XML.loaded dans un script simple.

```
if (doc.loaded) {
  gotoAndPlay(4);
}
```

XML.nextSibling

Disponibilité

Flash Player 5.

Usage

mon_xml.nextSibling

Description

Propriété (lecture seule) : évalue l'objet XML et fait référence au frère suivant dans la liste des enfants du nœud parent. Cette méthode renvoie null si le nœud ne possède pas un nœud frère suivant. Cette propriété est en lecture seule et ne peut pas être utilisée pour manipuler des nœuds enfants. Utilisez les méthodes appendChild(), insertBefore() et removeNode() pour manipuler les nœuds enfants.

Consultez également

XML.appendChild(), XML.insertBefore, XML.removeNode

XML.nodeName

Disponibilité

Flash Player 5.

Usage

mon_xml.nodeName

Description

Propriété : renvoie 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 représentant le nœud dans le fichier XML. Par exemple, TITLE est le nom de nœud d'une balise HTML TITLE. Si l'objet XML est un nœud texte (nodeType == 3), le nodeName est null.

Consultez également

XML.nodeType

XML.nodeType

Disponibilité

Flash Player 5.

Usage

mon_xml.nodeType

Description

Propriété (lecture seule) : prend ou renvoie une valeur nodeType, avec 1 pour un élément XML et 3 pour un nœud texte.

Consultez également

XML.nodeValue

XML.nodeValue

Disponibilité

Flash Player 5.

Usage

mon_xml.nodeValue

Description

Propriété : renvoie la 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 == 1), nodeValue est null et est en lecture seule.

Consultez également

XML.nodeType

XML.onData

Disponibilité

Flash Player 5

Usage

```
mon_xml.onData = function(src) {
  // vos instructions
}
```

Paramètres

src Données brutes, généralement au format XML, transmises par le serveur.

Renvoie

Rien.

Description

Gestionnaire d'événement : invoqué lorsque du texte XML a été complètement téléchargé du serveur ou lorsqu'une erreur se produit au cours de ce téléchargement. Ce gestionnaire est invoqué avant l'analyse du code XML et, par conséquent, peut être utilisé pour appeler une routine d'analyse personnalisée au lieu d'utiliser le programme d'analyse XML de Flash. La méthode `XML.onData` renvoie soit la valeur `undefined`, soit une chaîne contenant du texte XML téléchargé à partir du serveur. Si la valeur renvoyée est `undefined`, une erreur a eu lieu pendant le téléchargement du XML à partir du serveur.

Par défaut, la méthode `XML.onData` invoque `XML.onLoad`. Vous pouvez annuler la méthode `XML.onData` en utilisant votre propre comportement, `XML.onLoad` n'étant plus appelé si vous n'appelez pas votre propre implémentation de `XML.onData`.

Exemple

L'exemple suivant montre à quoi ressemble la méthode `onData` par défaut :

```
XML.prototype.onData = function (src) {
  if (src == undefined) {
    this.onLoad(false);
  } else {
    this.parseXML(src);
    this.loaded = true;
    this.onLoad(true);
  }
}
```

La méthode `XML.onData` peut être supplantée pour intercepter le texte XML sans l'analyser.

XML.onLoad

Disponibilité

Flash Player 5.

Usage

```
mon_xml.onLoad = function (succès) {
  //vos instructions ici
}
```

Paramètres

succès Une valeur booléenne indiquant si un objet XML a été chargé avec succès par une opération `XML.load()` ou `XML.sendAndLoad`.

Renvoie

Rien.

Description

Gestionnaire d'événements : appelé par Flash Player lorsqu'un document XML est reçu du serveur. Si le document XML est reçu avec succès, le paramètre *succès* est `true`. Si le document n'a pas été reçu, ou si une erreur est survenue lors de la réception de la réponse du serveur, le paramètre *success* est `false`. L'implémentation par défaut de cette méthode n'est pas active. Pour avoir la priorité sur l'implémentation par défaut, vous devez affecter une fonction contenant vos propres actions.

Exemple

L'exemple suivant crée un fichier SWF pour une application d'interface de commerce électronique basique. La méthode `sendAndLoad()` transmet un élément XML contenant le nom et le mot de passe de l'utilisateur et installe un gestionnaire `onLoad` pour traiter la réponse du serveur.

```
function monOnLoad(succès) {
  if (succès) {
    if (e.firstChild.nodeName == "REPONSELOGIN_xml" &&
        e.firstChild.attributes.status == "OK") {
      gotoAndPlay("connecté")
    } else {
      gotoAndStop("échecDuLogin")
    }
  } else {
    gotoAndStop("échecDeLaConnexion")
  }
}
var maRéponseLogin_xml = new XML();
maRéponseLogin_xml.onLoad = myOnLoad;
mon_xml.sendAndLoad("http://www.exemple.fr/login.cgi",
             maRéponseLogin_xml);
```

Consultez également

`function`, `XML.load()`, `XML.sendAndLoad`

XML.parentNode

Disponibilité

Flash Player 5.

Usage

mon_xml.parentNode

Description

Propriété (lecture seule) : fait référence au nœud parent de l'objet XML spécifié ou renvoie `null` si le nœud n'a pas de parent. Cette propriété est en lecture seule et ne peut pas être utilisée pour manipuler des nœuds enfants. Utilisez les méthodes `appendChild()`, `insertBefore()` et `removeNode()` pour manipuler les nœuds enfants.

XML.parseXML

Disponibilité

Flash Player 5.

Usage

mon_xml.parseXML(*source*)

Paramètres

source Texte XML à analyser et à transmettre à l'objet XML spécifié.

Renvoie

Rien.

Description

Méthode : analyse le texte XML spécifié dans le paramètre *source* et remplit l'objet XML spécifié avec l'arborescence XML résultante. Toutes les arborescences existantes dans l'objet XML sont supprimées.

XML.previousSibling

Disponibilité

Flash Player 5.

Usage

mon_xml.previousSibling

Description

Propriété (lecture seule) : renvoie une référence au frère précédent dans la liste des enfants du nœud parent. La valeur de cette propriété est `null` si le nœud n'a pas de nœud frère précédent. Cette propriété est en lecture seule et ne peut pas être utilisée pour manipuler des nœuds enfants. Utilisez `XML.appendChild()`, `XML.insertBefore` et `XML.removeNode` pour manipuler les nœuds enfants.

XML.removeNode

Disponibilité

Flash Player 5.

Usage

```
mon_xml.removeNode()
```

Paramètres

Aucun.

Renvoie

Rien.

Description

Méthode : supprime l'objet XML spécifié de son parent. Tous les descendants du nœud sont également supprimés.

XML.send

Disponibilité

Flash Player 5.

Usage

```
mon_xml.send(url, [fenêtre])
```

Paramètres

url URL de destination de l'objet XML spécifié.

fenêtre Fenêtre du navigateur affichant les données renvoyées par le serveur : `_self` spécifie l'image courante dans la fenêtre courante, `_blank` spécifie une nouvelle fenêtre, `_parent` spécifie le parent de l'image courante et `_top` spécifie l'image de premier niveau dans la fenêtre courante. Ce paramètre est facultatif ; si aucun paramètre *fenêtre* n'est spécifié, cela revient à spécifier `_self`.

Renvoie

Rien.

Description

Méthode : code l'objet XML spécifié en un document XML et l'envoie à l'URL spécifiée en utilisant la méthode `POST`.

XML.sendAndLoad

Disponibilité

Flash Player 5 ; comportement mis à jour dans Flash Player 7.

Usage

mon_xml.sendAndLoad(*url*,*objetXMLcible*)

Paramètres

url URL de destination de l'objet XML spécifié. Si le fichier SWF résultant de cet appel est ouvert dans un navigateur web, l'*url* doit être du même domaine que le fichier SWF ; pour plus de détails, consultez la description ci-dessous.

objetXMLcible Un objet XML créé avec la méthode constructeur XML et chargé de recevoir les informations renvoyées par le serveur.

Renvoie

Rien.

Description

Méthode : code l'objet XML spécifié en un document XML, l'envoie à l'URL spécifiée en utilisant la méthode POST, télécharge la réponse du serveur, puis la charge dans *objetXMLcible* spécifié dans les paramètres. La réponse du serveur est chargée de la même manière qu'avec la méthode load().

Dans des fichiers SWF exécutés dans une version du lecteur antérieure à Flash Player 7, l'*url* doit être du même super-domaine que le fichier SWF résultant de cet appel. Par exemple, un fichier SWF à l'adresse www.Domaine.com peut charger des variables d'un fichier SWF à l'adresse store.Domaine.com car les deux fichiers sont du même super-domaine de Domaine.com.

Dans des fichiers SWF d'une quelconque version exécutée dans Flash Player 7 ou ultérieur, l'*url* doit être exactement du même domaine (consultez *Fonctions de sécurité de Flash Player*, page 199). Par exemple, un fichier SWF à l'adresse www.Domaine.com peut uniquement charger des variables de fichiers SWF également à l'adresse www.Domaine.com. Si vous souhaitez charger des variables d'un autre domaine, vous pouvez placer un *fichier de régulation inter-domaines* sur le serveur hébergeant le fichier SWF auquel vous accédez actuellement. Pour plus d'informations, consultez *A propos de l'autorisation de chargement de données inter-domaines*, page 201.

Lorsque la méthode load() est exécutée, la propriété loaded de l'objet XML est définie sur la valeur false. Lorsque le téléchargement des données XML est terminé, la propriété loaded est définie sur la valeur true et la méthode onLoad() est invoquée. Les données XML ne sont pas analysées avant le téléchargement complet. Si l'objet XML contenait auparavant des arborescences XML, elles sont supprimées.

Consultez également

XML.load()

XML.status

Disponibilité

Flash Player 5.

Usage

mon_xml.status

Description

Propriété : définit et renvoie automatiquement une valeur numérique indiquant si un document XML a été analysé avec succès dans un objet XML. Les codes d'état numériques et leur description sont indiqués comme suit :

- 0 Aucune erreur ; analyse achevée avec succès.
- -2 Une section CDATA n'a pas été correctement achevée.
- -3 La déclaration XML n'est pas correctement terminée.
- -4 La déclaration DOCTYPE n'est pas correctement terminée.
- -5 Un commentaire n'est pas correctement terminé.
- -6 Un élément XML n'est pas correctement formé.
- -7 Mémoire disponible insuffisante.
- -8 Une valeur d'attribut n'est pas correctement terminée.
- -9 Une balise de début n'a pas de balise de fin.
- -10 Une balise de fin a été rencontrée sans balise de début correspondante.

XML.toString

Disponibilité

Flash Player 5.

Usage

mon_xml.toString()

Paramètres

Aucun.

Renvoie

Une chaîne.

Description

Méthode : évalue l'objet XML spécifié, construit une représentation textuelle de la structure XML en incluant le nœud, les enfants et les attributs et renvoie le résultat sous la forme d'une chaîne.

Pour les objets XML de haut niveau (ceux créés avec le constructeur), XML.toString produit la déclaration XML du document (stockée dans XML.xmlDecl), suivie de la déclaration DOCTYPE du document (stockée dans XML.docTypeDecl), suivie de la représentation textuelle de tous les nœuds XML de l'objet. La déclaration XML n'est pas produite si XML.xmlDecl est indéfini. La déclaration DOCTYPE n'est pas produite si XML.docTypeDecl est indéfini.

Exemple

Le code suivant est un exemple de XML.toString() qui envoie <h1>test</h1> au panneau de sortie.

```
node = new XML("<h1>test</h1>");
trace(node.toString());
```

Consultez également

XML.docTypeDecl, XML.xmlDecl

XML.xmlDecl

Disponibilité

Flash Player 5.

Usage

mon_xml.xmlDecl

Description

Propriété : spécifie les informations concernant la déclaration XML d'un document. Une fois le document XML analysé dans un objet XML, cette propriété est définie avec le texte de la déclaration XML du document. Cette propriété est définie avec une représentation chaîne de la déclaration XML et non un objet nœud XML. Si aucune déclaration XML n'a été rencontrée pendant l'analyse, la propriété est définie sur undefined.XML. La méthode XML.toString() produit le contenu de XML.xmlDecl avant tout autre texte dans l'objet XML. Si XML.xmlDecl contient le type undefined, aucune déclaration XML n'est produite.

Exemple

L'exemple suivant utilise XML.xmlDecl pour définir la déclaration XML pour un objet XML.

```
mon_xml.xmlDecl = "<?xml version=\"1.0\" ?>";
```

L'exemple suivant illustre une déclaration XML :

```
<?xml version="1.0" ?>
```

Consultez également

XML.docTypeDecl, XML.toString

Classe XMLNode

Disponibilité

Flash Player 5.

Description

La classe XMLnode supporte les propriétés, méthodes et collections suivantes. Pour plus d'informations sur leur utilisation, consultez les entrées correspondantes de la classe XML.

Propriété, méthode ou collection	Entrée de la classe XML correspondante
appendChild()	XML.appendChild()
attributes	XML.attributes
childNodes	XML.childNodes
cloneNode()	XML.cloneNode()
firstChild	XML.firstChild
hasChildNodes()	XML.hasChildNodes()
insertBefore()	XML.insertBefore
lastChild	XML.lastChild

Propriété, méthode ou collection	Entrée de la classe XML correspondante
nextSibling	XML.nextSibling
nodeName	XML.nodeName
nodeType	XML.nodeType
nodeValue	XML.nodeValue
parentNode	XML.parentNode
previousSibling	XML.previousSibling
removeNode()	XML.removeNode
toString()	XML.toString

Consultez également

Classe XML

Classe XMLSocket

Disponibilité

Flash Player 5.

Description

L'objet XMLSocket implémente les sockets clients qui permettent à l'ordinateur de lancer Flash Player pour communiquer avec un ordinateur serveur identifié par une adresse IP ou un nom de domaine. La classe XMLSocket s'utilise dans les applications client-serveur nécessitant un faible délai (par exemple, des systèmes de conversation en ligne en temps réel). Une solution de dialogue en ligne traditionnelle basée sur HTTP interroge fréquemment le serveur et télécharge les nouveaux messages à l'aide d'une requête HTTP. A l'inverse, une solution de dialogue en ligne XMLSocket maintient une connexion ouverte avec le serveur, ce qui permet au serveur d'envoyer immédiatement les messages entrants sans que le client en fasse la requête.

Pour utiliser la classe XMLSocket, le serveur doit exécuter un démon comprenant le protocole utilisé par la classe XMLSocket. Ce protocole est le suivant :

- Les messages XML sont envoyés sur une connexion socket TCP/IP continue en duplex intégral.
- Chaque message XML est un document XML complet, terminé par un octet zéro.
- Un nombre illimité de messages XML peut être envoyé et reçu sur une seule connexion XMLSocket.

Les restrictions suivantes s'appliquent à l'emplacement et la procédure de connexion d'un objet XMLSocket au serveur :

- La méthode `XMLSocket.connect()` permet uniquement la connexion aux numéros de port TCP supérieurs ou égaux à 1024. Dès lors, les démons communiquant avec l'objet XMLSocket doivent également être affectés à des numéros de port supérieurs ou égaux à 1024. Les numéros de port inférieurs à 1024 sont souvent utilisés par des services système (FTP, Telnet ou HTTP), interdisant ainsi l'accès à ces ports aux objets XMLSocket. La restriction du numéro de port limite les possibilités d'accès inappropriés et abusifs à ces ressources.

- La méthode `XMLSocket.connect()` ne peut se connecter qu'à des ordinateurs du même domaine que celui où réside le fichier SWF. Cette restriction ne s'applique pas aux fichiers SWF exécutés sur un disque local. Elle est identique aux règles de sécurité de `loadVariables()`, `XML.sendAndLoad()` et `XML.load()`. Pour vous connecter à un démon de serveur qui s'exécute dans un domaine autre que celui où réside le fichier SWF, vous pouvez créer un fichier de régulation de sécurité sur le serveur qui autorise l'accès à partir de domaines spécifiques. Pour plus d'informations sur la création d'un fichier de régulation pour les connexions XMLSocket consultez *A propos de l'autorisation de chargement de données inter-domaines*, page 201.

Le paramétrage d'un serveur pour communiquer avec l'objet XMLSocket peut être un véritable défi. Si votre application ne nécessite pas une interactivité en temps réel, utilisez l'action `loadVariables()` ou les connexions serveur XML basées sur HTTP de Flash (`XML.load()`, `XML.sendAndLoad()`, `XML.send()`), à la place de la classe XMLSocket.

Pour utiliser les méthodes de la classe XMLSocket, vous devez d'abord utiliser le constructeur, `new XMLSocket`, pour créer un objet XMLSocket.

Méthodes de la classe XMLSocket

Méthode	Description
`XMLSocket.close`	Ferme une connexion socket ouverte.
`XMLSocket.connect`	Etablit une connexion avec le serveur spécifié.
`XMLSocket.send`	Envoie un objet XML au serveur.

Gestionnaires d'événement de la classe XMLSocket

Gestionnaire d'événement	Description
`XMLSocket.onClose`	Un gestionnaire d'événements invoqué lorsqu'une connexion XMLSocket est fermée.
`XMLSocket.onConnect`	Un gestionnaire d'événements invoqué par Flash Player lorsqu'une requête de connexion initialisée via `XMLSocket.connect()` réussit ou échoue.
`XMLSocketXML.onData()`	Un gestionnaire d'événements invoqué lorsqu'un message XML est téléchargé depuis le serveur.
`XMLSocket.onXML`	Un gestionnaire d'événements invoqué lorsqu'un objet XML arrive depuis le serveur.

Constructeur de la classe XMLSocket

Disponibilité

Flash Player 5.

Usage

```
new XMLSocket()
```

Paramètres

Aucun.

Renvoie

Rien.

Description

Constructeur : crée un nouvel objet XMLSocket. L'objet XMLSocket n'est pas initialement connecté à un serveur. Vous devez appeler `XMLSocket.connect` pour connecter l'objet au serveur.

XMLSocket.close

Disponibilité

Flash Player 5.

Usage

```
monSocketXML.close()
```

Paramètres

Aucun.

Renvoie

Rien.

Description

Méthode : ferme la connexion spécifiée par l'objet XMLSocket.

Consultez également

`XMLSocket.connect`

XMLSocket.connect

Disponibilité

Flash Player 5 ; comportement mis à jour dans Flash Player 7.

Usage

monSocketXML.connect(*hôte, port*)

Paramètres

hôte Un nom de domaine DNS pleinement qualifié ou une adresse IP au format *aaa.bbb.ccc.ddd*. Vous pouvez également spécifier `null` pour établir une connexion au serveur hôte du fichier SWF. Si le fichier SWF résultant de cet appel est ouvert dans un navigateur web, l'*url* doit être du même domaine que le fichier SWF ; pour plus de détails, consultez la description ci-dessous.

port Le numéro du port TCP de l'hôte utilisé pour établir une connexion. Le numéro de port doit être supérieur ou égal à 1024.

Renvoie

Une valeur booléenne.

Description

Méthode : établit une connexion avec l'hôte Internet spécifié en utilisant le port TCP spécifié (supérieur ou égal à 1024) et renvoie `true` ou `false` selon que la connexion a été établie ou non avec succès. Contactez l'administrateur de votre réseau si vous ne connaissez pas le numéro du port de votre machine hôte Internet.

Si vous spécifiez `null` pour le paramètre *hôte*, l'hôte du fichier SWF qui appelle `XMLSocket.connect()` est contacté. Par exemple, si le fichier SWF a été téléchargé de l'adresse http://www.monSite.fr, la spécification de `null` pour le paramètre d'hôte revient à entrer l'adresse IP de www.monSite.fr.

Dans des fichiers SWF exécutés dans une version du lecteur antérieure à Flash Player 7, l'*url* doit être du même super-domaine que le fichier SWF résultant de cet appel. Par exemple, un fichier SWF à l'adresse www.Domaine.com peut charger des variables d'un fichier SWF à l'adresse store.Domaine.com car les deux fichiers sont du même super-domaine de Domaine.com.

Dans des fichiers SWF d'une quelconque version exécutée dans Flash Player 7 ou ultérieur, l'*url* doit être exactement du même domaine (consultez *Fonctions de sécurité de Flash Player*, page 199). Par exemple, un fichier SWF hébergé par www.DomaineX.com peut uniquement charger des variables à partir des fichiers SWF se trouvant sur www.DomainX.com. Pour charger des variables depuis un autre domaine, vous pouvez placer un *fichier de régulation inter-domaines* sur le serveur hébergeant le fichier SWF en cours d'accès (ce dernier doit être placé sur le serveur HTTP, sur le port 80, dans le même domaine que le serveur de socket). Pour plus d'informations, consultez *A propos de l'autorisation de chargement de données inter-domaines*, page 201.

Lorsque la méthode `load()` est exécutée, la propriété `loaded` de l'objet XML est définie sur la valeur `false`. Lorsque le téléchargement des données XML est terminé, la propriété `loaded` est définie sur la valeur `true` et la méthode `onLoad()` est invoquée. Les données XML ne sont pas analysées avant le téléchargement complet. Si l'objet XML contenait auparavant des arborescences XML, elles sont supprimées.

Si `XMLSocket.connect()` renvoie la valeur `true`, la première étape du processus de connexion est un succès ; la méthode `XMLSocket.onConnect` est invoquée plus tard pour déterminer si la connexion finale a réussi ou a échoué. Si `XMLSocket.connect()` renvoie `false`, la connexion n'a pas pu être établie.

Exemple

L'exemple suivant utilise `XMLSocket.connect()` pour établir une connexion à l'hôte du fichier SWF et utilise `trace` pour afficher la valeur renvoyée indiquant le succès ou l'échec de la connexion.

```
function maOnConnect(succès) {
  if (succès) {
    trace ("Connexion réussie !")
  } else {
    trace ("Echec de la connexion !")
  }
}
socket = new XMLSocket()
socket.onConnect = maOnConnect
if (!socket.connect(null, 2000)) {
  trace ("Echec de la connexion !")
}
```

Consultez également

`function`, `XMLSocket.onConnect`

XMLSocket.onClose

Disponibilité

Flash Player 5.

Usage

```
monSocketXML.onClose() = function() {
  // vos instructions
}
```

Paramètres

Aucun.

Renvoie

Rien.

Description

Gestionnaire d'événements : invoqué uniquement lorsqu'une connexion ouverte est fermée par le serveur. L'implémentation par défaut de cette méthode n'effectue aucune action. Pour avoir la priorité sur l'implémentation par défaut, vous devez affecter une fonction contenant vos propres actions.

Consultez également

`function`, `XMLSocket.onConnect`

XMLSocket.onConnect

Disponibilité

Flash Player 5.

Usage

```
monSocketXML.onConnect(succès)
  // vos instructions
}
```

Paramètres

succès Une valeur booléenne indiquant si une connexion socket a été établie avec succès (`true` ou `false`).

Renvoie

Rien.

Description

Gestionnaire d'événements : invoqué par Flash Player lorsqu'une requête de connexion initialisée via `XMLSocket.connect` réussit ou échoue. Si la connexion réussit, le paramètre *succès* est `true` ; sinon, le paramètre *succès* est `false`.

L'implémentation par défaut de cette méthode n'effectue aucune action. Pour avoir la priorité sur l'implémentation par défaut, vous devez affecter une fonction contenant vos propres actions.

Exemple

L'exemple suivant illustre le processus de spécification d'une fonction de remplacement pour la méthode `onConnect` dans une simple application de dialogue en ligne.

La fonction contrôle l'écran auquel les utilisateurs sont dirigés selon que la connexion a été établie avec succès ou non. Si la connexion est établie avec succès, les utilisateurs sont dirigés vers l'écran de dialogue principal de l'image nommée `débutDiscussion`. Si la connexion est un échec, les utilisateurs sont dirigés vers un écran d'informations de dépannage dans l'image nommée `connectionFailed`.

```
function maOnConnect(succès) {
  if (succès) {
    gotoAndPlay("débutDiscussion")
  } else {
    gotoAndStop("échecDeLaConnexion")
  }
}
```

Une fois l'objet XMLSocket créé à l'aide de la méthode constructeur, le script installe la méthode `onConnect` en utilisant l'opérateur d'affectation :

```
socket = new XMLSocket()
socket.onConnect = maOnConnect
```

Pour finir, la connexion est initiée. Si `connect()` renvoie `false`, le fichier SWF est envoyé directement vers l'image nommée `EchecDeLaConnexion` et `onConnect` n'est jamais invoqué. Si `connect()` renvoie `true`, le fichier SWF est envoyé vers une image appelée `attenteDeConnexion`, qui correspond à l'écran affichant un message d'attente. Le fichier SWF reste sur l'image `attenteDeConnexion` jusqu'à ce que le gestionnaire `onConnect` soit invoqué, ce qui arrive à un moment ultérieur, selon le délai d'attente du réseau.

```
if (!socket.connect(null, 2000)) {
  gotoAndStop("échecDeLaConnexion")
} else {
  gotoAndStop("attenteDeConnexion")
}
```

Consultez également

`function`, `XMLSocket.connect`

XMLSocketXML.onData()

Disponibilité

Flash Player 5.

Usage

```
XMLSocket.onData = function(src) {
  // vos instructions
}
```

Paramètres

`src` Une chaîne contenant les données envoyées par le serveur.

Renvoie

Rien.

Description

Gestionnaire d'événement : invoqué lorsqu'un message XML a été téléchargé du serveur, terminé par un octet zéro. Vous pouvez passer outre `XMLSocket.onData` pour intercepter les données envoyées par le serveur sans les analyser au format XML. Cette opération est utile si vous transférez des paquets de données formatés arbitrairement et que vous souhaitez manipuler les données directement lors de leur réception, sans que Flash Player ne les analyse au format XML.

Par défaut, la méthode `XMLSocket.onData` invoque la méthode `XMLSocket.onXML`. Si vous annulez `XMLSocket.onData` en utilisant votre propre comportement, `XMLSocket.onXML` n'est plus appelé si vous n'appelez pas votre propre implémentation de `XMLSocket.onData`.

```
XMLSocket.prototype.onData = function (src) {
  this.onXML(new XML(src));
}
```

Dans l'exemple ci-dessus, le paramètre *`src`* est une chaîne XML contenant du texte récupéré du serveur. La terminaison à octet zéro n'est pas incluse dans la chaîne.

XMLSocket.onXML

Disponibilité

Flash Player 5.

Usage

```
maXMLSocket.onXML(objet) = function() {
  // vos instructions
}
```

Paramètre

objet Un objet XML contenant un document XML analysé reçu d'un serveur.

Renvoie

Rien.

Description

Gestionnaire d'événements : invoqué par Flash Player lorsque l'objet XML spécifié contenant un document XML arrive sur une connexion XMLSocket ouverte. Une connexion XMLSocket peut être utilisée pour transférer un nombre illimité de documents XML entre le client et le serveur. Chaque document est terminé par un octet zéro. Lorsque Flash Player reçoit l'octet zéro, il analyse tout le code XML reçu depuis l'octet zéro précédent ou depuis que la connexion a été établie, s'il s'agit du premier message reçu. Chaque lot de code XML analysé est traité comme un seul document XML et transmis à la méthode `onXML`.

L'implémentation par défaut de cette méthode n'effectue aucune action. Pour annuler l'implémentation par défaut, vous devez affecter une fonction contenant vos propres actions.

Exemple

La fonction suivante prend la priorité sur l'implémentation par défaut de la méthode `onXML` dans une simple application de dialogue en ligne. La fonction `myOnXML` indique à l'application de dialogue en ligne de reconnaître un seul élément XML, `MESSAGE`, au format suivant :

```
<MESSAGE USER="John" TEXT="Bonjour, je m'appelle John !" />.
```

Le gestionnaire `onXML` doit d'abord être installé dans l'objet XMLSocket de la façon suivante :

```
socket.onXML = myOnXML;
```

La fonction `displayMessage()` est considérée comme étant définie par l'utilisateur et affichant le message reçu par l'utilisateur.

```
function monOnXML(doc) {
  var e = doc.firstChild;
  if (e != null && e.nodeName == "MESSAGE") {
    afficherMessage(e.attributes.user, e.attributes.text);
  }
}
```

Consultez également

`function`

XMLSocket.send

Disponibilité

Flash Player 5.

Usage

monSocketXML.send(*objet*)

Paramètres

objet Objet XML ou autres données à transmettre au serveur.

Renvoie

Rien.

Description

Méthode : convertit l'objet XML ou les données spécifiés dans l'objet *objet* en chaîne et la transmet au serveur, suivie d'un octet zéro. Si *objet* est un objet XML, la chaîne est la représentation textuelle XML de l'objet XML. L'opération d'envoi est asynchrone ; les résultats sont renvoyés immédiatement, mais les données peuvent être transmises plus tard. La méthode `XMLSocket.send()` ne renvoie pas de valeur indiquant si les données ont été transmises.

Si l'objet *monSocketXML* n'est pas connecté au serveur (avec `XMLSocket.connect`), l'opération `XMLSocket.send()` échoue.

Exemple

L'exemple suivant illustre comment spécifier un nom d'utilisateur et un mot de passe pour envoyer l'objet XML `mon_xml` au serveur :

```
var mon_xml = new XML();
var monLogin = mon_xml.createElement("login");
monLogin.attributes.nomDutilisateur = champDeTexteNomDutilisateur;
monLogin.attributes.motDePasse = champDeTexteNomMotDePasse;
mon_xml.appendChild(myLogin);
monXMLSocket.send(mon_xml);
```

Consultez également

`XMLSocket.connect`

ANNEXE A
Messages d'erreur

Macromedia Flash MX 2004 et Macromedia Flash MX Professionnel 2004 proposent des fonctions avancées de signalisation d'erreur de compilation si vous spécifiez ActionScript2.0 (valeur par défaut) lorsque vous publiez un fichier. Le tableau suivant contient la liste des messages d'erreur que le compilateur Flash est susceptible de générer.

Numéro de l'erreur	Texte du message
1093	Un nom de classe était attendu.
1094	Un nom de classe de base est attendu après le mot-clé 'extends'.
1095	Utilisation incorrecte d'un attribut de membre.
1096	Impossible d'utiliser un nom de membre plusieurs fois.
1097	Toutes les fonctions de membre doivent avoir des noms.
1099	Cette instruction est interdite dans une définition de classe.
1100	Une classe ou une interface de ce nom est déjà définie.
1101	Incompatibilité de types.
1102	Il n'existe aucune classe nommée '‹nomClasse›'.
1103	Il n'existe aucune propriété nommée '‹nomPropriete›'.
1104	Un appel de fonction pour un élément autre qu'une fonction a été tenté.
1105	Incompatibilité de types dans l'instruction d'affectation : [lhs-type] détecté au lieu de [rhs-type].
1106	Le membre est privé : accès impossible.
1107	Déclarations de variables interdites dans les interfaces.
1108	Déclarations d'événements interdites dans les interfaces.
1109	Déclarations de lecture/définitions interdites dans les interfaces.
1110	Membres privés interdits dans les interfaces.
1111	Corps de fonctions interdits dans les interfaces.
1112	Une classe ne peut pas s'étendre.

Numéro de l'erreur	Texte du message
1113	Une interface ne peut pas s'étendre.
1114	Aucune interface de ce nom n'est définie.
1115	Une classe ne peut pas étendre une interface.
1116	Une interface ne peut pas étendre une classe.
1117	Un nom d'interface est attendu après le mot-clé 'implements'.
1118	Une classe ne peut pas implémenter une autre classe, mais uniquement des interfaces.
1119	La classe doit implémenter la méthode 'nomMethode' depuis l'interface 'nomInterface'.
1120	La mise en œuvre d'une méthode d'interface doit être une méthode et non une propriété.
1121	Une classe ne peut pas étendre la même interface plusieurs fois.
1122	La mise en œuvre de la méthode d'interface ne correspond pas à sa définition.
1123	Cet élément est disponible uniquement dans ActionScript 1.0.
1124	Cet élément est disponible uniquement dans ActionScript 2.0.
1125	Membres statiques interdits dans les interfaces.
1126	L'expression renvoyée doit correspondre au type de renvoi de la fonction.
1127	Une instruction RETURN est requise dans cette fonction.
1128	Attribut utilisé en dehors de la classe.
1129	Une fonction dont le type de renvoi est Void ne renvoie aucune valeur.
1130	La clause 'extends' doit précéder la clause 'implements'.
1131	Un identifiant de type est attendu après ':'.
1132	Les interfaces doivent utiliser le mot-clé 'extends', et non pas 'implements'.
1133	Une classe ne peut pas étendre plus d'une classe.
1134	Une interface ne peut pas étendre plus d'une interface.
1135	La méthode nommée '<nomMethode>' n'existe pas.
1136	Instruction interdite dans une définition d'interface.
1137	Une fonction set requiert un seul paramètre.
1138	Une fonction get ne requiert aucun paramètre.
1139	Les classes ne peuvent être définies que dans des scripts de classe ActionScript 2.0 externes.
1140	Les scripts de classe ActionScript 2.0 peuvent définir uniquement des éléments de classe ou d'interface.

Numéro de l'erreur	Texte du message
1141	Le nom de cette classe, '<A.B.C>', n'est pas compatible avec le nom d'une autre classe chargée, '<A.B>'.
1142	Impossible de charger la classe '<nomClasse>'.
1143	Les interfaces ne peuvent être définies que dans des scripts de classe ActionScript 2.0 externes.
1144	Il est impossible d'accéder aux variables d'occurrence dans des fonctions statiques.
1145	Impossible d'imbriquer les définitions de classe et d'interface.
1146	La propriété référencée ne contient aucun attribut statique.
1147	L'appel de l'opérateur de superclasse ne correspond pas au superconstructeur.
1148	Seul l'attribut public est autorisé pour les méthodes d'interface.
1149	Impossible d'utiliser le mot-clé import en tant que directive.
1150	Vous devez exporter votre animation au format Flash 7 pour utiliser cette action.
1151	Vous devez exporter votre animation au format Flash 7 pour utiliser cette expression.
1152	Cette clause d'exception n'est pas placée correctement.
1153	Une classe doit comporter un seul constructeur.
1154	Un constructeur ne peut pas renvoyer de valeur.
1155	Un constructeur ne peut pas spécifier de type de renvoi.
1156	Une variable ne peut pas être de type Void.
1157	Un paramètre de fonction ne peut pas être de type Void.
1158	Accès aux membres statiques uniquement via les classes.
1159	Plusieurs interfaces mises en œuvre contiennent la même méthode avec des types différents.
1160	Une classe ou une interface de ce nom existe déjà.
1161	Impossible de supprimer les classes, les interfaces et les types intégrés.
1162	Il n'existe aucune classe de ce nom.
1163	Le mot-clé '<mot-cle>' est réservé à une utilisation avec ActionScript 2.0 et ne peut pas être utilisé ici.
1164	La définition de l'attribut personnalisé n'est pas terminée.
1165	Une seule classe ou interface peut être définie par fichier .as ActionScript 2.0.
1166	La classe en cours de compilation, '<A.b>', ne correspond pas à la classe importée, '<A.B>'.
1167	Vous devez indiquer un nom de classe.
1168	Le nom de classe que vous avez entré présente une erreur de syntaxe.

Numéro de l'erreur	Texte du message
1169	Le nom d'interface que vous avez entré présente une erreur de syntaxe.
1170	Le nom de classe de base que vous avez entré présente une erreur de syntaxe.
1171	Le nom d'interface de base que vous avez entré présente une erreur de syntaxe.
1172	Vous devez indiquer un nom d'interface.
1173	Vous devez indiquer un nom de classe ou d'interface.
1174	Le nom de classe ou d'interface que vous avez entré présente une erreur de syntaxe.
1175	'variable' n'est pas accessible dans cette étendue.
1176	Plusieurs occurrences de l'attribut 'get/set/private/public/static' ont été trouvées.
1177	Un attribut de classe a été utilisé de manière incorrecte.
1178	Les variables et les fonctions d'occurrence ne peuvent pas être utilisées pour l'initialisation de variables statiques.
1179	Des circularités d'exécution ont été détectées entre les classes suivantes : %1
1180	La version Flash Player ciblée ne prend pas en charge le débogage.
1181	La version Flash Player ciblée ne prend pas en charge l'événement releaseOutside.
1182	La version Flash Player ciblée ne prend pas en charge l'événement dragOver.
1183	La version Flash Player ciblée ne prend pas en charge l'événement dragOut.
1184	La version Flash Player ciblée ne prend pas en charge les actions de déplacement.
1185	La version Flash Player ciblée ne prend pas en charge l'action loadMovie.
1186	La version Flash Player ciblée ne prend pas en charge l'action getURL.
1187	La version Flash Player ciblée ne prend pas en charge l'action FSCommand.
1188	Les instructions d'importation ne sont pas autorisées dans les définitions de classe ou d'interface.
1189	La classe '‹A.B›' ne peut pas être importée, car son nom de feuille est déjà en cours de résolution dans la classe en cours de définition, '‹C.B›'.
1190	La classe '‹A.B›' ne peut pas être importée, car son nom de feuille est déjà en cours de résolution dans la classe importée '‹C.B›'.
1191	Les variables d'occurrence d'une classe peuvent être initialisées uniquement dans des expressions constantes de compilation.
1192	Les fonctions des membres de classe ne peuvent pas porter le même nom qu'une fonction de constructeur de superclasse.
1193	Le nom de la classe, '‹nomClasse›', est en conflit avec le nom d'une autre classe chargée.
1194	Le corps d'un constructeur doit d'abord contenir un appel au superconstructeur.
1195	L'identifiant '‹nomClasse›' n'est pas résolu dans un objet intégré '‹nomClasse›' lors de l'exécution.

Numéro de l'erreur	Texte du message
1196	La classe '‹A.B.nomClasse›' doit être définie dans un fichier dont le chemin d'accès relatif est ‹'A.B›'.
1197	Le caractère générique '*' est utilisé de manière incorrecte dans le nom de classe '‹nomClasse›'.
1198	La fonction de membre '‹nomClasse›' a une casse différente de celle du nom de la classe en cours de définition, '‹nomClasse›', et ne sera pas traitée en tant que constructeur de classe à l'exécution.
1199	Le seul type d'itérateur de boucle for-in accepté est le type Chaîne.
1200	Une fonction de définition ne peut pas renvoyer de valeur.
1201	Les seuls attributs autorisés pour les fonctions constructeur sont public et private.

ANNEXE B
Priorité et associativité des opérateurs

Ce tableau répertorie l'ensemble des opérateurs ActionScript et leur associativité et les classe par ordre de priorité décroissante.

Opérateur	Description	Associativité
	Priorité la plus élevée	
+	Unaire plus	Droite à gauche
-	Unaire moins	Droite à gauche
~	NOT au niveau du bit	Droite à gauche
!	NOT logique	Droite à gauche
not	NOT logique (style Flash 4)	Droite à gauche
++	Post-incrémentation	Gauche à droite
--	Post-décrémentation	Gauche à droite
( )	Appel de fonction	Gauche à droite
[ ]	Elément de tableau	Gauche à droite
.	Membre de structure	Gauche à droite
++	Pré-incrémentation	Droite à gauche
--	Pré-décrémentation	Droite à gauche
new	Affectation d'objet	Droite à gauche
delete	Désaffectation d'objet	Droite à gauche
typeof	Type d'objet	Droite à gauche
void	Renvoie une valeur non définie	Droite à gauche
*	Multiplier	Gauche à droite
/	Diviser	Gauche à droite
%	Modulo	Gauche à droite
+	Additionner	Gauche à droite

Opérateur	Description	Associativité
add	Concaténation de chaîne (auparavant &)	Gauche à droite
-	Soustraire	Gauche à droite
<<	Décalage gauche au niveau du bit	Gauche à droite
>>	Décalage droit au niveau du bit	Gauche à droite
>>>	Décalage droit au niveau du bit (non signé)	Gauche à droite
<	Inférieur à	Gauche à droite
<=	Inférieur ou égal à	Gauche à droite
>	Supérieur à	Gauche à droite
>=	Supérieur ou égal à	Gauche à droite
instanceof	Occurrence de	Gauche à droite
lt	Inférieur à (version chaîne)	Gauche à droite
le	Inférieur ou égal à (version chaîne)	Gauche à droite
gt	Supérieur à (version chaîne)	Gauche à droite
ge	Supérieur ou égal à (version chaîne)	Gauche à droite
==	Egal	Gauche à droite
!=	Différent	Gauche à droite
eq	Egal (version chaîne)	Gauche à droite
ne	Différent (version chaîne)	Gauche à droite
&	AND au niveau du bit	Gauche à droite
^	XOR au niveau du bit	Gauche à droite
|	OR au niveau du bit	Gauche à droite
&&	AND logique	Gauche à droite
and	AND logique (Flash 4)	Gauche à droite
||	OR logique	Gauche à droite
or	OR logique (Flash 4)	Gauche à droite
?:	Conditionnel	Droite à gauche
=	Affectation	Droite à gauche
*=, /=, %=, +=, -=, &=, |=, ^=, <<=, >>=, >>>=	Affectation de composant	Droite à gauche
,	Virgule	Gauche à droite
	Priorité la moins élevée	

ANNEXE C
Touches du clavier et valeurs de code correspondantes

Les tableaux suivants répertorient toutes les touches d'un clavier standard et les valeurs de code ASCII correspondantes utilisées pour identifier les touches dans ActionScript. Pour plus d'informations, consultez l'entrée *Classe Key* dans le Chapitre 12, *Dictionnaire ActionScript*, page 215.

Lettres A à Z et chiffres (clavier standard) de 0 à 9

Le tableau suivant répertorie toutes les touches d'un clavier standard pour les lettres de A à Z et les chiffres de 0 à 9, avec les valeurs de code ASCII correspondantes utilisées pour identifier les touches dans ActionScript.

Touche alphabétique ou numérique	Code de touche
A	65
B	66
C	67
D	68
E	69
F	70
G	71
H	72
I	73
J	74
K	75
L	76
M	77
N	78
O	79

Touche alphabétique ou numérique	Code de touche
P	80
Q	81
R	82
S	83
T	84
U	85
V	86
W	87
X	88
Y	89
Z	90
0	48
1	49
2	50
3	51
4	52
5	53
6	54
7	55
8	56
9	57

Touches du clavier numérique

Le tableau suivant répertorie les touches d'un clavier numérique et indique les valeurs de code ASCII correspondantes permettant d'identifier les touches dans ActionScript.

Touche du clavier numérique	Code de touche
0 (clavier numérique)	96
1 (clavier numérique)	97
2 (clavier numérique)	98
3 (clavier numérique)	99
4 (clavier numérique)	100
5 (clavier numérique)	101

Touche du clavier numérique	Code de touche
6 (clavier numérique)	102
7 (clavier numérique)	103
8 (clavier numérique)	104
9 (clavier numérique)	105
Multiplier	106
Additionner	107
Entrée	108
Soustraire	109
Décimal	110
Diviser	111

Touches de fonction

Le tableau suivant répertorie les touches de fonction d'un clavier standard et indique les valeurs de code ASCII correspondantes permettant d'identifier les touches dans ActionScript.

Touche de fonction	Code de touche
F1	112
F2	113
F3	114
F4	115
F5	116
F6	117
F7	118
F8	119
F9	120
F10	121
F11	122
F12	123
F13	124
F14	125
F15	126

Autres touches

Le tableau suivant répertorie les touches d'un clavier standard autres que les lettres, les nombres, les touches de fonction et les touches du clavier numérique, et indique les valeurs de code ASCII correspondantes permettant d'identifier les touches dans ActionScript.

Touche	Code de touche
Retour arrière	8
Tab	9
Effacer	12
Entrée	13
Maj	16
Contrôle	17
Alt	18
Verr Maj	20
Echap	27
Espace	32
Pg. Préc.	33
Pg. Suiv.	34
Fin	35
Origine	36
Flèche gauche	37
Flèche vers le haut	38
Flèche droite	39
Flèche vers le bas	40
Insertion	45
Suppr	46
Aide	47
Verr num	144
; :	186
= +	187
- _	189
/ ?	191
` ~	192
[{	219
\ |	220

Touche	Code de touche
]}	221
"'	222

ANNEXE D
Ecriture de scripts destinés à des versions antérieures de Flash Player

ActionScript a considérablement changé lors de la publication de Macromedia Flash MX 2004 et de Macromedia Flash MX Professionnel 2004. Lorsque vous créez un contenu pour Flash Player 7, vous exploitez pleinement la puissance d'ActionScript. Bien qu'il soit toujours possible de créer dans Flash MX 2004 du contenu destiné à des versions antérieures de Flash Player, vous ne pourrez pas utiliser tous les éléments d'ActionScript.

Ce chapitre donne des conseils pour écrire des scripts corrects syntaxiquement pour la version de Flash Player ciblée.

A propos du ciblage d'anciennes versions de Flash Player

Lorsque vous écrivez vos scripts, utilisez les informations de disponibilité des éléments du dictionnaire ActionScript (consultez le Chapitre 12, *Dictionnaire ActionScript*, page 215) pour déterminer si l'élément que vous voulez utiliser est pris en charge par la version de Flash Player que vous ciblez. Pour identifier les éléments disponibles, vous pouvez également afficher la boîte à outils Actions. Les éléments non pris en charge par votre version cible y sont surlignés en jaune.

Si vous créez du contenu pour Flash Player 6 ou Flash Player 7, utilisez de préférence ActionScript 2.0, qui propose un certain nombre de fonctions importantes qui ne sont pas disponibles dans ActionScript 1, notamment de meilleures fonctions de signalisation des erreurs de compilation et des capacités de programmation orientée objet plus robustes.

Pour connaître les différences entre la manière dont certaines fonctions sont implémentées lors de la publication de fichiers pour Flash Player 7 et la manière dont les fonctions sont implementées dans les fichiers publiés pour des versions antérieures, consultez *Portage de scripts existants sur Flash Player 7*, page 17.

Pour indiquer la version de Flash Player et d'ActionScript à utiliser lors de la publication d'un document, choisissez Fichier > Paramètres de publication, puis activez les options appropriées dans l'onglet Flash. Si vous devez cibler Flash Player 4, consultez la section suivante.

Utilisation de Flash MX 2004 pour créer du contenu destiné à Flash Player 4

Pour utiliser Flash MX 2004 pour créer du contenu destiné à Flash Player 4, activez l'option FlashPlayer 4 dans l'onglet Flash de la boîte de dialogue Paramètres de publication (Fichier > Paramètres de publication).

Le langage ActionScript de Flash Player 4 ne possède qu'un seul type de données de base, qui est utilisé pour les manipulations de chaînes et de nombres. Lorsque vous développez une application pour Flash Player 4, vous devez utiliser les opérateurs de chaîne déconseillés indiqués dans la catégorie Eléments déconseillés > Opérateurs de la boîte à outils Actions.

Lorsque vous effectuez une publication pour Flash Player 4, vous pouvez utiliser les fonctions suivantes de Flash MX 2004 :

- L'opérateur d'accès tableau et objet (`[]`).
- L'opérateur point (`.`).
- Les opérateurs logiques, d'affectation, de pré-incrémentation et de post-incrémentation/décrémentation.
- L'opérateur modulo (`%`), et toutes les méthodes et propriétés de la classe Math.

Les éléments de langage suivants ne sont pas pris en charge de façon native par Flash Player 4. Flash MX 2004 les exporte sous la forme d'une série d'approximations, qui crée des résultats moins précis numériquement. De plus, du fait de l'ajout de séries d'approximations dans le fichier SWF, ces éléments de langage occupent plus d'espace dans les fichiers SWF de Flash Player 4 qu'ils n'en occupent dans Flash Player 5 ou dans les fichiers SWF d'une version ultérieure.

- Les actions `for`, `while`, `do..while`, `break` et `continue`.
- Les actions `print()` et `printAsBitmap()`.
- L'action `switch`.

Pour plus d'informations, consultez *A propos du ciblage d'anciennes versions de Flash Player,* page 907.

Utilisation de Flash MX 2004 pour ouvrir des fichiers Flash 4

Le code ActionScript de Flash 4 ne possédait qu'un seul véritable type de données : les chaînes. Il utilisait différents types d'opérateurs dans les expressions pour indiquer si la valeur devait être traitée comme une chaîne ou comme un nombre. Dans les versions ultérieures de Flash, vous pouvez utiliser un ensemble d'opérateurs sur tous les types de données.

Si vous utilisez la version 5 (ou ultérieure) de Flash pour ouvrir un fichier créé dans Flash 4, Flash convertit automatiquement les expressions ActionScript afin de les rendre compatibles avec la nouvelle syntaxe. Dans votre code ActionScript, vous remarqueriez alors les conversions suivantes des type de données et d'opérateurs :

- L'opérateur = de Flash 4 était utilisé pour des égalités numériques. Dans la version 5 (et les versions ultérieures) de Flash, == est l'opérateur d'égalité et = est l'opérateur d'affectation. Tous les opérateurs = des fichiers Flash 4 sont automatiquement convertis en ==.

- Flash effectue automatiquement les conversions pour assurer le bon fonctionnement des opérateurs. L'introduction de plusieurs types de données donne une nouvelle signification aux opérateurs suivants :

  +, ==, !=, <>, <, >, >=, <=

  Dans ActionScript de Flash 4, ces opérateurs étaient toujours des opérateurs numériques. Dans la version 5 (et ultérieures) de Flash, ils se comportent différemment, selon le type de données des opérandes. Pour éviter toutes différences sémantiques dans les fichiers importés, la fonction `Number()` est insérée autour des opérandes de ces opérateurs. (Les nombres constants sont déjà des nombres évidents et ne sont donc pas encadrés de `Number()`).
- Dans Flash 4, la séquence d'échappement `\n` générait le caractère de retour chariot (ASCII 13). Dans la version 5 (et ultérieures) de Flash, pour respecter la norme ECMA-262, `\n` génère le caractère de changement de ligne (ASCII 10). Une séquence `\n` dans les fichiers FLA de Flash 4 est automatiquement convertie en `\r`.
- L'opérateur `&` de Flash 4 était utilisé pour les additions de chaînes. Dans la version 5 (et ultérieures) de Flash, `&` est l'opérateur AND au niveau du bit. L'opérateur d'addition de chaînes est maintenant appelé `add`. Tous les opérateurs `&` des fichiers de Flash 4 sont automatiquement convertis en opérateurs `add`.
- De nombreuses fonctions de Flash 4 ne nécessitaient pas l'usage de parenthèses de fermeture (par exemple, `Get Timer`, `Set Variable`, `Stop` et `Play`). Pour créer une syntaxe cohérente, la fonction `getTimer` et toutes les actions nécessitent maintenant l'usage de parenthèses de fermeture. Ces parenthèses sont automatiquement ajoutées lors de la conversion.
- Dans la version 5 (et ultérieures) de Flash, lorsque la fonction `getProperty` est exécutée sur un clip qui n'existe pas, elle renvoie la valeur `undefined` et non 0. L'instruction `undefined == 0` est `false` dans ActionScript après Flash 4 (dans Flash 4, `undefined == 1`). Pour résoudre ce problème lors de la conversion de fichiers Flash 4, insérez la fonction `Number()` dans les comparaisons d'égalité. Dans l'exemple suivant, `Number()` oblige à la conversion de `undefined` en 0 pour permettre la comparaison :

  ```
  getProperty("clip", _width) == 0
  Number(getProperty("clip", _width)) == Number(0)
  ```

Remarque : Si vous utilisez des mots-clés de Flash 5 ou d'une version ultérieure en tant que noms de variable dans votre code ActionScript Flash 4, la syntaxe renvoie une erreur lors de sa compilation dans Flash MX 2004. Pour résoudre ce problème, renommez toutes vos variables. Consultez *Mots-clés*, page 36 et *Affectation d'un nom à une variable*, page 44.

Utilisation de la syntaxe à barre oblique

La syntaxe à barre oblique était utilisée dans Flash 3 et 4 pour indiquer le chemin cible d'un clip ou d'une variable. Dans la syntaxe à barre oblique, les barres obliques sont utilisées à la place de points. Pour indiquer une variable, vous la précédez d'un deux-points :

```
monClip/clipEnfant:maVariable
```

Pour rédiger le même chemin cible en syntaxe à point (consultez *Syntaxe pointée*, page 33), qui est supportée dans Flash Player 5 (et versions ultérieures), utilisez le code suivant :

```
monClip.clipEnfant.maVariable
```

La syntaxe à barre oblique était le plus souvent utilisée avec l'action `tellTarget` dont l'utilisation n'est plus recommandée. L'action `with` est maintenant préférée à `tellTarget` pour des raisons de compatibilité avec la syntaxe à point. Pour plus d'informations, consultez *`tellTarget`* et *`with`* dans le Chapitre 12, *Dictionnaire ActionScript*, page 215.

ANNEXE E
Programmation orientée objet avec ActionScript 1

Les informations contenues dans cette annexe sont tirées de la documentation de Macromedia Flash MX et concernent l'utilisation du modèle d'objet ActionScript 1 permettant l'écriture de scripts. Elles sont ci-incluses pour les raisons suivantes :

- L'écriture de scripts orientés objet supportant Flash Player 5 requiert l'utilisation d'ActionScript 1.
- Si vous utilisez déjà ActionScript 1 pour l'écriture de scripts orientés objet, mais que vous n'êtes pas prêt à passer à ActionScript 2.0, cette annexe vous permettra de trouver les informations nécessaires à l'écriture de scripts avec ActionScript 1.

Si vous n'avez jamais utilisé ActionScript pour écrire des scripts orientés objet et que vous n'avez pas besoin de Flash Player 5, ne tenez pas compte des informations contenues dans cette annexe, car l'écriture de scripts orientés objet à l'aide d'ActionScript 1 n'est pas conseillée. Veuillez alors consulter Chapitre 9, *Création de classes avec ActionScript 2.0*, page 163 pour plus d'informations sur l'utilisation d'ActionScript 2.0.

Remarque : Certains exemples de cette annexe utilisent la méthode `Object.RegisterClass()`. Cette dernière n'est supportée que par Flash Player 6 et les versions ultérieures ; veuillez ne pas l'utiliser avec Flash Player 5.

A propos d'ActionScript 1

ActionScript est un langage de programmation orienté objet. La programmation orientée objet utilise des *objets*, ou structures de données, afin de regrouper les propriétés et les méthodes contrôlant le comportement ou l'apparence de l'objet. Les objets permettent d'organiser et de réutiliser le code. Après avoir défini un objet, il est possible de se référer à ce dernier grâce à son nom, sans avoir à le redéfinir à chaque utilisation.

Une *classe* est une catégorie générique d'objets. Une classe définit une série d'objets ayant des propriétés communes et pouvant être contrôlés de la même manière. Les propriétés sont des attributs définissant un objet, tels que la taille, la position, la couleur, la transparence, etc. Les propriétés sont définies par classe ; les valeurs des propriétés sont définies pour chaque objet d'une classe. Les méthodes sont des fonctions pouvant définir ou rechercher les propriétés d'un objet. Par exemple, vous pouvez définir une méthode permettant de calculer la taille d'un objet. Tout comme les propriétés, les méthodes sont définies pour chaque classe d'objets, puis appelées pour chaque objet d'une classe.

ActionScript inclut plusieurs classes intégrées, comme la classe MovieClip. Vous pouvez également créer des classes permettant de définir les catégories des objets de vos applications.

Les objets d'ActionScript peuvent être de simples containers de données ou peuvent être représentés graphiquement sur la scène sous forme de clips, boutons ou champs de texte. Tous les clips sont des occurrences de la classe intégrée MovieClip, et tous les boutons sont des occurrences de la classe intégrée Button. Chaque occurrence d'un clip contient toutes les propriétés (par exemple, `_height`, `_rotation`, `_totalframes`) et toutes les méthodes (par exemple, `gotoAndPlay`, `loadMovie`, `startDrag`) de la classe MovieClip.

Pour définir une classe, vous créez une fonction spéciale appelée *fonction constructeur*. Notez que les classes intégrées ont des fonctions constructeur intégrées. Par exemple, si vous souhaitez inclure dans votre application des informations sur un cycliste, vous pouvez créer une fonction constructeur, `cycliste`, avec les propriétés `temps` et `distance` et la méthode `vitesse()`, qui vous indique la vitesse à laquelle se déplace le cycliste :

```
function cycliste(t, d) {
  this.temps = t;
  this.distance = d;
  this.vitesse = function() {return this.temps / this.distance;};
}
```

Dans cet exemple, vous allez créer une fonction dont l'exécution requiert deux éléments d'information, ou « paramètres » : `t` et `d`. Lorsque vous appelez la fonction pour créer de nouvelles occurrences de l'objet, vous lui transférez les paramètres. Le code suivant crée des occurrences de l'objet `cycliste` appelées `emma` et `hamish`.

```
emma = new cycliste(30, 5);
hamish = new cycliste(40, 5);
```

Dans la création de scripts orientés objet, les classes peuvent échanger entre elles des propriétés et des méthodes selon un ordre spécifique, ce qui est appelé *héritage*. Vous pouvez utiliser l'héritage pour développer ou redéfinir les propriétés et les méthodes d'une classe. Une classe qui hérite d'une autre classe est appelée *sous-classe*. Une classe qui transmet des propriétés et des méthodes à une autre classe est appelée *super-classe*. Une classe peut être à la fois une sous-classe et une super-classe.

Un objet est un type de données complexe contenant aucune ou plusieurs propriétés et méthodes. Chaque propriété, tout comme une variable, possède un nom et une valeur. Les propriétés sont liées à l'objet et contiennent des valeurs qui peuvent être modifiées ou récupérées. Ces valeurs peuvent être de n'importe quel type de données : chaîne, nombre, booléen, objet, clip ou undefined (non défini). Les propriétés suivantes sont de différents types de données :

```
client.nom = "DuChmoque";
client.âge = 30;
client.adhérent = true;
client.compte.enregActuel = 000609;
client.nomDoccurrenceMC._visible = true;
```

La propriété d'un objet peut également être un objet. A la ligne 4 de l'exemple précédent, `compte` est une propriété de l'objet `client` et `enregActuel` est une propriété de l'objet `compte`. La propriété `enregActuel` est de type nombre.

Création d'un objet personnalisé dans ActionScript 1

Pour créer un objet personnalisé, définissez une fonction constructeur. Une fonction constructeur reçoit toujours le nom du type d'objet qu'elle sert à créer. Vous pouvez utiliser le mot-clé `this` dans le corps de la fonction constructeur afin de faire référence à l'objet créé par le constructeur (lorsque vous appelez une fonction constructeur, Flash lui transmet `this` en tant que paramètre masqué). Dans l'exemple suivant, la fonction constructeur crée un cercle avec la propriété `rayon` :

```
function Cercle(rayon) {
  this.rayon = rayon;
}
```

Après avoir défini la fonction constructeur, vous devez créer une occurrence de l'objet. Utilisez l'opérateur `new` devant le nom de la fonction constructeur, puis donnez un nom de variable à la nouvelle occurrence. Par exemple, le code suivant utilise l'opérateur `new` pour créer un objet Cercle d'un rayon de 5 et l'affecte à la variable `monCercle` :

```
monCercle = new Cercle(5);
```

Remarque : Un objet est du même domaine que la variable à laquelle il est affecté.

Affectation de méthodes à un objet personnalisé dans ActionScript 1

Vous pouvez définir les méthodes d'un objet à l'intérieur de la fonction constructeur de l'objet. Toutefois, cette technique n'est pas conseillée, car elle définit la méthode à chaque fois que vous utilisez la fonction constructeur, comme dans l'exemple suivant, qui crée les méthodes `aire()` et `diamètre()` :

```
function Cercle(rayon) {
  this.rayon = rayon;
  this.aire = Math.PI * rayon * rayon;
  this.diamètre = function() {return 2 * this.rayon;}
}
```

Chaque fonction constructeur a une propriété `prototype` qui est créée automatiquement en même temps que la fonction. La propriété `prototype` indique les valeurs de propriétés par défaut des objets créés avec cette fonction. Chaque nouvelle occurrence d'un objet possède une propriété `__proto__`, qui fait référence à la propriété `prototype` de la fonction constructeur qui a servi à la créer. Ainsi, si vous affectez des méthodes à une propriété `prototype` d'un objet, ces dernières sont disponibles pour toute occurrence nouvellement créée de cet objet. Mieux vaut affecter une méthode à la propriété `prototype` de la fonction constructeur, car elle existe à un endroit et est référencée par de nouvelles occurrences de l'objet (ou classe). Vous pouvez utiliser les propriétés `prototype` et `__proto__` pour étendre des objets, ce qui permet de réutiliser du code selon une méthode orientée objet. Pour plus d'informations, consultez *Création d'héritages dans ActionScript 1*, page 916.

La procédure suivante explique comment affecter une méthode `aire()` à un objet Cercle personnalisé.

Pour affecter une méthode à un objet personnalisé :

1 Définissez la fonction constructeur `Cercle()`, comme suit.

```
function Cercle(rayon) {
  this.rayon = rayon;
}
```

2 Définissez la méthode `aire()` de l'objet Cercle. La méthode `aire()` calcule l'aire du cercle. Vous pouvez utiliser un libellé de fonction afin de définir la méthode `aire()` et affecter la propriété `aire` à l'objet prototype du cercle comme suit :

```
Cercle.prototype.aire = function () {
     return Math.PI * this.rayon * this.rayon;
};
```

3 Créez une occurrence de l'objet Cercle comme suit :

```
var monCercle = new Cercle(4);
```

4 Appelez la méthode `aire()` du nouvel objet `monCercle`, comme suit :

```
var monAireCercle = monCercle.area();
```

ActionScript cherche l'objet `monCercle` pour la méthode `aire()`. Puisque l'objet n'a pas de méthode `aire()`, son objet prototype `Circle.prototype` est recherché pour la méthode `aire()`. ActionScript la trouve et l'appelle.

Définition des méthodes du gestionnaire d'événement dans ActionScript 1

Vous pouvez créer une classe ActionScript pour les clips et définir les méthodes du gestionnaire d'événement dans l'objet prototype de cette nouvelle classe. La définition des méthodes dans l'objet prototype permet à toutes les occurrences de ce symbole de répondre de la même manière à ces événements.

Vous pouvez également ajouter une action de gestionnaire d'événement `onClipEvent` ou `on()` à une occurrence individuelle afin de fournir des instructions uniques qui ne s'exécutent que lorsque l'événement de cette occurrence se produit. Les actions `onClipEvent()` et `on()` ne supplantent pas la méthode du gestionnaire d'événement ; ainsi, les deux événements exécutent leurs scripts. Toutefois, si vous définissez les méthodes de gestionnaire d'événement dans l'objet prototype et que vous définissez également une méthode de gestionnaire d'événement pour une occurrence précise, la définition de l'occurrence supplante celle du prototype.

Afin de définir une méthode de gestionnaire d'événement dans un objet prototype d'un objet :

1 Placez un symbole de clip avec l'identifiant de liaison `Lidentifiant` dans la bibliothèque.

2 Dans le panneau Actions (Fenêtre > Panneaux de développement > Actions), utilisez l'action `function` pour définir une nouvelle classe, comme indiqué ici :

```
// définir une classe
function maClasseDeClip(){}
```

Cette nouvelle classe sera affectée à toutes les occurrences du clip ajoutées à l'application par le scénario, ou qui y seront ajoutées via les méthodes `attachMovie()` ou `duplicateMovieClip()`. Si vous souhaitez que ces clips accèdent aux méthodes et propriétés de l'objet intégré MovieClip, vous devez faire en sorte que la nouvelle classe hérite de la classe MovieClip.

3 Entrez le code suivant :

```
// hériter de la classe MovieClip
maClasseDeClip.prototype = new MovieClip();
```

La classe `maClasseDeClip` héritera désormais de toutes les propriétés et méthodes de la classe MovieClip.

4 Entrez un code semblable au suivant pour définir les méthodes de gestionnaire d'événement de la nouvelle classe :

```
// définir des méthodes de gestionnaire d'événement pour la classe
  maClasseDeClip
maClasseDeClip.prototype.onLoad = function() {trace ("clip chargé");}
maClasseDeClip.prototype.onEnterFrame = function() {trace ("clip entré dans
  l'image");}
```

5 Sélectionnez Fenêtre > Bibliothèque pour ouvrir le panneau Bibliothèque s'il n'est pas déjà visible.

6 Sélectionnez les symboles que vous souhaitez associer à votre nouvelle classe, puis sélectionnez Liaison dans le menu contextuel situé dans la partie supérieure droite du panneau Bibliothèque.

7 Dans la boîte de dialogue Propriétés de liaison, activez l'option Exporter pour ActionScript.

8 Entrez un identifiant dans le champ Identifiant.

L'identifiant doit être identique pour tous les symboles que vous voulez associer à la nouvelle classe. Dans l'exemple `maClasseDeClip`, l'identifiant est `Lidentifiant`.

9 Entrez du code tel que le suivant dans la fenêtre de script :

```
// enregistrer la classe
Object.registerClass("Lidentifiant", maClasseDeClip);
_root.attachMovie("Lidentifiant","monNom",1);
```

Cela permet d'enregistrer le symbole dont l'identifiant du lien est `Lidentifiant` et dont la classe est `maClasseDeClip`. Toutes les occurrences de `maClasseDeClip` ont des méthodes de gestionnaire d'événement se comportant comme définies à l'étape 4. Elles se comportent également comme toutes les occurrences de la classe MovieClip, puisque vous avez fait en sorte que la nouvelle classe hérite de la classe MovieClip dans l'étape 3.

```
function maClasseDeClip(){}

maClasseDeClip.prototype = new MovieClip();
maClasseDeClip.prototype.onLoad = function(){
  trace("clip chargé");
}
maClasseDeClip.prototype.onPress = function(){
  trace("enfoncé");
}

maClasseDeClip.prototype.onEnterFrame = function(){
  trace("clip image entrée");
}

maClasseDeClip.prototype.maFonction = function(){
  trace("maFonction appelée");
}

Object.registerClass("monIDdeClip",maClasseDeClip);
_root.attachMovie("monIDdeClip","ablue2",3);
```

Création d'héritages dans ActionScript 1

L'héritage est un moyen d'organiser, d'étendre et de réutiliser une fonctionnalité. Les sous-classes héritent des propriétés et des méthodes des super-classes, auxquelles elles ajoutent des propriétés et méthodes spécialisées. Par exemple, dans le monde réel, Cycle pourrait être une super-classe dont VTT et Tricycle seraient des sous-classes. Ces deux sous-classes contiennent ou *héritent* des méthodes et propriétés de la super-classe (par exemple, `roues`). Chaque sous-classe possède également des propriétés et des méthodes spécifiques qui étendent la super-classe (par exemple, la sous-classe VTT pourrait comporter une propriété `pignons`). Vous pouvez utiliser les éléments `prototype` et `__proto__` pour créer un héritage dans ActionScript.

Toutes les fonctions constructeur ont une propriété `prototype` qui est créée automatiquement en même temps que la fonction. La propriété `prototype` indique les valeurs de propriétés par défaut des objets créés avec cette fonction. Vous pouvez utiliser la propriété `prototype` pour affecter des propriétés et méthodes à une classe. Pour plus d'informations, consultez *Affectation de méthodes à un objet personnalisé dans ActionScript 1*, page 913.

Toutes les occurrences d'une classe possèdent une propriété `__proto__` qui indique de quel objet elles sont héritières. Lorsque vous utilisez une fonction constructeur pour créer un objet, la propriété `__proto__` est configurée de manière à faire référence à la propriété `prototype` de sa fonction constructeur.

L'héritage fonctionne selon une hiérarchie précise. Quand vous appelez une propriété ou une méthode d'un objet, ActionScript recherche dans l'objet si l'élément existe. Si tel n'est pas le cas, ActionScript recherche dans la propriété `__proto__` de l'objet les informations (`myObject.__proto__`). Si la propriété n'est pas une propriété de l'objet `__proto__` de l'objet, ActionScript recherche alors dans `myObject.__proto__.__proto__` et ainsi de suite.

L'exemple suivant définit la fonction constructeur `Cycle()` :

```
function Cycle (longueur, couleur) {
  this.longueur = longueur;
  this.couleur = couleur;
}
```

Le code suivant ajoute la méthode `roule()` à la classe Cycle :

```
Cycle.prototype.roule = function() {this._x = _x + 20;};
```

Au lieu d'ajouter une méthode `roule()` aux classes VTT et Tricycle, vous pouvez créer la classe VTT et définir Cycle comme sa superclasse :

```
VTT.prototype = new Cycle();
```

Vous devez ensuite appeler la méthode `roule()` de VTT, comme ci-dessous :

```
VTT.roule();
```

Les clips n'héritent pas les uns des autres. Pour créer un héritage entre clips, vous pouvez utiliser la méthode `Object.registerClass()` afin d'affecter une classe autre que MovieClip à des clips. Consulter `Object.registerClass()` dans le Chapitre 12, *Dictionnaire ActionScript*, page 215.

Pour plus d'informations sur les héritages, consultez les entrées `Object.__proto__`, `#initclip`, `#endinitclip` et `super` dans le Chapitre 12, *Dictionnaire ActionScript*, page 215.

Ajout de propriétés de lecture/définition à des objets dans ActionScript 1

Vous pouvez créer des propriétés de lecture/définition d'un objet à l'aide de la méthode `Object.addProperty()`.

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.

Lorsque Flash lit une propriété de lecture/définition, il ouvre la fonction lecture et la valeur renvoyée par la fonction devient une valeur de `prop`. Lorsque Flash écrit une propriété de lecture/définition, il ouvre la fonction de définition et transmet la nouvelle valeur comme paramètre. Si une propriété portant le nom donné existe déjà, la nouvelle propriété la remplace.

Vous pouvez ajouter des propriété de lecture/définition à des objets prototypes. Si vous ajoutez une propriété de lecture/définition à un objet prototype, 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 endroit, 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.

Pour plus d'informations, consultez *`Object.addProperty()`* dans le Chapitre 12, *Dictionnaire ActionScript*, page 215.

La commande Déboguer > Lister les variables en mode test supporte les propriétés de lecture/définition ajoutés aux objets à l'aide de la méthode `Object.addProperty()`. Les propriétés ainsi ajoutées à un objet s'affichent à côté des autres propriétés de l'objet dans le panneau de sortie. Les propriétés de lecture/définition sont identifiées dans le panneau de sortie par le préfixe `[lecture/définition]`. Pour plus d'informations sur la commande des variables de liste, consultez *Utilisation du panneau de sortie*, page 82.

Utilisation des propriétés de l'objet Function dans ActionScript 1

Vous pouvez spécifier l'objet auquel une fonction est appliquée, ainsi que les valeurs de paramètre transmises à cette fonction à l'aide des méthodes `call()` et `apply()` de l'objet Function. Toutes les fonctions dans ActionScript sont représentées par un objet Function, de sorte que toutes les fonctions supportent les méthodes `call()` et `apply()`. Lorsque vous créez une classe personnalisée à l'aide de la fonction constructeur ou lorsque vous définissez les méthodes d'une classe personnalisée à l'aide d'une fonction, vous pouvez ouvrir les méthodes `call()` et `apply()` de la fonction.

Invocation d'une fonction à l'aide de la méthode Function.call dans ActionScript 1

La méthode `Function.call()` ouvre la fonction représentée par un objet Function.

Dans presque tous les cas, l'opérateur d'appel de fonction (`()`) peut être utilisé au lieu de la méthode `call()`. L'opérateur de la fonction call crée un code concis et lisible. La méthode `call()` est surtout utile lorsque le paramètre `this` de l'invocation de fonction doit être explicitement contrôlé. Normalement, si une fonction est invoquée en tant que méthode d'un objet, dans le corps de la fonction, `this` est défini sur `monObjet` comme suit :

```
monObjet.maMéthode(1, 2, 3);
```

Dans certains cas, vous aurez besoin que `this` pointe autre part, si par exemple une fonction doit être invoquée comme méthode d'objet alors qu'elle n'est pas stockée comme méthode de cet objet.

```
monObjet.maMéthode.call(monAutreObjet, 1, 2, 3);
```

Vous pouvez transmettre la valeur `null` pour le paramètre *`cetObjet`* pour invoquer une fonction en tant que fonction ordinaire et pas 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)
```

Pour plus d'informations, consultez *`Function.call`* dans le Chapitre 12, *Dictionnaire ActionScript*, page 215.

Pour invoquer une fonction à l'aide de la méthode Function.call :

- Utilisez la syntaxe suivante :

  `maFonction``.call(`*`cetObjet, paramètre1, ..., paramètreN`*`)`

  Cette méthode prend les paramètres suivants :

  - Le paramètre *`cetObjet`* spécifie la valeur `this` dans le corps de la fonction.
  - Les paramètres *`paramètre1...`*, *`paramètreN`* spécifient les paramètres devant être transmis à *`maFonction`*. Vous pouvez spécifier zéro ou plusieurs paramètres.

Spécification de l'objet auquel une fonction est appliquée à l'aide de Function.apply() dans ActionScript 1

La méthode `Function.apply()` spécifie la valeur `this` à utiliser dans toute fonction appelée par ActionScript. Cette méthode spécifie également les paramètres à transmettre à toute fonction appelée.

Les paramètres sont spécifiés sous forme d'objet Array. Ceci est souvent utile lorsque le nombre de paramètres à transmettre n'est pas connu avant l'exécution du script.

Pour plus d'informations, consultez *`Function.apply()`* dans le Chapitre 12, *Dictionnaire ActionScript*, page 215.

Pour spécifier l'objet auquel une fonction est appliquée à l'aide de Function.apply() :

- Utilisez la syntaxe suivante :

  maFonction.apply(*cetObjet*, *objetArguments*)

  Cette méthode prend les paramètres suivants :

  - Le paramètre *cetObjet* spécifie l'objet auquel *maFonction* s'applique.
  - Le paramètre *objetArguments* définit un tableau dont les éléments sont transmis dans *maFonction* en tant que paramètres.

INDEX

D

E

F

G

H

I

O

P

R

S

T

U

V

X