

# EXTENSION DE FLASH®

## Extension de Flash® pour Windows® et Macintosh

Si le présent guide est fourni avec un logiciel régi par un contrat d'utilisateur final, ce guide ainsi que le logiciel décrit sont fournis sous licence et peuvent être utilisés ou copiés uniquement selon les clauses et conditions de la licence. A moins d'une autorisation expresse accordée par cette licence, aucune partie du présent guide ne peut être reproduite, stockée dans un système d'interrogation ou transmise, sous quelque forme ou par quelque moyen que ce soit (électronique, mécanique, par enregistrement ou autre) sans l'autorisation écrite préalable d'Adobe Systems Incorporated. Veuillez noter que le contenu du présent guide est protégé par la loi sur les droits d'auteur, même s'il n'est pas distribué avec un logiciel régi par un contrat de licence utilisateur.

Les informations contenues dans le présent guide sont fournies à titre purement informatif ; elles sont susceptibles d'être modifiées sans préavis et ne doivent pas être interprétées comme étant un engagement de la part d'Adobe Systems Incorporated. Adobe Systems Incorporated n'accepte aucune responsabilité quant aux erreurs ou inexactitudes pouvant être contenues dans le présent guide.

Veuillez noter que les illustrations et images existantes que vous souhaitez éventuellement inclure dans votre projet sont susceptibles d'être protégées par les lois sur les droits d'auteur. L'inclusion non autorisée de tels éléments dans vos nouveaux travaux peut constituer une violation des droits du propriétaire. Veuillez vous assurer que vous obtenez toute autorisation nécessaire auprès du détenteur du copyright.

Toute référence à des noms de sociétés dans les modèles types n'est utilisée qu'à titre d'exemple et ne fait référence à aucune société réelle.

Adobe, le logo Adobe, ActionScript, Dreamweaver, Fireworks, Flash et FlashHelp sont des marques déposées ou des marques de commerce d'Adobe Systems Incorporated aux Etats-Unis et/ou dans d'autres pays.

Mac OS est une marque commerciale d'Apple Inc., déposée aux Etats-Unis et dans d'autres pays. Windows est une marque commerciale ou une marque déposée de Microsoft Corporation aux Etats-Unis et/ou dans d'autres pays. Toutes les autres marques citées appartiennent à leurs propriétaires respectifs.

Ce produit comprend un ou plusieurs logiciels développés par l'Apache Software Foundation (<http://www.apache.org/>). La technologie MPEG Layer-3 de compression audio est utilisée sous licence de Fraunhofer IIS et Thomson Multimedia (<http://www.iis.fhg.de/amm/>). L'utilisation de fichiers audio compressés en MP3 n'est pas autorisée dans le Logiciel pour diffuser du contenu en direct ou en temps réel. Si vous avez besoin d'un décodeur MP3 pour assurer des diffusions en direct ou en temps réel, la responsabilité de vous procurer la licence d'exploitation de la technologie MP3 vous incombe. Certaines parties de ce produit contiennent du code utilisé sous licence de Nellymoser ([www.nellymoser.com](http://www.nellymoser.com)). La vidéo de Flash CS3 est optimisée par la technologie vidéo On2 TrueMotion. © 1992-2005 On2 Technologies, Inc. Tous droits réservés. <http://www.on2.com>. Ce produit comprend un ou plusieurs logiciels développés par le groupe OpenSymphony (<http://www.opensymphony.com/>).



Technologie de compression et décompression vidéo Sorenson Spark™ utilisée sous licence de Sorenson Media, Inc.

Adobe Systems Incorporated, 345 Park Avenue, San Jose, Californie 95110, Etats-Unis.

A l'attention des utilisateurs du Gouvernement des Etats-Unis. Ce logiciel et sa documentation sont des « articles commerciaux », conformément à la définition de ce terme dans le document 48 C.F.R. §2.101, comprenant d'une part un « logiciel informatique commercial » et d'autre part une « documentation de logiciel informatique commercial », conformément à la définition de ces termes dans le document 48 C.F.R. §12.212 ou 48 C.F.R. §227.7202, si approprié. Conformément aux documents 48 C.F.R. §12.212 ou 48 C.F.R. §§227.7202-1 à 227.7202-4, si approprié, le logiciel informatique commercial et la documentation de logiciel informatique commercial sont accordés sous licence aux utilisateurs du Gouvernement des Etats-Unis (a) uniquement en tant que produits commerciaux et (b) uniquement avec les droits accordés à tous les autres utilisateurs selon les termes et conditions mentionnés dans le présent contrat. Droits non publiés réservés dans le cadre des lois sur les droits d'auteur en vigueur aux Etats-Unis. Adobe Systems Incorporated, 345 Park Avenue, San Jose, CA 95110-2704, Etats-Unis. A l'attention des utilisateurs du Gouvernement des Etats-Unis, Adobe s'engage à respecter la législation relative à l'égalité des chances y compris, le cas échéant, les dispositions du décret 11246, tel qu'amendé, à la section 402 de la loi sur l'assistance aux vétérans du Vietnam (Vietnam Era Veterans Readjustment Assistance Act) de 1974 (38 USC 4212), et à la section 503 de la loi sur la réadaptation (Rehabilitation Act) de 1973, telle qu'amendée, et la réglementation des articles 41 CFR, alinéas 60-1 à 60-60, 60-250 et 60-741. La clause relative à l'égalité des chances et les règlements énoncés dans la phrase précédente doivent être compris comme tels lorsqu'il y est fait référence.

# Table des matières

<b>Introduction</b>	<b>5</b>
Présentation de l'API JavaScript d'Adobe Flash	6
Nouveautés de l'API JavaScript	11
Modèle d'objet de document (DOM) de Flash	13
Exemples d'implémentation	19
 <b>Chapitre 1 : Fonctions et méthodes de haut niveau</b>	 <b>21</b>
 <b>Chapitre 2 : Objets</b>	 <b>37</b>
Objet actionsPanel	42
Objet BitmapInstance	50
Objet BitmapItem	55
Objet CompiledClipInstance	58
Objet compilerErrors	63
Objet ComponentInstance	65
Objet componentsPanel	66
Objet Contour	68
Objet Document	72
Objet drawingLayer	209
Objet Edge	217
Objet Effect	222
Objet Element	226
Objet Fill	244
Objet Filter	250
Objet flash (fl)	264
Objet FLfile	317
Objet folderItem	334
Objet fontItem	335
Objet Frame	336
Objet halfEdge	352
Objet Instance	358
Objet Item	360
Objet Layer	370
Objet library	376

Objet Math .....	393
Objet Matrix .....	396
Objet outputPanel .....	400
Objet Oval .....	403
Objet Parameter .....	406
Objet Path .....	412
Objet Project .....	419
Objet ProjectItem .....	428
Objet Rectangle .....	435
Objet Screen .....	439
Objet ScreenOutline .....	448
Objet Shape .....	460
Objet SoundItem .....	467
Objet Stroke .....	473
Objet SymbolInstance .....	489
Objet SymbolItem .....	505
Objet Text .....	511
Objet TextAttrs .....	532
Objet TextRun .....	542
Objet Timeline .....	544
Objet ToolObj .....	578
Objet Tools .....	589
Objet Vertex .....	597
Objet XMLUI .....	600
Objet VideoItem .....	609
 <b>Chapitre 3 : Extensibilité de niveau C .....</b>	 <b>611</b>
Intégration de fonctions en C .....	612
Types de données .....	619
API de niveau C .....	620

# Introduction

En tant qu'utilisateur de l'application Adobe® Flash® CS3 Professional, vous connaissez peut-être le code `ActionScript™` qui permet de créer des scripts se lançant au moment de l'exécution dans Adobe® Flash® Player. L'interface de programmation d'applications JavaScript (ou API JavaScript) de Flash décrite dans ce document constitue un outil de programmation complémentaire qui offre la possibilité de créer des scripts exploitables dans l'environnement de programmation.

Ce document décrit les objets, méthodes et propriétés disponibles dans l'API JavaScript. Il suppose la maîtrise des commandes de l'environnement de programmation. Pour toute question au sujet du rôle d'une commande spécifique, consultez d'autres documents dans l'aide de Flash, comme *Utilisation de Flash*.

Ce document suppose également que vous connaissez la syntaxe JavaScript ou ActionScript et les concepts de base de la programmation (fonctions, paramètres et types de données).

Cette section contient les rubriques suivantes :

<a href="#">Présentation de l'API JavaScript d'Adobe Flash .....</a>	<a href="#">6</a>
<a href="#">Nouveautés de l'API JavaScript .....</a>	<a href="#">11</a>
<a href="#">Modèle d'objet de document (DOM) de Flash. ....</a>	<a href="#">13</a>
<a href="#">Exemples d'implémentation. ....</a>	<a href="#">19</a>

# Présentation de l'API JavaScript d'Adobe Flash

L'API Flash JavaScript permet d'écrire des scripts destinés à effectuer plusieurs actions dans l'environnement de développement Flash (pendant que le programme Flash est en cours d'exécution). Cette fonctionnalité est différente du langage ActionScript qui permet d'écrire des scripts destinés à effectuer des actions dans l'environnement Flash Player (pendant la lecture d'un fichier SWF). Elle diffère également des commandes JavaScript utilisables dans les pages qui s'affichent dans un navigateur Web.

Grâce à l'utilisation de l'API JavaScript, vous écrivez des scripts d'application Flash qui permettent de simplifier le processus de programmation. Par exemple, vous pouvez écrire des scripts pour automatiser des tâches répétitives, ajouter des outils personnalisés au panneau Outils ou ajouter des effets au scénario.

L'API JavaScript de Flash a été conçue sur le même modèle que les API JavaScript d'Adobe® Dreamweaver® et d'Adobe® Fireworks® (elles-mêmes basées sur l'API JavaScript de Netscape). L'API JavaScript de Flash repose sur un modèle d'objet de document (DOM, Document Object Model) qui permet d'accéder aux documents Flash à l'aide d'objets créés en JavaScript. Elle comporte tous les éléments de l'API JavaScript de Netscape, ainsi que le DOM de Flash. Les objets qui sont ajoutés à l'aide de ce dernier, ainsi que leurs méthodes et propriétés, sont décrits dans le présent document. Vous pouvez utiliser tous les éléments du langage JavaScript natif dans un script Flash, mais seuls les éléments dont l'utilisation est logique dans le contexte d'un document Flash prendront effet.

L'API JavaScript contient également certaines méthodes qui permettent d'implémenter des extensions par une combinaison de code JavaScript et C. Pour plus d'informations, consultez la section [Chapitre 3, « Extensibilité de niveau C »](#), à la page 611.

L'interpréteur JavaScript de Flash est la version 1.5 du moteur SpiderMonkey de Mozilla qui est disponible sur le Web à l'adresse suivante [www.mozilla.org/js/spidermonkey/](http://www.mozilla.org/js/spidermonkey/). SpiderMonkey, l'une des deux implémentations de référence du langage JavaScript développé par Mozilla.org, est intégré au navigateur Mozilla.

SpiderMonkey implémente le langage JavaScript de base tel qu'il est défini par la spécification ECMAScript (ECMA-262) édition 3 et est conforme à tous les niveaux à celle-ci. Seuls les objets hôtes spécifiques au navigateur (qui ne figurent pas dans la spécification ECMA-262) ne sont pas pris en charge. De même, de nombreux guides de référence JavaScript établissent une distinction entre le langage JavaScript de base et le langage JavaScript côté client (associé au navigateur). Seul le langage JavaScript de base s'applique à l'interpréteur Flash JavaScript.

## Création de fichiers JSFL

Pour créer et modifier des fichiers JavaScript Flash, ou fichiers JSFL, utilisez Adobe Flash CS3 Professional ou un éditeur de texte de votre choix. Ces fichiers auront par défaut l'extension .jsfl si vous utilisez Flash.

Il est également possible de créer un fichier JSFL en sélectionnant des commandes dans le panneau Historique, puis en cliquant sur le bouton Enregistrer du panneau Historique ou en choisissant Enregistrer comme commande à partir du menu du panneau. Le fichier de commandes est alors enregistré dans le dossier Commands avec l'extension JSFL (voir « [Enregistrement de fichiers JSFL](#) », à la page 7). Vous pouvez ensuite l'ouvrir et le modifier comme tout autre fichier script.

Le panneau Historique fournit lui aussi d'autres options utiles. Vous pouvez copier les commandes sélectionnées dans le presse-papiers et visualiser les commandes JavaScript qui sont générées pendant votre travail dans Flash.

### **Pour copier des commandes dans le presse-papiers à partir du panneau Historique :**

1. Sélectionnez une ou plusieurs commandes dans le panneau Historique.
2. Utilisez l'une des méthodes suivantes :
  - Cliquez sur le bouton Copier.
  - Dans le menu du panneau, sélectionnez Copier ces étapes.

### **Pour visualiser les commandes JavaScript dans le panneau Historique :**

- Dans le menu du panneau, choisissez Affichage > JavaScript dans le panneau.

## Enregistrement de fichiers JSFL

Vous pouvez rendre les scripts JSFL disponibles dans l'environnement de programmation Flash en les stockant dans un ou plusieurs sous-dossiers du dossier Configuration. Par défaut, le dossier Configuration se trouve à l'emplacement suivant :

- Windows® 2000 ou Windows® XP :  
*lecteur d'amorçage*\Documents and Settings\*utilisateur*\Local Settings\Application Data\Adobe\Flash CS3\langue\Configuration\  
*lecteur d'amorçage*\Documents and Settings\*utilisateur*\Local Settings\Application Data\Adobe\Flash CS3\langue\Configuration\
- Mac OS® X :  
Disque dur Macintosh/Utilisateurs/*utilisateur*/Library/Application Support/Adobe/Flash CS3/*langue*/Configuration/

Pour déterminer l'emplacement du dossier Configuration, utilisez `fl.configDirectory` ou `fl.configURI` comme dans les exemples suivants :

```
// stocker le répertoire dans une variable
var configDir = fl.configDirectory;
//afficher le répertoire dans le panneau Sortie
fl.trace(fl.configDirectory);
```

Dans le dossier Configuration, les dossiers suivants peuvent contenir des scripts qui sont accessibles depuis l'environnement de programmation : Behaviors (pour prendre en charge l'interface utilisateur des comportements), Commands (pour les scripts qui figurent dans le menu Commandes), Effects (pour les effets de scénario), JavaScript (pour les scripts utilisés par l'Assistant de script), Tools (pour les outils extensibles du panneau Outils) et WindowSWF (pour les panneaux qui s'affichent dans le menu Fenêtres). Ce document va étudier les scripts utilisés pour les commandes, les effets et les outils.

Si vous modifiez un script dans le dossier Commands, il est immédiatement disponible dans Flash. Si vous modifiez le script d'un effet ou d'un outil extensible, vous devez fermer Flash et le redémarrer, ou employer la commande `fl.reloadEffects()` ou `fl.reloadTools()`. Toutefois, si vous avez utilisé un script pour ajouter un outil extensible au panneau Outils puis que vous modifiez ce script, vous devez supprimer l'outil du panneau Outils puis l'ajouter de nouveau, ou fermer Flash et le redémarrer, pour que l'outil modifié soit disponible.

Pour être accessibles à partir de l'environnement de programmation, les fichiers de commandes, d'effets et d'outils peuvent être stockés à trois endroits différents.

- Pour les scripts qui prendront la forme d'éléments du menu Commandes, enregistrez les fichiers JSFL dans le dossier Commands qui se trouve à l'emplacement suivant :
  - Windows 2000 ou Windows XP :  
*lecteur d'amorçage*\Documents and Settings\utilisateur\Local Settings\Application Data\Adobe\Flash CS3\langue\Configuration\Commands
  - Mac OS X :  
Disque dur Macintosh/Utilisateurs/*utilisateur*/Library/Application Support/Adobe/Flash CS3/*langue*/Configuration/Commands
- Pour les scripts qui prendront la forme d'outils extensibles dans le panneau Outils, enregistrez les fichiers JSFL dans le dossier Tools qui se trouve à l'emplacement suivant :
  - Windows 2000 ou Windows XP :  
*lecteur d'amorçage*\Documents and Settings\utilisateur\Local Settings\Application Data\Adobe\Flash CS3\langue\Configuration\Tools
  - Mac OS X :  
Disque dur Macintosh/Utilisateurs/*utilisateur*/Library/Application Support/Adobe/Flash CS3/*langue*/Configuration/Tools



- Pour les scripts qui prendront la forme d'effets de scénario dans le panneau Effets, enregistrez les fichiers JSFL dans le dossier Effects, qui réside à l'emplacement suivant :
  - Windows 2000 ou Windows XP :  
*lecteur d'amorçage*\Documents and Settings\*utilisateur*\Local Settings\Application Data\Adobe\Flash CS3\*langue*\Configuration\Effects
  - Mac OS X :  
 Disque dur Macintosh/Utilisateurs/*utilisateur*/Library/Application Support/Adobe/Flash CS3/*langue*/Configuration/Effects

Si d'autres fichiers doivent accompagner un fichier JSFL, par exemple des fichiers XML, ils doivent être enregistrés dans le même répertoire que ce fichier JSFL.

## Exécution de fichiers JSFL

Il existe plusieurs manières d'exécuter des fichiers JSFL. Les plus courantes sont développées dans la section ci-dessous.

**Pour exécuter un script qui réside dans le dossier Commands, effectuez l'une des actions suivantes :**

- Choisissez Commandes > *nom du script*.
- Utilisez un raccourci clavier que vous aurez attribué au script. Pour attribuer un raccourci clavier, utilisez Edition > Raccourcis clavier puis choisissez Commandes du menu dessin dans la liste déroulante Commandes. Développez le noeud Commandes de l'arborescence afin d'afficher la liste des scripts disponibles.

**Pour exécuter un script de commande qui ne réside pas dans le dossier Commands, effectuez l'une des actions suivantes :**

- Dans l'environnement de programmation, choisissez Commandes > Exécuter une commande, puis sélectionnez le script à exécuter.
- Depuis un script, utilisez la commande `fl.runScript()`.
- Dans le système de fichiers, double-cliquez sur le fichier de script.

**Pour ajouter dans le panneau Outils un élément mis en oeuvre dans un fichier JSFL :**

1. Copiez le fichier JSFL de cet outil (ainsi que les éventuels fichiers associés) dans le dossier Tools (consultez la section « [Enregistrement de fichiers JSFL](#) », à la page 7).
2. Choisissez Edition > Personnaliser le panneau Outils (Windows) ou Flash > Personnaliser le panneau Outils (Macintosh).
3. Ajoutez le nouvel outil à la liste des outils disponibles.

#### 4. Cliquez sur OK.

Vous pouvez ajouter des commandes individuelles de l'API JavaScript dans un fichier ActionScript à l'aide de la fonction `MMExecute()` qui est décrite dans le *Guide de référence du langage ActionScript 3.0*. Toutefois, la fonction `MMExecute()` n'est exécutée que dans le contexte d'un élément d'interface utilisateur personnalisé, par exemple comme composant de l'inspecteur Propriétés, ou dans un panneau SWF au sein de l'environnement de création. Même si elles sont appelées en ActionScript, les commandes API JavaScript sont inopérantes dans Flash Player ou hors de l'environnement de création.

#### **Pour lancer une commande à partir d'un script ActionScript :**

- Employez la syntaxe suivante (vous pouvez concaténer plusieurs commandes au sein d'une chaîne) :

```
MMExecute(Javascript command string);
```

Vous pouvez également exécuter un script à partir de la ligne de commande.

#### **Pour exécuter un script à partir de la ligne de commande de Windows :**

- Utilisez la syntaxe suivante (en ajoutant les informations requises au sujet du chemin d'accès) :

```
"flash.exe" myTestFile.jsfl
```

#### **Pour exécuter un script à partir de l'application « Terminal » sur Macintosh :**

- Utilisez la syntaxe suivante (en ajoutant les informations requises concernant le chemin d'accès) :

```
osascript -e 'tell application "flash" to open alias "Mac OS  
X:Users:user:myTestFile.jsfl" '
```

La commande `osascript` peut elle aussi exécuter du code AppleScript dans un fichier. Par exemple, vous pouvez placer le texte suivant dans un fichier nommé `myScript` :

```
tell application "flash"  
  open alias "Mac OS X:Users:user:myTestFile.jsfl"  
end tell
```

Ensuite, pour lancer le script, employez la commande suivante :

```
osascript myScript
```

# Nouveautés de l'API JavaScript

Dans Flash CS3, plusieurs objets ont été ajoutés. Par ailleurs, certains objets existants possèdent désormais des méthodes ou propriétés supplémentaires. Ces ajouts, ainsi que d'autres modifications, sont récapitulés ci-dessous.

Si vous n'avez jamais utilisé l'API JavaScript, vous pouvez passer directement à la section traitant du modèle d'objet de document, le DOM (Document Object Model) (voir « [Modèle d'objet de document \(DOM\) de Flash](#) », à la page 13).

## Nouveaux objets

Les objets suivants constituent une nouveauté de Flash CS3 :

[Objet actionsPanel](#)

[Objet compilerErrors](#)

[Objet Oval](#)

[Objet Rectangle](#)

## Nouvelles méthodes et propriétés

Les méthodes et propriétés suivantes pour les objets existants constituent une nouveauté dans Flash CS3 :

```
document.as3AutoDeclare  
document.as3Dialect  
document.as3ExportFrame  
document.as3StrictMode  
document.as3WarningsMode  
document.asVersion  
document.canSaveAVersion()  
document.docClass  
document.getMobileSettings()  
document.getPlayerVersion()  
document.id  
document.resetOvalObject()  
document.resetRectangleObject()  
document.revertToLastVersion()  
document.saveAVersion()  
document.setMobileSettings()  
document.setOvalObjectProperty()  
document.setPlayerVersion()  
document.setRectangleObjectProperty()  
document.synchronizeWithHeadVersion()
```

```
element.getTransformationPoint()
element.rotation
element.scaleX
element.scaleY
element.setTransformationPoint()
element.skewX
element.skewY
element.transformX
element.transformY
element.x
element.y
filter.enabled
fl.actionsPanel
fl.as3PackagePaths
fl.addEventListener()
fl.closeAllPlayerDocuments()
fl.compilerErrors
fl.clipCopyString()
fl.downloadLatestVersion()
fl.packagePaths
fl.findDocumentDOM()
fl.findObjectInDocByName()
fl.findObjectInDocByType()
fl.removeEventListener()
fl.revertDocumentToLastVersion()
fl.resetPackagePaths()
fl.resetAS3PackagePaths()
fl.saveAVersionOfDocument()
fl.scriptURI
fl.selectElement()
fl.selectTool()
fl.synchronizeDocumentWithHeadVersion()
item.linkageBaseClass
shape.isOvalObject
shape.isRectangleObject
timeline.copyMotion()
timeline.copyMotionAsAS3()
timeline.pasteMotion()
```

## Autres modifications

Les méthodes suivantes sont dotées de nouveaux paramètres dans Flash CS3 :

```
document.addNewText()  
fl.closeAll()
```

La propriété suivante est déconseillé :

```
stroke.breakAtCorners
```

Le fichier mm\_jsapi.h, utilisé au moment de la mise en œuvre des fichiers d'extension de Flash (voir [Chapitre 3](#), « [Extensibilité de niveau C](#) », à la [page 611](#)), a changé.

## Modèle d'objet de document (DOM) de Flash

Le DOM de l'API JavaScript de Flash se compose d'un ensemble de fonctions de haut niveau (voir « [Fonctions et méthodes de haut niveau](#) », à la [page 21](#)) et de deux objets de haut niveau, l'objet FLfile et l'objet flash (fl). La disponibilité de chaque objet dans les scripts est assurée, puisqu'il existe dès l'exécution de l'environnement de création de Flash. Pour plus d'informations, voir les sections [Objet FLfile](#) et [Objet flash \(fl\)](#).

La référence à l'objet flash peut être `flash` ou `fl`. Par exemple, pour fermer tous les fichiers FLA ouverts, vous pouvez utiliser l'une ou l'autre de ces deux instructions :

```
flash.closeAll();  
fl.closeAll();
```

L'objet flash contient les objets *enfants* suivants :

Objet	Appel
<a href="#">Objet actionsPanel</a>	Utilisez <code>fl.actionsPanel</code> pour accéder à l'objet actionsPanel. Cet objet correspond au panneau Actions de l'environnement de programmation Flash.
<a href="#">Objet compilerErrors</a>	Utilisez <code>fl.compilerErrors</code> pour accéder à l'objet compilerErrors. Cet objet correspond au panneau Erreurs de compilation dans l'environnement de programmation Flash.
<a href="#">Objet componentsPanel</a>	Utilisez <code>fl.componentsPanel</code> pour accéder à l'objet componentsPanel. Cet objet correspond au panneau Composants de l'environnement de création Flash.
<a href="#">Objet Document</a>	Utilisez <code>fl.documents</code> pour récupérer dans un tableau la liste de tous les documents ouverts ; utilisez <code>fl.documents[index]</code> pour accéder à un document particulier, et <code>fl.getDocumentDOM()</code> pour accéder au document actif (celui qui a le focus).

Objet	Appel
Objet <a href="#">drawingLayer</a>	Utilisez <a href="#">fl.drawingLayer</a> pour accéder à l'objet <a href="#">drawingLayer</a> .
Objet <a href="#">Effect</a>	Utilisez <a href="#">fl.effects</a> pour récupérer un tableau contenant tous les descripteurs d'effets enregistrés au démarrage de Flash ; utilisez <a href="#">fl.effects[index]</a> pour accéder à un effet particulier, et <a href="#">fl.activeEffect</a> pour accéder au descripteur de l'effet qui est actuellement appliqué.
Objet <a href="#">Math</a>	Utilisez <a href="#">fl.Math</a> pour accéder à l'objet <a href="#">Math</a> .
Objet <a href="#">outputPanel</a>	Utilisez <a href="#">fl.outputPanel</a> pour accéder à l'objet <a href="#">outputPanel</a> . Cet objet correspond au panneau de sortie de l'environnement de création Flash.
Objet <a href="#">Project</a>	Utilisez <a href="#">fl.getProject()</a> afin de renvoyer un objet <a href="#">Project</a> pour le projet actuellement ouvert.
Objet <a href="#">Tools</a>	Utilisez <a href="#">fl.tools</a> pour accéder à un tableau d'objets <a href="#">Tools</a> .
Objet <a href="#">XMLUI</a>	Utilisez <a href="#">fl.xmlui</a> pour accéder à un objet de l'interface utilisateur XML (XMLUI). L'objet XMLUI permet de lire et définir les propriétés d'une boîte de dialogue de XMLUI.

## Objet Document

La propriété `fl.documents` est l'une des plus importantes de l'objet de haut niveau flash (consultez la propriété [fl.documents](#)). La propriété `fl.documents` contient un tableau des objets Document, dont chacun représente l'un des fichiers FLA actuellement ouverts dans l'environnement de création. Les propriétés de chaque objet Document représentent la plupart des éléments que peut contenir un fichier FLA. Une grande partie du DOM est donc composée d'objets et propriétés enfants de l'objet Document. Pour plus d'informations, consultez la section [Objet Document](#).

Ainsi, pour vous référer au premier document ouvert, vous pouvez utiliser l'instruction `flash.documents[0]` ou `fl.documents[0]`. Le premier document est le premier document Flash qui a été ouvert dans l'environnement de création au cours de la session actuelle. Lors de la fermeture du premier document, les index des autres documents ouverts sont décrémentés.

Pour connaître l'index d'un document particulier, utilisez l'instruction

```
flash.findDocumentIndex(nameOfDocument) ou
fl.findDocumentIndex(nameOfDocument) Voir la section fl.findDocumentIndex\(\).
```

Pour accéder au document qui a le focus, utilisez l'instruction `flash.getDocumentDOM()` ou `fl.getDocumentDOM()` Voir la section [fl.getDocumentDOM\(\)](#). Cette dernière est la syntaxe utilisée dans la plupart des exemples de ce guide.

Pour trouver un document particulier dans le tableau `fl.documents`, créez une boucle qui testera la propriété `document.name` de chaque document (voir [fl.documents](#) et [document.name](#)).

Vous pouvez accéder à tous les objets du DOM qui ne figurent pas dans la liste ci-dessus (voir « [Modèle d'objet de document \(DOM\) de Flash](#) », à la page 13) à l'aide de l'objet `Document`. Par exemple, pour accéder à la bibliothèque d'un document, utilisez la propriété `document.library`, qui renvoie un objet `library` :

```
fl.getDocumentDOM().library
```

Pour accéder au tableau des éléments de la bibliothèque, utilisez la propriété `library.items` ; chaque élément de ce tableau est un objet `Item` :

```
fl.getDocumentDOM().library.items
```

Pour accéder à un élément spécifique de la bibliothèque, indiquez un membre du tableau `library.items` :

```
fl.getDocumentDOM().library.items[0]
```

Autrement dit, l'objet `Item` est un enfant de l'objet `library`, qui est lui-même un enfant de l'objet `Document`. Pour plus d'informations, consultez [document.library](#), [Objet library](#), [library.items](#) et [Objet Item](#).

## Indication de la cible d'une action

Sauf indication contraire, les méthodes affectent la sélection ou le document qui a le focus. Par exemple, le script ci-dessous double la taille de la sélection actuelle, car aucun objet précis n'est spécifié :

```
fl.getDocumentDOM().scaleSelection(2, 2);
```

Il peut être nécessaire de définir comme cible d'une action l'élément actuellement sélectionné dans le document Flash. Pour ce faire, utilisez le tableau renvoyé par la propriété `document.selection` (voir [document.selection](#)). Le premier élément de ce tableau représente l'élément actuellement sélectionné, comme le montre l'exemple ci-dessous :

```
var accDescription = fl.getDocumentDOM().selection[0].description;
```

Le script ci-dessous double la taille du premier élément de la scène contenu dans le tableau d'éléments, et non pas l'élément sélectionné :

```
var element =  
    fl.getDocumentDOM().getTimeline().layers[0].frames[0].elements[0];  
if (element) {  
    element.width = element.width*2;  
    element.height = element.height*2;  
}
```

Il est également possible de parcourir en boucle la liste de tous les éléments de la scène pour augmenter leur largeur et leur hauteur d'un pourcentage spécifique, comme dans l'exemple suivant :

```
var elementArray =
    fl.getDocumentDOM().getTimeline().layers[0].frames[0].elements;
for (var i=0; i < elementArray.length; i++) {
    var offset = 10;
    elementArray[i].width += offset;
    elementArray[i].height += offset;
}
```

## Résumé de la structure du DOM

La liste suivante présente la structure arborescente du DOM. Les chiffres qui apparaissent au début de chaque ligne représentent le niveau de chaque objet. Par exemple, un objet précédé de « 03 » est un enfant de l'objet de niveau « 02 » qui le précède immédiatement, et celui-ci est lui-même un enfant de l'objet de niveau « 01 » qui le précède.

Dans certains cas, il est possible de référencer un objet en à travers une propriété de son objet parent. Par exemple, la propriété `document.timelines` contient un tableau d'objets Timeline (voir [document.timelines](#) et [Objet Timeline](#)). Ces propriétés sont indiquées dans l'arborescence ci-dessous.

Enfin, certains objets sont des sous-classes (et non pas des enfants) d'autres objets. Un objet qui est une sous-classe d'un autre objet possède ses propres méthodes et/ou propriétés, en sus des méthodes et propriétés de l'objet parent dont il est dérivé (superclasse). Dans la hiérarchie des objets, les sous-classes sont au même niveau que leur superclasse. Par exemple, l'objet Item est une superclasse de l'objet BitmapItem (voir [Objet Item](#) et [Objet BitmapItem](#)). Ces relations sont indiquées dans l'arborescence ci-dessous :

### 01 Fonctions et méthodes de haut niveau

#### 01 [Objet FLfile](#)

#### 01 [Objet flash \(fl\)](#)

##### 02 [Objet compilerErrors](#)

##### 02 [Objet componentsPanel](#)

##### 02 [Objet Document](#) (tableau `fl.documents`)

##### 03 [Objet Filter](#)

##### 03 [Objet Matrix](#)

##### 03 [Objet Fill](#)

##### 03 [Objet Stroke](#)



- 03 [Objet library](#)
  - 04 [Objet Item](#) (tableau [library.items](#) )
  - 04 [Objet BitmapItem](#) (sous-classe de [Objet Item](#))
  - 04 [Objet folderItem](#) (sous-classe de [Objet Item](#))
  - 04 [Objet fontItem](#) (sous-classe de [Objet Item](#))
  - 04 [Objet SoundItem](#) (sous-classe de [Objet Item](#))
  - 04 [Objet SymbolItem](#) (sous-classe de [Objet Item](#))
  - 04 [Objet VideoItem](#) (sous-classe de [Objet Item](#))
- 03 [Objet Timeline](#) (tableau [document.timelines](#) )
  - 04 [Objet Layer](#) (tableau [timeline.layers](#))
    - 05 [Objet Frame](#) (tableau [layer.frames](#) )
      - 06 [Objet Element](#) (tableau [frame.elements](#))
        - 07 [Objet Matrix](#) ([Element.matrix](#))
      - 06 [Objet Instance](#) (classe abstraite, sous-classe de [Objet Element](#))
      - 06 [Objet BitmapInstance](#) (sous-classe de [Objet Instance](#))
      - 06 [Objet CompiledClipInstance](#) (sous-classe de [Objet Instance](#))
      - 06 [Objet ComponentInstance](#) (sous-classe de [Objet SymbolInstance](#))
        - 07 [Objet Parameter](#) ([componentInstance.parameters](#))
      - 06 [Objet SymbolInstance](#) (sous-classe de [Objet Instance](#))
      - 06 [Objet Text](#) (sous-classe de [Objet Element](#))
        - 07 [Objet TextRun](#) (tableau [text.textRuns](#))
          - 08 [Objet TextAttrs](#) (tableau [textRun.textAttrs](#))
    - 06 [Objet Shape](#) (sous-classe de [Objet Element](#))
      - 07 [Objet Oval](#)
      - 07 [Objet Rectangle](#)
      - 07 [Objet Contour](#) (tableau [shape.contours](#))
        - 08 [Objet halfEdge](#)
        - 09 [Objet Vertex](#)
        - 09 [Objet Edge](#)

- 07 [Objet Edge](#) (`tableau shape.edges`)
  - 08 [Objet halfEdge](#)
  - 09 [Objet Vertex](#)
  - 09 [Objet Edge](#)
- 07 [Objet Vertex](#) (`tableau shape.vertices`)
  - 08 [Objet halfEdge](#)
  - 09 [Objet Vertex](#)
  - 09 [Objet Edge](#)
- 03 [Objet ScreenOutline](#)
  - 04 [Objet Screen](#) (`tableau screenOutline.screens`)
  - 05 [Objet Parameter](#) (`tableau screen.parameters`)
- 02 [Objet drawingLayer](#)
  - 03 [Objet Path](#)
  - 04 [Objet Contour](#)
- 02 [Objet Effect](#) (`tableau fl.effects`)
- 02 [Objet Math](#)
- 02 [Objet outputPanel](#)
- 02 [Objet Project](#)
  - 03 [Objet ProjectItem](#) (`tableau project.items`)
- 02 [Objet Tools](#) (`tableau fl.tools`)
  - 03 [Objet ToolObj](#) (`tableau tools.toolObjs`)
- 02 [Objet XMLUI](#)

# Exemples d'implémentation

Quelques exemples d'implémentations JSFL sont disponibles pour Adobe Flash CS3 Professional. Examinez ces fichiers et installez-les pour vous familiariser avec l'API JavaScript. Vous les trouverez dans le dossier Samples/ExtendingFlash, contenu dans le fichier ZIP qui est disponible à l'adresse [www.adobe.com/go/learn\\_fl\\_samples\\_fr](http://www.adobe.com/go/learn_fl_samples_fr).

## Exemple de commande Shape

Un exemple de script d'API JavaScript nommé Shape.jsfl est disponible dans le dossier ExtendingFlash/Shape (voir « [Exemples d'implémentation](#) » ci-dessus). Ce script affiche des informations sur les contours de la forme dans le panneau Sortie.

### Pour installer et exécuter le script Shape :

1. Copiez le fichier Shape.jsfl dans le dossier Configuration/Commands (voir « [Enregistrement de fichiers JSFL](#) », à la page 7).
2. Dans un document Flash (fichier FLA), sélectionnez un objet shape.
3. Sélectionnez Commandes > Forme pour exécuter le script.

## Exemple de commande de lecture et de définition de filtres

Un exemple de script d'API JavaScript nommé filtersGetSet.jsfl est disponible dans le dossier ExtendingFlash/filtersGetSet (voir « [Exemples d'implémentation](#) », à la page 19). Ce script ajoute des filtres à un objet sélectionné et affiche des informations sur les filtres en cours d'ajout dans le panneau Sortie.

### Pour installer et exécuter le script filtersGetSet :

1. Copiez le fichier filtersGetSet.jsfl dans le dossier Configuration/Commands (voir « [Enregistrement de fichiers JSFL](#) », à la page 7).
2. Dans un document Flash (fichier FLA), sélectionnez un objet de type texte, clip ou bouton.
3. Sélectionnez Commandes > filtersGetSet pour exécuter le script.

## Exemple d'outil PolyStar

Un exemple de script d'API JavaScript nommé PolyStar.jsfl est disponible dans le dossier ExtendingFlash/PolyStar (voir « [Exemples d'implémentation](#) », à la page 19).

Le fichier PolyStar.jsfl duplique l'outil Polygone intégré au panneau Outils de Flash. Le script illustre la compilation de l'outil Polygone à l'aide de l'API JavaScript et comprend des commentaires détaillés sur l'action du code. La lecture attentive de ce fichier permet de mieux comprendre l'utilisation de l'API JavaScript. Consultez également le fichier PolyStar.xml dans le répertoire Tools pour vous familiariser avec la compilation de votre propre outil.

## Exemple de panneau de tracé de bitmap

Un jeu de fichiers nommés TraceBitmap.fla et TraceBitmap.swf se trouvent dans le dossier ExtendingFlash/TraceBitmapPanel (voir « [Exemples d'implémentation](#) », à la page 19). Ces fichiers illustrent la manière de concevoir et créer un panneau de contrôle des fonctions de Flash. Ils montrent également l'emploi de la fonction `MMExecute()` pour appeler des commandes JavaScript depuis un script ActionScript.

### Pour exécuter l'exemple TraceBitmap :

1. Si Flash est en cours d'exécution, quittez l'application.
2. Copiez le fichier TraceBitmap.swf dans le dossier WindowSWF, un sous-dossier de Configuration (voir « [Enregistrement de fichiers JSFL](#) », à la page 7). Par exemple, pour Windows le dossier se trouve dans *lecteur d'amorçage*\Documents and Settings\utilisateur\Local Settings\Application Data\Adobe\Flash CS3\langue\Configuration\WindowSWF.
3. Démarrez Flash.
4. Créez ou ouvrez un document Flash (fichier FLA), puis importez-y un fichier bitmap ou une image JPEG.  
Vous pouvez utiliser le fichier flower.jpg qui se trouve dans le dossier TraceBitmapPanel, ou toute autre image de votre choix.
5. Lorsque l'image importée est sélectionnée, choisissez Fenêtre > Autres panneaux > TraceBitmap.
6. Cliquez sur Envoyer.  
L'image est convertie en un groupe de formes.

## Exemple de DLL

Un exemple de déploiement d'une DLL se trouve dans le dossier ExtendingFlash/dllSampleComputeSum (voir « [Exemples d'implémentation](#) », à la page 19). Pour plus d'informations sur la création des DLL, consultez le [Chapitre 3](#), « [Extensibilité de niveau C](#) », à la page 611.

# Fonctions et méthodes de haut niveau

Ce chapitre décrit les fonctions et méthodes de haut niveau qui sont disponibles dans l'interface de programmation d'application JavaScript (API JavaScript) d'Adobe Flash. Pour plus d'informations sur l'emplacement de stockage des fichiers de l'API JavaScript, consultez la section « [Enregistrement de fichiers JSFL](#) », à la page 7.

Les listes suivantes présentent les domaines de l'environnement de programmation concernés par chaque fonction ou méthode. Les fonctions et méthodes sont ensuite énumérées par ordre alphabétique.

## Méthodes globales

Vous pouvez appeler les méthodes suivantes à partir de tout script de l'API JavaScript :

```
alert()  
confirm()  
prompt()
```

## Effets de scénario

Les fonctions suivantes sont spécifiques aux effets de scénario :

```
configureEffect()  
executeEffect()  
removeEffect()
```

# Outils extensibles

Les fonctions suivantes sont disponibles dans les scripts qui créent des outils extensibles :

```
activate()
configureTool()
deactivate()
keyDown()
keyUp()
mouseDoubleClick()
mouseDown()
mouseMove()
mouseUp()
notifySettingsChanged()
setCursor()
```

## activate()

### Disponibilité

Flash MX 2004.

### Utilisation

```
function activate() {
    // statements
}
```

### Paramètres

Aucun.

### Valeur renvoyée

Aucune.

### Description

Fonction appelée lorsque l'outil extensible est activé (c'est-à-dire lorsqu'il est sélectionné dans le panneau Outils). Cette fonction vous permet d'exécuter toute tâche d'initialisation requise par l'outil.

## Exemple

L'exemple suivant définit la valeur de `tools.activeTool` lorsque l'outil extensible est sélectionné dans le panneau Outils :

```
function activate() {  
    var theTool = fl.tools.activeTool  
}
```

## Voir aussi

[tools.activeTool](#)

# alert()

## Disponibilité

Flash MX 2004.

## Utilisation

```
alert ( alertText )
```

## Paramètres

*alertText* Cette chaîne spécifie le message qui apparaît dans la boîte de dialogue Alerte.

## Valeur renvoyée

Aucune.

## Description

Cette méthode affiche une chaîne et un bouton OK dans une boîte de dialogue modale.

## Exemple

L'exemple suivant affiche le message « Process Complete » (Traitement terminé) dans une boîte de dialogue d'alerte.

```
alert("Process Complete");
```

## Voir aussi

[confirm\(\)](#), [prompt\(\)](#)

# configureEffect()

## Disponibilité

Flash MX 2004.

## Utilisation

```
function configureEffect() {  
    // Statements  
}
```

## Paramètres

Aucun.

## Valeur renvoyée

Aucune.

## Description

Fonction appelée une fois au chargement de Flash. Placez toute instruction d'initialisation globale associée à vos effets dans cette fonction. Elle ne peut pas être utilisée pour accéder aux paramètres des données propres à chaque instance d'un effet.

## Voir aussi

[executeEffect\(\)](#), [removeEffect\(\)](#)

# configureTool()

## Disponibilité

Flash MX 2004.

## Utilisation

```
function configureTool() {  
    // statements  
}
```

## Paramètres

Aucun.

## Valeur renvoyée

Aucune.



## Description

Fonction appelée à l'ouverture de Flash et lorsque l'outil extensible est chargé dans le panneau Outils. Elle permet de définir toutes informations nécessaires à Flash à propos de cet outil.

## Exemple

Les exemples suivants montrent deux implémentations possibles de cette fonction:

```
function configureTool() {  
    theTool = fl.tools.activeTool;  
    theTool.setToolName("myTool");  
    theTool.setIcon("myTool.png");  
    theTool.setMenuString("My Tool's menu string");  
    theTool.setToolTip("my tool's tool tip");  
    theTool.setOptionsFile( "mtTool.xml" );  
}
```

```
function configureTool() {  
    theTool = fl.tools.activeTool;  
    theTool.setToolName("ellipse");  
    theTool.setIcon("Ellipse.png");  
    theTool.setMenuString("Ellipse");  
    theTool.setToolTip("Ellipse");  
    theTool.showTransformHandles( true );  
}
```

# confirm()

## Disponibilité

Flash 8.

## Utilisation

```
confirm ( strAlert )
```

## Paramètres

*strAlert* Cette chaîne spécifie le message qui apparaît dans la boîte de dialogue d'alerte.

## Valeur renvoyée

Valeur booléenne : *true* si l'utilisateur clique sur OK, *false* s'il clique sur Annuler.

## Description

Cette méthode affiche dans une boîte de dialogue modale une chaîne accompagnée des boutons OK et Annuler.

### REMARQUE

Si aucun document (fichiers FLA) n'est ouvert, cette méthode échoue en produisant une condition d'erreur.

## Exemple

L'exemple suivant affiche le message « Sort data? » (Trier les données ?) dans une boîte de dialogue d'alerte.

```
confirm("Sort data?");
```

## Voir aussi

[alert\(\)](#), [prompt\(\)](#)

# deactivate()

## Disponibilité

Flash MX 2004.

## Utilisation

```
function deactivate() {  
    // statements  
}
```

## Paramètres

Aucun.

## Valeur renvoyée

Aucune.

## Description

Cette fonction est appelée lorsque l'outil extensible est désactivé (c'est-à-dire lorsque l'utilisateur sélectionne un autre outil). Elle permet d'effectuer toute éventuelle opération nécessaire avant la désactivation de cet outil.

## Exemple

L'exemple suivant affiche un message dans le panneau Sortie lorsque l'outil devient inactif :

```
function deactivate() {  
    fl.trace( "Tool is no longer active" );  
}
```

# executeEffect()

## Disponibilité

Flash MX 2004.

## Utilisation

```
function executeEffect() {  
    // statements  
}
```

## Paramètres

Aucun.

## Valeur renvoyée

Aucune.

## Description

Cette fonction est appelée lorsque l'utilisateur applique un effet pour la première fois ou modifie les propriétés d'un effet. Le code de cette fonction modifie le ou les objets originaux pour obtenir l'effet désiré. Il est également responsable de la copie de l'original dans un calque masqué si la fonction `removeEffect` le nécessite.

## Voir aussi

[configureEffect\(\)](#), [removeEffect\(\)](#)

# keyDown()

## Disponibilité

Flash MX 2004.

## Utilisation

```
function keyDown() {  
    // statements  
}
```

## Paramètres

Aucun.

## Valeur renvoyée

Aucune.

## Description

Cette fonction est appelée si l'outil extensible est actif lorsque l'utilisateur appuie sur une touche. Le script doit alors appeler `tools.getKeyDown()` pour identifier la touche en question.

## Exemple

L'exemple suivant affiche des informations sur la touche sur laquelle a appuyé l'utilisateur lorsque l'outil extensible est actif.

```
function keyDown() {  
    fl.trace("key " + fl.tools.getKeyDown() + " was pressed");  
}
```

## Voir aussi

[keyUp\(\)](#), [tools.getKeyDown\(\)](#)

# keyUp()

## Disponibilité

Flash MX 2004.

## Utilisation

```
function keyUp() {  
    // statements  
}
```

## Paramètres

Aucun.

## Valeur renvoyée

Aucune.

## Description

Cette fonction est appelée si l'outil extensible est actif lorsque l'utilisateur relâche une touche.

### Exemple

L'exemple suivant affiche un message dans le panneau Sortie lorsque l'outil extensible est actif et que l'utilisateur relâche une touche.

```
function keyUp() {  
    fl.trace("Key is released");  
}
```

### Voir aussi

[keyDown\(\)](#)

## mouseDoubleClick()

### Disponibilité

Flash MX 2004.

### Utilisation

```
function mouseDoubleClick() {  
    // statements  
}
```

### Paramètres

Aucun.

### Valeur renvoyée

Aucune.

### Description

Fonction appelée lorsque l'outil extensible est actif et que l'utilisateur double-clique sur la scène.

### Exemple

L'exemple suivant affiche un message dans le panneau Sortie lorsque l'outil extensible est actif et que l'utilisateur double-clique sur le bouton de la souris.

```
function mouseDoubleClick() {  
    fl.trace("Mouse was double-clicked");  
}
```

# mouseDown()

## Disponibilité

Flash MX 2004.

## Utilisation

```
function mouseDown( [ pt ] ) {  
    // statements  
}
```

## Paramètres

*pt* Point indiquant les coordonnées de la souris lors du clic. Il est transmis à la fonction lors de l'appui sur le bouton de la souris. Ce paramètre est facultatif.

## Valeur renvoyée

Aucune.

## Description

Cette fonction est appelée si l'outil extensible est actif lors d'un clic souris alors que le pointeur se trouve dans la scène.

## Exemple

Les exemples suivants illustrent l'utilisation de cette fonction lorsque l'outil extensible est actif. Le premier exemple affiche un message dans le panneau Sortie pour indiquer un clic souris. Le second exemple affiche les coordonnées *x* et *y* de la souris lors du clic.

```
function mouseDown() {  
    fl.trace("Mouse button has been pressed");  
}  
function mouseDown(pt) {  
    fl.trace("x = "+ pt.x+" :: y = "+pt.y);  
}
```

# mouseMove()

## Disponibilité

Flash MX 2004.

## Utilisation

```
function mouseMove( [ pt ] ) {  
    // statements  
}
```

## Paramètres

*pt* Point indiquant les coordonnées actuelles de la souris. Il est transmis à la fonction lors de tout mouvement de la souris, ce qui permet de connaître l'emplacement de celle-ci. Si la scène est en mode Modification ou Modification en place, les coordonnées du point sont relatives à l'objet en cours de modification. Sinon, les coordonnées du point sont relatives à la scène.

Ce paramètre est facultatif.

## Valeur renvoyée

Aucune.

## Description

Fonction appelée lorsque l'outil extensible est actif et que l'utilisateur passe au-dessus d'un point précis de la scène. Le ou les boutons de la souris peuvent être enfoncés ou non.

## Exemple

Les exemples suivants illustrent les utilisations possibles de cette fonction. Le premier exemple affiche un message dans le panneau Sortie pour indiquer un déplacement de la souris. Le second exemple affiche les coordonnées *x* et *y* de la souris lors de son déplacement.

```
function mouseMove() {  
    fl.trace("moving");  
}  
  
function mouseMove(pt) {  
    fl.trace("x = " + pt.x + " :: y = " + pt.y);  
}
```

# mouseUp()

## Disponibilité

Flash MX 2004.

## Utilisation

```
function mouseUp() {  
    // statements  
}
```

## Paramètres

Aucun.

## Valeur renvoyée

Aucune.

## Description

Fonction appelée lorsque l'outil extensible est actif et que l'utilisateur relâche le bouton de la souris après avoir cliqué sur la scène.

## Exemple

L'exemple suivant affiche un message dans le panneau Sortie lorsque l'outil extensible est actif et que l'utilisateur relâche le bouton de la souris.

```
function mouseUp() {  
    fl.trace("mouse is up");  
}
```

# notifySettingsChanged()

## Disponibilité

Flash MX 2004.

## Utilisation

```
function notifySettingsChanged() {  
    // statements  
}
```

## Paramètres

Aucun.



## Valeur renvoyée

Aucune.

## Description

Cette fonction est appelée lorsqu'un outil est actif et que l'utilisateur modifie les options de cet outil dans l'inspecteur Propriétés. La propriété `tools.activeTool` vous permet d'identifier les valeurs actuelles des options (voir [tools.activeTool](#)).

## Exemple

L'exemple suivant affiche un message dans le panneau Sortie lorsque l'outil extensible est actif et que l'utilisateur modifie les options correspondantes dans l'inspecteur Propriétés.

```
function notifySettingsChanged() {  
    var theTool = fl.tools.activeTool;  
    var newValue = theTool.myProp;  
}
```

# prompt()

## Disponibilité

Flash MX 2004.

## Utilisation

`prompt(promptMsg [, text])`

## Paramètres

*promptMsg* Chaîne affichée dans la boîte de dialogue d'invite (taille maximale de 256 caractères avec Macintosh OS X).

*texte* Chaîne facultative à afficher par défaut dans le champ texte.

## Valeur renvoyée

La chaîne saisie par l'utilisateur si celui-ci a ensuite cliqué sur OK, ou `null` si l'utilisateur a cliqué sur Annuler.

## Description

Cette méthode affiche dans une boîte de dialogue modale une invite et un texte facultatif, accompagnés des boutons OK et Annuler.

## Exemple

L'exemple suivant invite l'utilisateur à taper son nom. Si l'utilisateur saisit du texte et clique sur OK, le texte s'affiche dans le panneau de sortie.

```
var userName = prompt("Enter user name", "Type user name here");  
fl.trace(userName);
```

## Voir aussi

[alert\(\)](#), [confirm\(\)](#)

# removeEffect()

## Disponibilité

Flash MX 2004.

## Utilisation

```
function removeEffect() {  
    // statements  
}
```

## Paramètres

Aucun.

## Valeur renvoyée

Aucune.

## Description

Cette fonction est appelée lorsque l'utilisateur modifie les propriétés d'un effet ou choisit l'élément Supprimer l'effet dans le menu contextuel. Le code de cette fonction rétablit l'état d'origine du ou des objets. Par exemple, si l'effet sépare un texte en plusieurs fragments, la méthode `removeEffect()` restaure le texte d'origine.

## Voir aussi

[configureEffect\(\)](#), [executeEffect\(\)](#)

# setCursor()

## Disponibilité

Flash MX 2004.

## Utilisation

```
function setCursor() {  
    // statements  
}
```

## Paramètres

Aucun.

## Valeur renvoyée

Aucune.

## Description

Cette fonction est appelée lorsque l'outil extensible est actif et que l'utilisateur déplace la souris, ce qui permet au script de définir des pointeurs personnalisés. Le script doit appeler `tools.setCursor()` pour indiquer le pointeur à utiliser. La liste des pointeurs et des valeurs entières correspondantes figure dans la section `tools.setCursor()`.

## Exemple

```
function setCursor() {  
    fl.tools.setCursor( 1 );  
}
```



Ce chapitre décrit brièvement chacun des objets disponibles dans l'interface de programmation d'application JavaScript de Flash (API JavaScript). Ces objets sont classés par ordre alphabétique dans le tableau suivant.

Objet	Description
Objet <code>actionsPanel</code>	L'objet <code>actionsPanel</code> représente le panneau Actions actuellement affiché.
Objet <code>BitmapInstance</code>	L'objet <code>BitmapInstance</code> est une sous-classe de l' <a href="#">Objet Instance</a> et représente un bitmap dans une image.
Objet <code>BitmapItem</code>	Un objet <code>BitmapItem</code> réfère à une image bitmap dans la bibliothèque d'un document. L'objet <code>BitmapItem</code> est une sous-classe de l' <a href="#">Objet Item</a> .
Objet <code>CompiledClipInstance</code>	L'objet <code>CompiledClipInstance</code> est une sous-classe de l' <a href="#">Objet Instance</a> .
Objet <code>compilerErrors</code>	L'objet <code>compilerErrors</code> , représentant le panneau Erreurs de compilation, est une propriété de l' <a href="#">Objet flash (fl)</a> et est accessible via <code>fl.compilerErrors</code> .
Objet <code>ComponentInstance</code>	L'objet <code>ComponentInstance</code> est une sous-classe de l' <a href="#">Objet SymbolInstance</a> et représente un composant dans une image.
Objet <code>componentsPanel</code>	L'objet <code>componentsPanel</code> , qui représente le panneau Composants, est une propriété de l' <a href="#">Objet flash (fl)</a> et est accessible via <code>fl.componentsPanel</code> .
Objet Contour	Un objet Contour représente une trajectoire en tirets fermée marquant les limites d'une forme.
Objet Document	L'objet Document représente la scène.
Objet <code>drawingLayer</code>	L'objet <code>drawingLayer</code> est accessible en JavaScript sous forme d'un enfant de l'objet flash.

Objet	Description
<a href="#">Objet Edge</a>	L'objet Edge représente les limites d'une forme sur la scène.
<a href="#">Objet Effect</a>	L'objet Effect représente une instance d'un effet dans le scénario.
<a href="#">Objet Element</a>	Tout ce qui apparaît sur la scène est du type Element.
<a href="#">Objet Fill</a>	L'objet Fill contient toutes les propriétés du paramètre de couleur de remplissage du panneau Outils ou de la forme sélectionnée.
<a href="#">Objet Filter</a>	L'objet Filter contient toutes les propriétés de tous les filtres.
<a href="#">Objet flash (fl)</a>	L'objet flash représente l'application Flash.
<a href="#">Objet FLfile</a>	L'objet FLfile permet de créer des extensions Flash capables d'accéder aux dossiers et fichiers locaux, de les modifier et de les supprimer.
<a href="#">Objet folderItem</a>	L'objet folderItem est une sous-classe de l' <a href="#">Objet Item</a> .
<a href="#">Objet fontItem</a>	L'objet fontItem est une sous-classe de l' <a href="#">Objet Item</a> .
<a href="#">Objet Frame</a>	L'objet Frame représente les images dans le calque.
<a href="#">Objet halfEdge</a>	Coté dirigé du contour d'un <a href="#">Objet Shape</a> .
<a href="#">Objet Instance</a>	L'objet Instance est une sous-classe de l' <a href="#">Objet Element</a> .
<a href="#">Objet Item</a>	L'objet Item est une classe de base abstraite.
<a href="#">Objet Layer</a>	L'objet Layer représente un calque dans le scénario.
<a href="#">Objet library</a>	L'objet library représente le panneau Bibliothèque.
<a href="#">Objet Math</a>	L'objet Math est une propriété en lecture seule de l'objet flash ; voir <a href="#">fl.Math</a> .
<a href="#">Objet Matrix</a>	L'objet Matrix représente une matrice de transformation.
<a href="#">Objet outputPanel</a>	L'objet outputPanel représente le panneau de sortie, qui affiche des informations facilitant le débogage, par exemple les erreurs de syntaxe.
<a href="#">Objet Oval</a>	L'objet Oval est une forme tracée à l'aide de l'outil Ovale. Pour déterminer si un élément est un objet Oval, utilisez <a href="#">shape.isOvalObject</a> .

Objet	Description
Objet Parameter	Le type de l'objet Parameter est accessible à partir du tableau <code>screen.parameters</code> (qui correspond à la fenêtre de l'inspecteur des propriétés dans l'environnement de programmation Flash) ou du tableau <code>componentInstance.parameters</code> (qui correspond au composant Inspecteur des propriétés dans l'environnement de programmation).
Objet Path	L'objet Path définit une séquence de segments de ligne (droite, courbe ou les deux), qui est particulièrement destinée à la création d'outils extensibles.
Objet Project	L'objet Project représente un fichier de projet Flash (FLP).
Objet ProjectItem	L'objet ProjectItem représente un élément (fichier sur disque) qui a été ajouté à un projet.
Objet Rectangle	L'objet Rectangle est une forme tracée à l'aide de l'outil Rectangle. Pour déterminer si un élément est un objet Rectangle, utilisez <code>shape.isRectangleObject</code> .
Objet Screen	L'objet Screen représente un écran séparé dans une diapositive ou un formulaire.
Objet ScreenOutline	L'objet ScreenOutline représente le groupe d'écrans d'une diapositive ou d'un formulaire.
Objet Shape	L'objet Shape est une sous-classe de l' <b>Objet Element</b> . L'objet Shape autorise un contrôle plus précis que les API de dessin pour la manipulation ou la création de formes géométriques sur la scène.
Objet SoundItem	L'objet SoundItem est une sous-classe de l' <b>Objet Item</b> . Il représente un élément de la bibliothèque utilisé pour créer un son.
Objet Stroke	L'objet Stroke contient tous les paramètres d'un trait, y compris les paramètres personnalisés.
Objet SymbolInstance	L'objet SymbolInstance est une sous-classe de l' <b>Objet Instance</b> et représente un symbole dans une image.
Objet SymbolItem	L'objet SymbolItem est une sous-classe de l' <b>Objet Item</b> .
Objet Text	L'objet Text représente un élément de texte dans un document.

Objet	Description
<a href="#">Objet TextAttrs</a>	L'objet TextAttrs contient toutes les propriétés de texte pouvant être appliquées à une sous-sélection. Cet objet est une sous-classe de l' <a href="#">Objet Text</a> .
<a href="#">Objet TextRun</a>	L'objet TextRun représente un flux de caractères dont les attributs correspondent à toutes les propriétés de l' <a href="#">Objet TextAttrs</a> .
<a href="#">Objet Timeline</a>	L'objet Timeline représente le scénario de Flash, qui est accessible, pour le document actuel, à l'aide de <code>fl.getDocumentDOM().getTimeline()</code> .
<a href="#">Objet ToolObj</a>	Un objet toolObj représente un outil dans le panneau Outils.
<a href="#">Objet Tools</a>	L'objet Tools est accessible à partir de l'objet flash ( <code>fl.tools</code> ).
<a href="#">Objet Vertex</a>	L'objet Vertex est la partie de la structure de données d'une forme qui contient les coordonnées.
<a href="#">Objet VideoItem</a>	L'objet VideoItem est une sous-classe de l' <a href="#">Objet Item</a> .
<a href="#">Objet XMLUI</a>	L'objet XMLUI permet de lire et de définir les propriétés d'une boîte de dialogue XMLUI, mais aussi d'accepter ou d'annuler des opérations à partir d'une telle boîte de dialogue.





# Objet actionsPanel

## Disponibilité

Flash CS3 Professional.

## Description

L'objet `actionsPanel`, qui représente le panneau Actions en cours d'affichage, est une propriété de l'objet `flash` (`fl`) et est accessible via `fl.actionsPanel` (voir [Objet flash \(fl\)](#)).

## Résumé des méthodes de l'objet actionsPanel

Vous pouvez utiliser les méthodes suivantes avec l'objet `actionsPanel` :

Méthode	Description
<code>actionsPanel.getClassForObject()</code>	Renvoie la classe d'une variable spécifiée.
<code>actionsPanel.getScriptAssistMode()</code>	Précise si le mode Assistant de script est activé ou non.
<code>actionsPanel.getSelectedText()</code>	Renvoie le texte actuellement sélectionné dans le panneau Actions.
<code>actionsPanel.getText()</code>	Renvoie le texte dans le panneau Actions.
<code>actionsPanel.hasSelection()</code>	Indique si du texte est actuellement sélectionné dans le panneau Actions.
<code>actionsPanel.replaceSelectedText()</code>	Remplace le texte actuellement sélectionné par celui qui est spécifié.
<code>actionsPanel.setScriptAssistMode()</code>	Active ou désactive le mode Assistant de script.
<code>actionsPanel.setSelection()</code>	Sélectionne un jeu de caractères particulier dans le panneau Actions.
<code>actionsPanel.setText()</code>	Efface le texte affiché dans le panneau Actions pour ensuite y ajouter celui qui est spécifié.

# actionsPanel.getClassForObject()

## Disponibilité

Flash CS3 Professional.

## Utilisation

```
actionsPanel.getClassForObject(ASvariableName)
```

## Paramètres

*ASvariableName* Chaîne représentant le nom d'une variable ActionScript.

## Valeur renvoyée

Une chaîne représentant la classe dont *ASvariableName* est membre.

## Description

Méthode ; renvoie la classe de la variable spécifiée qui doit être définie dans le panneau Actions actuellement affiché. En outre, le curseur ou le texte sélectionné dans le panneau Actions doit être positionné après la définition de la variable.

## Exemple

L'exemple suivant affiche la classe attribuée à la variable myVar si le curseur est placé après l'instruction `var myVar:ActivityEvent;` dans le panneau Actions.

```
// Placer le code suivant dans le panneau Actions,  
// et positionner le curseur quelque part après la fin de la ligne  
var myVar:ActivityEvent;  
  
// Placer le code suivant dans le fichier JSFL  
var theClass = fl.actionsPanel.getClassForObject("myVar");  
fl.trace(theClass); // trace : "ActivityEvent"
```

# actionsPanel.getScriptAssistMode()

## Disponibilité

Flash CS3 Professional.

## Utilisation

```
actionsPanel.getScriptAssistMode()
```

## Paramètres

Aucun.

### Valeur renvoyée

Valeur booléenne précisant si le mode Assistant de script est activé (`true`) ou désactivé (`false`).

### Description

Méthode ; précise si le mode Assistant de script est activé ou non.

### Exemple

L'exemple suivant affiche un message si le mode Assistant de script n'est pas activé.

```
mAssist = fl.actionsPanel.getScriptAssistMode();  
if (!mAssist) {  
    alert("For more guidance when writing ActionScript code, try Script  
    Assist mode");  
}
```

### Voir aussi

[actionsPanel.setScriptAssistMode\(\)](#)

## actionsPanel.getSelectedText()

### Disponibilité

Flash CS3 Professional.

### Utilisation

```
actionsPanel.getSelectedText()
```

### Paramètres

Aucun.

### Valeur renvoyée

Chaîne contenant le texte actuellement sélectionné dans le panneau Actions.

### Description

Méthode ; renvoie le texte actuellement sélectionné dans le panneau Actions.

### Exemple

L'exemple suivant affiche le texte actuellement sélectionné dans le panneau Actions.

```
var apText = fl.actionsPanel.getSelectedText();  
fl.trace(apText);
```

### Voir aussi

[actionsPanel.getText\(\)](#), [actionsPanel.hasSelection\(\)](#),  
[actionsPanel.replaceSelectedText\(\)](#), [actionsPanel.setSelection\(\)](#)

## actionsPanel.getText()

### Disponibilité

Flash CS3 Professional.

### Utilisation

```
actionsPanel.getText()
```

### Paramètres

Aucun.

### Valeur renvoyée

Chaîne contenant tout le texte affiché dans le panneau Actions.

### Description

Méthode ; renvoie le texte dans le panneau Actions.

### Exemple

L'exemple suivant affiche le texte présent dans le panneau Actions.

```
var apText = fl.actionsPanel.getText();  
fl.trace(apText);
```

### Voir aussi

[actionsPanel.getSelectedText\(\)](#), [actionsPanel.setText\(\)](#)

## actionsPanel.hasSelection()

### Disponibilité

Flash CS3 Professional.

### Utilisation

```
actionsPanel.hasSelection()
```

### Paramètres

Aucun.

### Valeur renvoyée

Une valeur booléenne spécifiant si du texte est sélectionné (*true*) ou non (*false*) dans le panneau Actions.

### Description

Méthode ; indique si du texte est actuellement sélectionné dans le panneau Actions.

### Exemple

L'exemple suivant affiche le texte actuellement sélectionné dans le panneau Actions. Si aucun texte n'est sélectionné, l'intégralité du texte contenu dans le panneau Actions est affiché.

```
if (fl.actionsPanel.hasSelection()) {  
    var apText = fl.actionsPanel.getSelectedText();  
}  
else {  
    var apText = fl.actionsPanel.getText();  
}  
fl.trace(apText);
```

### Voir aussi

[actionsPanel.getSelectedText\(\)](#), [actionsPanel.getText\(\)](#),  
[actionsPanel.replaceSelectedText\(\)](#), [actionsPanel.setSelection\(\)](#)

## actionsPanel.replaceSelectedText()

### Disponibilité

Flash CS3 Professional.

### Utilisation

`actionsPanel.replaceSelectedText(replacementText)`

### Paramètres

*replacementText* Chaîne représentant le texte qui remplacer le texte sélectionné dans le panneau Actions.

### Valeur renvoyée

Valeur booléenne *true* si le panneau Actions est trouvé, sinon *false*.

### Description

Méthode ; remplace le texte actuellement sélectionné par celui qui est spécifié dans *replacementText*. Si *replacementText* contient plus de caractères que le texte sélectionné, tout caractère situé après le texte sélectionné suivra désormais *replacementText* et ne sera donc pas écrasé.

## Exemple

L'exemple suivant remplace le texte actuellement sélectionné dans le panneau Actions.

```
if (fl.actionsPanel.hasSelection()) {  
    fl.actionsPanel.replaceSelectedText("// © 2006 Adobe Inc.");  
}
```

## Voir aussi

[actionsPanel.getSelectedText\(\)](#), [actionsPanel.hasSelection\(\)](#),  
[actionsPanel.setSelection\(\)](#), [actionsPanel.setText\(\)](#)

# actionsPanel.setScriptAssistMode()

## Disponibilité

Flash CS3 Professional.

## Utilisation

```
actionsPanel.setScriptAssistMode(bScriptAssist)
```

## Paramètres

*bScriptAssist* Valeur booléenne spécifiant si le mode Assistant de script doit être activé ou désactivé.

## Valeur renvoyée

Une valeur booléenne précisant si le mode Assistant de script a été correctement activé ou désactivé.

## Description

Méthode ; active ou désactive le mode Assistant de script.

## Exemple

L'exemple suivant active/désactive le mode Assistant de script.

```
fl.trace(fl.actionsPanel.getScriptAssistMode());  
if (fl.actionsPanel.getScriptAssistMode()){  
    fl.actionsPanel.setScriptAssistMode(false);  
}  
else {  
    fl.actionsPanel.setScriptAssistMode(true);  
}  
fl.trace(fl.actionsPanel.getScriptAssistMode());
```

## Voir aussi

[actionsPanel.getScriptAssistMode\(\)](#)

# actionsPanel.setSelection()

## Disponibilité

Flash CS3 Professional.

## Utilisation

```
actionsPanel.setSelection(startIndex, numberOfChars)
```

## Paramètres

*startIndex* Nombre entier (commençant à zéro) qui indique le premier caractère à sélectionner.

*numberOfChars* Nombre entier spécifiant le nombre de caractères à sélectionner.

## Valeur renvoyée

Une valeur booléenne qui indique si les caractères demandés peuvent être sélectionnés (*true*) ou non (*false*).

## Description

Méthode ; sélectionne un jeu de caractères particulier dans le panneau Actions.

## Exemple

L'exemple suivant remplace les caractères « 2006 » affichés dans le panneau Actions par le texte spécifié.

```
// Tapez ce qui suit comme première ligne dans le panneau Actions
// 2006 - Répond à la demande de l'utilisateur 40196
// Taper ce qui suit dans le fichier JSFL
fl.actionsPanel.setSelection(3,4);
fl.actionsPanel.replaceSelectedText("// Dernière mise à jour : 2007");
```

## Voir aussi

```
actionsPanel.getSelectedText(), actionsPanel.hasSelection(),
actionsPanel.replaceSelectedText()
```



# actionsPanel.setText()

## Disponibilité

Flash CS3 Professional.

## Utilisation

```
actionsPanel.setText(replacementText)
```

## Paramètres

*replacementText* Chaîne représentant le texte à introduire dans le panneau Actions.

## Valeur renvoyée

La valeur booléenne `true` si le texte spécifié a été placé dans le panneau Actions, sinon `false`.

## Description

Méthode ; efface le texte affiché dans le panneau Actions pour ensuite y ajouter celui qui est spécifié dans *replacementText*.

## Exemple

L'exemple suivant remplace le texte actuellement affiché dans le panneau Actions par le texte spécifié.

```
fl.actionsPanel.setText("// A supprimé ce code - désormais inutile");
```

## Voir aussi

```
actionsPanel.getText(), actionsPanel.replaceSelectedText()
```

# Objet BitmapInstance

**Héritage** [Objet Element](#) > [Objet Instance](#) > objet BitmapInstance

## Disponibilité

Flash MX 2004.

## Description

L'objet BitmapInstance est une sous-classe de l'objet Instance et représente un bitmap dans une image (voir [Objet Instance](#)).

## Résumé des méthodes de l'objet BitmapInstance

Outre les méthodes de l'[Objet Instance](#), les méthodes suivantes sont disponibles avec l'objet BitmapInstance :

Méthode	Description
<a href="#">bitmapInstance.getBits()</a>	Vous permet de créer des effets de bitmap en extrayant les bits du bitmap, en les manipulant et en les renvoyant à Flash.
<a href="#">bitmapInstance.setBits()</a>	Active les bits d'un élément bitmap existant.

## Résumé des propriétés de l'objet BitmapInstance

Outre les propriétés de l'[Objet Instance](#), les propriétés suivantes sont également disponibles avec l'objet BitmapInstance :

Propriété	Description
<a href="#">bitmapInstance.hPixels</a>	En lecture seule ; entier indiquant la largeur du bitmap, en pixels.
<a href="#">bitmapInstance.vPixels</a>	En lecture seule ; entier indiquant la hauteur du bitmap, en pixels.

# bitmapInstance.getBits()

## Disponibilité

Flash MX 2004.

## Utilisation

`bitmapInstance.getBits()`

## Paramètres

Aucun.

## Valeur renvoyée

Un objet qui contient les propriétés `width` (largeur), `height` (hauteur), `depth` (profondeur), `bits` et, si le bitmap comporte une table de couleurs, `cTab`. L'élément `bits` est un tableau d'octets. L'élément `cTab` est un tableau de valeurs colorimétriques sous la forme `"#RRGGBB"`. La longueur du tableau est celle de la table de couleurs.

Le tableau d'octets n'est utilisable que s'il est référencé par une DLL ou une bibliothèque partagée. En règle générale, il est uniquement destiné à la création d'un outil extensible ou d'un effet. Pour plus d'informations sur la création de DLL en vue de les utiliser avec du code JavaScript Flash, consultez le [Chapitre 3, « Extensibilité de niveau C »](#).

## Description

Méthode permettant de créer des effets de bitmap en extrayant les bits du bitmap, en les manipulant et en les renvoyant à Flash. Voir aussi [bitmapInstance.setBits\(\)](#).

## Exemple

Le code suivant crée une référence à l'objet sélectionné, teste s'il est de type bitmap, et lit la hauteur, la largeur et la profondeur de couleurs du bitmap :

```
var isBitmap = fl.getDocumentDOM().selection[0].instanceType;
if(isBitmap == "bitmap"){
    var bits = fl.getDocumentDOM().selection[0].getBits();
    fl.trace("height = " + bits.height);
    fl.trace("width = " + bits.width);
    fl.trace("depth = " + bits.depth);
}
```

## Voir aussi

[bitmapInstance.setBits\(\)](#)

# bitmapInstance.hPixels

## Disponibilité

Flash MX 2004.

## Utilisation

`bitmapInstance.hPixels`

## Description

Propriété en lecture seule ; entier indiquant la largeur du bitmap, c'est-à-dire le nombre de pixels dans la dimension horizontale.

## Exemple

Le code suivant lit la largeur du bitmap en pixels :

```
// obtenir le nombre de pixels sur l'axe horizontal.  
var bmObj = fl.getDocumentDOM().selection[0];  
var isBitmap = bmObj.instanceType;  
if(isBitmap == "bitmap"){  
    var numHorizontalPixels = bmObj.hPixels;  
}
```

## Voir aussi

[bitmapInstance.vPixels](#)

# bitmapInstance.setBits()

## Disponibilité

Flash MX 2004.

## Utilisation

`bitmapInstance.setBits(bitmap)`

## Paramètres

*bitmap* Un objet qui contient les propriétés *height* (hauteur), *width* (largeur), *depth* (profondeur), *bits* et *cTab*. Les propriétés *height*, *width* et *depth* sont des entiers. L'élément *bits* est un tableau d'octets. La propriété *cTab* n'est nécessaire que pour les bitmaps ayant une profondeur de couleurs égale ou inférieure à 8 bits. C'est une chaîne qui représente une valeur colorimétrique sous la forme "#RRGGBB".

### REMARQUE

Le tableau d'octets n'est utilisable que s'il est référencé par une bibliothèque externe. En règle générale, il est uniquement destiné à la création d'un outil extensible ou d'un effet.

## Valeur renvoyée

Aucune.

## Description

Méthode activant les bits d'un élément bitmap existant. Elle permet de créer des effets de bitmap en extrayant les bits du bitmap, en les manipulant et en renvoyant le bitmap à Flash.

## Exemple

Le code suivant teste si la sélection est de type bitmap, puis fixe la hauteur de ce bitmap à 150 pixels :

```
var isBitmap = fl.getDocumentDOM().selection[0].instanceType;
if(isBitmap == "bitmap"){
    var bits = fl.getDocumentDOM().selection[0].getBits();
    bits.height = 150;
    fl.getDocumentDOM().selection[0].setBits(bits);
}
```

## Voir aussi

[bitmapInstance.getBits\(\)](#)

# bitmapInstance.vPixels

## Disponibilité

Flash MX 2004.

## Utilisation

`bitmapInstance.vPixels`

## Description

Propriété en lecture seule ; entier indiquant la hauteur du bitmap, c'est-à-dire le nombre de pixels dans la dimension verticale.

## Exemple

Le code suivant lit la hauteur du bitmap en pixels :

```
// Obtenir le nombre de pixels sur l'axe vertical.  
var bmObj = fl.getDocumentDOM().selection[0];  
var isBitmap = bmObj.instanceType;  
if(isBitmap == "bitmap"){  
    var numVerticalPixels = bmObj.vPixels;  
}
```

## Voir aussi

[bitmapInstance.hPixels](#)

# Objet BitmapItem

Héritage [Objet Item](#) > objet BitmapItem

## Disponibilité

Flash MX 2004.

## Description

Un objet BitmapItem réfère à une image bitmap dans la bibliothèque d'un document. L'objet BitmapItem est une sous-classe de l'objet Item (voir [Objet Item](#)).

## Résumé des propriétés de l'objet BitmapItem

Outre les propriétés de l'[Objet Item](#), l'objet BitmapItem possède les propriétés suivantes :

Propriété	Description
<code>bitmapItem.allowSmoothing</code>	Valeur booléenne qui spécifie si le lissage d'un bitmap est autorisé ou non.
<code>bitmapItem.compressionType</code>	Chaîne qui détermine le type de compression d'image appliquée au bitmap.
<code>bitmapItem.quality</code>	Entier qui définit la qualité du bitmap.
<code>bitmapItem.useImportedJPEGQuality</code>	Valeur booléenne qui spécifie si la qualité JPEG importée par défaut doit être utilisée ou non.

## bitmapItem.allowSmoothing

### Disponibilité

Flash MX 2004.

### Utilisation

`bitmapItem.allowSmoothing`

### Description

Propriété : valeur booléenne indiquant si le lissage d'un bitmap est autorisé (`true`) ou non (`false`).

### Exemple

Le code suivant met la propriété `allowSmoothing` du premier élément de la bibliothèque du document actuel à `true` :

```
fl.getDocumentDOM().library.items[0].allowSmoothing = true;  
alert(fl.getDocumentDOM().library.items[0].allowSmoothing);
```

# bitmapItem.compressionType

## Disponibilité

Flash MX 2004.

## Utilisation

`bitmapItem.compressionType`

## Description

Propriété : chaîne déterminant le type de compression d'image appliquée au bitmap.

Les valeurs acceptables sont `photo` ou `lossless`. Si la valeur de

`bitmapItem.useImportedJPEGQuality` est `false`, `photo` correspond à une qualité JPEG

comprise entre 0 et 100 ; si la valeur de `bitmapItem.useImportedJPEGQuality` est `true`,

"`photo`" correspond à la qualité JPEG avec la valeur de qualité par défaut du document.

La valeur "`lossless`" correspond aux formats GIF ou PNG

(voir [bitmapItem.useImportedJPEGQuality](#)).

## Exemple

Le code suivant met la propriété `compressionType` du premier élément de la bibliothèque du document actuel à `photo` :

```
fl.getDocumentDOM().library.items[0].compressionType = "photo";  
alert(fl.getDocumentDOM().library.items[0].compressionType);
```

# bitmapItem.quality

## Disponibilité

Flash MX 2004.

## Utilisation

`bitmapItem.quality`

## Description

Propriété ; entier définissant la qualité du bitmap. Pour utiliser la qualité par défaut du

document, spécifiez `-1` ; sinon, spécifiez un entier entre 0 et 100. Uniquement disponible

pour la compression JPEG.

## Exemple

Le code suivant met la propriété `quality` du premier élément de la bibliothèque du document actuel à `65` :

```
fl.getDocumentDOM().library.items[0].quality = 65;  
alert(fl.getDocumentDOM().library.items[0].quality);
```



# bitmapItem.useImportedJPEGQuality

## Disponibilité

Flash MX 2004.

## Utilisation

`bitmapItem.useImportedJPEGQuality`

## Description

Propriété : valeur booléenne indiquant si la qualité JPEG importée par défaut doit être utilisée (`true`) ou non (`false`). Uniquement disponible pour la compression JPEG.

## Exemple

Le code suivant met la propriété `useImportedJPEGQuality` du premier élément de la bibliothèque du document actuel à `true` :

```
fl.getDocumentDOM().library.items[0].useImportedJPEGQuality = true;  
alert(fl.getDocumentDOM().library.items[0].useImportedJPEGQuality);
```

# Objet CompiledClipInstance

**Héritage** [Objet Element](#) > [Objet Instance](#) > objet CompiledClipInstance

## Disponibilité

Flash MX 2004.

## Description

L'objet CompiledClipInstance est une sous-classe de l'objet Instance. C'est essentiellement une occurrence d'un clip qui a été convertie en élément compilé de la bibliothèque de clips (voir [Objet Instance](#)).

## Résumé des propriétés de l'objet CompiledClipInstance

Outre les propriétés de l'[Objet Instance](#), l'objet CompiledClipInstance possède les propriétés suivantes :

Propriété	Description
<a href="#">compiledClipInstance.accName</a>	Chaîne équivalente au champ Nom du panneau Accessibilité.
<a href="#">compiledClipInstance.actionScript</a>	Chaîne représentant le code ActionScript de cette occurrence ; équivaut à <a href="#">symbolInstance.actionScript</a> .
<a href="#">compiledClipInstance.description</a>	Chaîne équivalente au champ Description du panneau Accessibilité.
<a href="#">compiledClipInstance.forceSimple</a>	Valeur booléenne qui active et désactive l'accessibilité aux enfants de l'objet.
<a href="#">compiledClipInstance.shortcut</a>	Chaîne équivalente au champ Raccourci du panneau Accessibilité.
<a href="#">compiledClipInstance.silent</a>	Valeur booléenne qui active et désactive l'accessibilité à l'objet ; équivalente à l'inverse de la fonction Rendre l'objet accessible dans le panneau Accessibilité.
<a href="#">compiledClipInstance.tabIndex</a>	Entier équivalent au champ Index de tabulation du panneau Accessibilité.

# compiledClipInstance.accName

## Disponibilité

Flash MX 2004.

## Utilisation

`compiledClipInstance.accName`

## Description

Propriété ; chaîne équivalente au champ Nom du panneau Accessibilité. Les logiciels de lecture vocale identifient l'objet en prononçant ce nom.

## Exemple

L'exemple suivant lit et définit le nom d'accessibilité du premier objet sélectionné :

```
// Obtenir le nom de l'objet.  
var theName = fl.getDocumentDOM().selection[0].accName;  
// Définir le nom de l'objet.  
fl.getDocumentDOM().selection[0].accName = 'Home Button';
```

# compiledClipInstance.actionScript

## Disponibilité

Flash MX 2004.

## Utilisation

`compiledClipInstance.actionScript`

## Description

Propriété ; chaîne représentant le code ActionScript de cette occurrence ; équivaut à [symbolInstance.actionScript](#).

## Exemple

Le code suivant assigne du code ActionScript aux éléments spécifiés :

```
// Assigner du code ActionScript à une instance spécifique  
// d'un clip compilé.  
fl.getDocumentDOM().getTimeline().layers[0].frames[0].elements[0]  
    .actionScript = "on(click) {trace('button is clicked');}";  
// Assigner du code ActionScript à l'instance de clip compilé sélectionnée.  
fl.getDocumentDOM().selection[0].actionScript =  
    "on(click) {trace('button is clicked');}";
```

## compiledClipInstance.description

### Disponibilité

Flash MX 2004.

### Utilisation

`compiledClipInstance.description`

### Description

Propriété ; chaîne équivalente au champ Description du panneau Accessibilité. Cette description est lue par les logiciels de lecture vocale.

### Exemple

L'exemple suivant montre comment lire et définir la propriété `description` :

```
// Obtenir la description de la sélection.  
var theDescription = fl.getDocumentDOM().selection[0].description;  
// Définir la description de la sélection.  
fl.getDocumentDOM().selection[0].description =  
    "This is compiled clip number 1";
```

## compiledClipInstance.forceSimple

### Disponibilité

Flash MX 2004.

### Utilisation

`compiledClipInstance.forceSimple`

### Description

Propriété ; valeur booléenne qui active et désactive l'accessibilité aux enfants de l'objet. Cette propriété est équivalente à l'inverse du paramètre `Rendre les objets enfant accessibles` du panneau Accessibilité. Si `forceSimple` est `true`, son effet est similaire à la désactivation de l'option `Rendre les objets enfant accessibles`. Si `forceSimple` est `false`, son effet est similaire à l'activation de l'option `Rendre les objets enfant accessibles`.

### Exemple

L'exemple suivant montre comment lire et définir la propriété `forceSimple` :

```
// Vérifier si les enfants de l'objet sont accessibles.  
var areChildrenAccessible = fl.getDocumentDOM().selection[0].forceSimple;  
// Activer l'accessibilité aux enfants de l'objet.  
fl.getDocumentDOM().selection[0].forceSimple = false;
```

## compiledClipInstance.shortcut

### Disponibilité

Flash MX 2004.

### Utilisation

`compiledClipInstance.shortcut`

### Description

Propriété ; chaîne équivalente au champ Raccourci du panneau Accessibilité. Ce raccourci est lu par les logiciels de lecture vocale. Cette propriété n'est pas disponible pour les champs texte dynamique.

### Exemple

L'exemple suivant montre comment lire et définir la propriété `shortcut` :

```
// Obtenir le raccourci de l'objet.  
var theShortcut = fl.getDocumentDOM().selection[0].shortcut;  
// Définir le raccourci de l'objet.  
fl.getDocumentDOM().selection[0].shortcut = "Ctrl+I";
```

## compiledClipInstance.silent

### Disponibilité

Flash MX 2004.

### Utilisation

`compiledClipInstance.silent`

### Description

Propriété ; valeur booléenne qui active et désactive l'accessibilité à l'objet ; équivalente à l'inverse de la fonction Rendre l'objet accessible dans le panneau Accessibilité. Ainsi, si `silent` est `true`, l'option Rendre l'objet accessible est désactivée. Si `silent` est `false`, l'option Rendre l'objet accessible est activée.

### Exemple

L'exemple suivant montre comment lire et définir la propriété `silent` :

```
// Vérifier que l'objet est accessible.  
var isSilent = fl.getDocumentDOM().selection[0].silent;  
// Rendre l'objet accessible.  
fl.getDocumentDOM().selection[0].silent = false;
```

# compiledClipInstance.tabIndex

## Disponibilité

Flash MX 2004.

## Utilisation

`compiledClipInstance.tabIndex`

## Description

Propriété ; entier équivalent au champ Index de tabulation du panneau Accessibilité.  
Cette propriété crée l'ordre de tabulation dans lequel les objets seront sélectionnés lorsque l'utilisateur appuiera sur la touche de tabulation.

## Exemple

L'exemple suivant montre comment lire et définir la propriété `tabIndex` :

```
// Obtenir le tabIndex de l'objet.  
var theTabIndex = fl.getDocumentDOM().selection[0].tabIndex;  
// Définir le tabIndex de l'objet.  
fl.getDocumentDOM().selection[0].tabIndex = 1;
```

# Objet compilerErrors

## Disponibilité

Flash CS3 Professional.

## Description

L'objet `compilerErrors`, qui représente le panneau Erreurs de compilation, est une propriété de l'objet `flash (fl)` et est accessible via `fl.compilerErrors` (voir [Objet flash \(fl\)](#)).

## Résumé des méthodes de l'objet compilerErrors

Vous pouvez utiliser les méthodes suivantes avec l'objet `compilerErrors` :

Méthode	Description
<code>compilerErrors.clear()</code>	Efface le contenu du panneau Erreurs de compilation.
<code>compilerErrors.save()</code>	Enregistre le contenu du panneau Erreurs de compilation dans un fichier texte local.

## `compilerErrors.clear()`

### Disponibilité

Flash CS3 Professional.

### Utilisation

```
compilerErrors.clear()
```

### Paramètres

Aucun.

### Valeur renvoyée

Aucune.

### Description

Méthode ; efface le contenu du panneau Erreurs de compilation.

### Exemple

L'exemple suivant efface le contenu du panneau Erreurs de compilation :

```
fl.compilerErrors.clear();
```

## Voir aussi

`compilerErrors.save()`

# compilerErrors.save()

## Disponibilité

Flash CS3 Professional.

## Utilisation

```
compilerErrors.save(fileURI [, bAppendToFile [, bUseSystemEncoding]])
```

## Paramètres

*fileURI* Chaîne exprimée sous la forme « fichier:/// URI » et indiquant le nom du fichier enregistré. Si *fileURI* existe déjà alors que vous n'avez pas précisé de valeur `true` pour *bAppendToFile*, le remplacement de *fileURI* s'effectue sans message d'avertissement.

*bAppendToFile* Valeur booléenne facultative précisant si le contenu du panneau Erreurs de compilation doit être ajouté à *fileURI* (`true`) ou non (`false`). La valeur par défaut est `false`.

*bUseSystemEncoding* Valeur booléenne facultative précisant si le texte du panneau Erreur de compilation doit être enregistré ou non en utilisant le codage système. Si cette valeur est `false` (valeur par défaut), le texte du panneau Erreurs de compilation est enregistré en codage UTF-8 et intègre des caractères marqueurs d'ordre d'octets au début du texte. La valeur par défaut est `false`.

## Valeur renvoyée

Aucune.

## Description

Méthode ; enregistre le contenu du panneau Erreurs de compilation dans un fichier texte local.

## Exemple

L'exemple suivant enregistre le contenu du panneau Erreurs de compilation dans le fichier nommé `errors.log` qui est conservé dans le dossier `C:\tests` :

```
fl.compilerErrors.save("file:///c:/tests/errors.log");
```

## Voir aussi

`compilerErrors.clear()`



# Objet ComponentInstance

**Héritage** [Objet Element](#) > [Objet Instance](#) > [Objet SymbolInstance](#) > objet ComponentInstance

## Disponibilité

Flash MX 2004.

## Description

L'objet ComponentInstance est une sous-classe de l'objet SymbolInstance et représente un composant dans une image (voir [Objet SymbolInstance](#)).

## Résumé des propriétés de l'objet ComponentInstance

Outre les propriétés de l'[Objet SymbolInstance](#), l'objet ComponentInstance possède également la propriété suivante :

Propriété	Description
<a href="#">componentInstance.parameters</a>	Propriété en lecture seule ; tableau contenant les propriétés ActionScript 2.0, accessibles à l'aide de l'Inspecteur de propriétés du composant.

## componentInstance.parameters

### Disponibilité

Flash MX 2004.

### Utilisation

`componentInstance.parameters`

### Description

Propriété en lecture seule ; tableau contenant les propriétés ActionScript 2.0 accessibles à l'aide de l'Inspecteur de propriétés du composant. Voir la section « [Objet Parameter](#) », à la page 406.

### Exemple

L'exemple suivant montre comment lire et définir la propriété `parameters` :

```
var parms = fl.getDocumentDOM().selection[0].parameters;  
parms[0].value = "some value";
```

### Voir aussi

[Objet Parameter](#)

# Objet componentsPanel

## Disponibilité

Flash MX 2004.

## Description

L'objet `componentsPanel`, qui représente le panneau Composants, est une propriété de l'objet `flash (fl)` et est accessible via `fl.componentsPanel` (voir [Objet flash \(fl\)](#)).

## Résumé des méthodes de l'objet componentsPanel

Vous pouvez utiliser les méthodes suivantes avec l'objet `componentsPanel` :

Méthode	Description
<code>componentsPanel.addItemToDocument()</code>	Ajoute le composant spécifié au document à l'emplacement indiqué.
<code>componentsPanel.reload()</code>	Actualise la liste de composants affichée dans le panneau Composants.

## `componentsPanel.addItemToDocument()`

## Disponibilité

Flash MX 2004.

## Utilisation

`componentsPanel.addItemToDocument(position, categoryName, componentName)`

## Paramètres

*position* Un point (par exemple, {x:0, y:100}) indiquant l'emplacement auquel le composant doit être ajouté. Spécifiez le paramètre *position* par rapport au point central du composant, et non par rapport à son point d'alignement (également *point d'origine* ou *point zéro*).

*categoryName* Chaîne spécifiant le nom de la catégorie du composant (par exemple, "Data"). Les noms de catégories valides sont énumérés dans le panneau Composants.

*componentName* Chaîne spécifiant le nom du composant dans la catégorie spécifiée (par exemple, `WebServiceConnector`). Les noms de composants valides sont énumérés dans le panneau Composants.

### Valeur renvoyée

Aucune.

### Description

Ajoute le composant spécifié au document à l'emplacement indiqué.

### Exemple

Les exemples ci-dessous illustrent diverses façons d'utiliser cette méthode :

```
fl.componentsPanel.addItemToDocument({x:0, y:0}, "User Interface",  
    "CheckBox");  
fl.componentsPanel.addItemToDocument({x:0, y:100}, "Data",  
    "WebServiceConnector");  
fl.componentsPanel.addItemToDocument({x:0, y:200}, "User Interface",  
    "Button");
```

## componentsPanel.reload()

### Disponibilité

Flash 8.

### Utilisation

```
componentsPanel.reload()
```

### Paramètres

Aucun.

### Valeur renvoyée

Valeur booléenne `true` si la liste du panneau Composants est actualisée, et `false` dans le cas contraire.

### Description

Méthode qui actualise la liste de composants affichée dans le panneau Composants.

### Exemple

L'exemple suivant actualise le panneau Composants :

```
fl.componentsPanel.reload();
```

# Objet Contour

## Disponibilité

Flash MX 2004.

## Description

Un objet Contour représente une trajectoire en tirets fermée marquant les limites d'une forme.

## Résumé des méthodes de l'objet Contour

Vous pouvez utiliser la méthode suivante avec l'objet Contour :

Propriété	Description
<code>contour.getHalfEdge()</code>	Renvoie un <a href="#">Objet halfEdge</a> sur le contour de la sélection.

## Résumé des propriétés de l'objet Contour

Vous pouvez utiliser les propriétés suivantes avec l'objet Contour :

Propriété	Description
<code>contour.interior</code>	Lecture seule : la valeur est <code>true</code> si le contour délimite une zone, ou <code>false</code> sinon.
<code>contour.orientation</code>	Lecture seule ; entier indiquant l'orientation du contour.

## contour.getHalfEdge()

## Disponibilité

Flash MX 2004.

## Utilisation

`contour.getHalfEdge()`

## Paramètres

Aucun.

## Valeur renvoyée

Un [Objet halfEdge](#)

## Description

Méthode ; renvoie un [Objet halfEdge](#) sur le contour de la sélection.

## Exemple

Cet exemple analyse tous les contours d'une forme sélectionnée et affiche les coordonnées des sommets dans le panneau de sortie :

```
// forme sélectionnée,

var elt = fl.getDocumentDOM().selection[0];
elt.beginEdit();

var contourArray = elt.contours;
var contourCount = 0;
for (i=0; i<contourArray.length; i++)
{
    var contour = contourArray[i];
    contourCount++;
    var he = contour.getHalfEdge();

    var iStart = he.id;
    var id = 0;
    while (id != iStart)
    {
        // Obtenir le sommet suivant.
        var vrt = he.getVertex();

        var x = vrt.x;
        var y = vrt.y;
        fl.trace("vrt: " + x + ", " + y);

        he = he.getNext();
        id = he.id;
    }
}
elt.endEdit();
```

## contour.interior

### Disponibilité

Flash MX 2004.

### Utilisation

`contour.interior`

### Description

Propriété en lecture seule ; la valeur est `true` si le contour délimite une zone ou `false` dans le cas contraire.

### Exemple

Cet exemple analyse tous les contours d'une forme sélectionnée et affiche la valeur de la propriété `interior` pour chaque contour dans le panneau de sortie :

```
var elt = fl.getDocumentDOM().selection[0];
elt.beginEdit();

var contourArray = elt.contours;

var contourCount = 0;
for (i=0; i<contourArray.length; i++) {
    var contour = contourArray[i];
    fl.trace("Next Contour, interior:" + contour.interior );
    contourCount++;
}
elt.endEdit();
```

## contour.orientation

### Disponibilité

Flash MX 2004.

### Utilisation

`contour.orientation`

### Description

Propriété en lecture seule ; entier indiquant l'orientation du contour. La valeur de l'entier est -1 si l'orientation est anti-horaire, 1 si l'objet est orienté dans le sens horaire, et 0 si ce contour ne renferme pas de surface.

## Exemple

Cet exemple analyse tous les contours de la forme sélectionnée et affiche la valeur de la propriété `orientation` pour chaque contour dans le panneau de sortie :

```
var elt = fl.getDocumentDOM().selection[0];
elt.beginEdit();

var contourArray = elt.contours;

var contourCount = 0;
for (i=0; i<contourArray.length; i++) {
    var contour = contourArray[i];
    fl.trace("Next Contour, orientation:" + contour.orientation);
    contourCount++;
}
elt.endEdit();
```

# Objet Document

## Disponibilité

Flash MX 2004.

## Description

L'objet Document représente la scène. En d'autres termes, seuls les fichiers FLA sont considérés comme des documents. Si vous voulez renvoyer l'objet Document pour le document actif, utilisez `fl.getDocumentDOM()`.

## Résumé des méthodes de l'objet Document

Vous pouvez utiliser les méthodes suivantes avec l'objet Document :

Méthode	Description
<code>document.addDataToDocument()</code>	Enregistre les données spécifiées avec un document.
<code>document.addDataToSelection()</code>	Enregistre les données spécifiées avec le ou les objets sélectionnés.
<code>document.addFilter()</code>	Applique un filtre aux objets sélectionnés.
<code>document.addItem()</code>	Ajoute à l'objet Document spécifié un élément provenant d'un document ouvert ou d'une bibliothèque.
<code>document.addNewLine()</code>	Ajoute un nouveau trajet entre deux points.
<code>document.addNewOval()</code>	Ajoute un nouvel objet Oval dans le rectangle de délimitation spécifié.
<code>document.addNewPublishProfile()</code>	Ajoute un nouveau profil de publication et l'active.
<code>document.addNewRectangle()</code>	Ajoute un nouveau rectangle ou rectangle arrondi dans la délimitation spécifiée.
<code>document.addNewScene()</code>	Ajoute une nouvelle séquence ( <a href="#">Objet Timeline</a> ) après la séquence actuellement sélectionnée, puis fait de cette nouvelle séquence la séquence sélectionnée.
<code>document.addNewText()</code>	Insère un nouveau champ texte vide.
<code>document.align()</code>	Aligne la sélection.



Méthode	Description
<code>document.allowScreens()</code>	Cette méthode doit être utilisée avant la propriété <code>document.screenOutline</code> .
<code>document.arrange()</code>	Organise la sélection sur la scène.
<code>document.breakApart()</code>	Effectue une opération de séparation sur la sélection en cours.
<code>document.canEditSymbol()</code>	Indique si le menu Modifier les symboles et ses fonctionnalités sont activés.
<code>document.canRevert()</code>	Détermine s'il est possible d'utiliser la méthode <code>document.revert()</code> ou <code>fl.revertDocument()</code> .
<code>document.canSaveAVersion()</code>	Détermine si une version du document spécifié peut être enregistrée sur le serveur Version Cue.
<code>document.canTestMovie()</code>	Détermine s'il est possible d'utiliser la méthode <code>document.testMovie()</code> .
<code>document.canTestScene()</code>	Détermine s'il est possible d'utiliser la méthode <code>document.testScene()</code> .
<code>document.changeFilterOrder()</code>	Modifie la position d'index du filtre dans la liste Filtres.
<code>document.clipCopy()</code>	Copie la sélection actuelle du document dans le Presse-papiers.
<code>document.clipCut()</code>	Coupe la sélection actuelle du document et la place dans le Presse-papiers.
<code>document.clipPaste()</code>	Colle le contenu du Presse-papiers dans le document.
<code>document.close()</code>	Ferme le document spécifié.
<code>document.convertLinesToFills()</code>	Convertit des lignes en trame pour les objets sélectionnés.
<code>document.convertToSymbol()</code>	Convertit le ou les éléments sélectionnés dans la scène en un nouveau symbole.
<code>document.crop()</code>	Sur la base de l'objet de dessin sélectionné qui se trouve au-dessus, recadre tous les objets de dessin de niveau inférieur.
<code>document.deleteEnvelope()</code>	Supprime l'enveloppe (cadre de sélection contenant un ou plusieurs objets) de l'objet sélectionné.
<code>document.deletePublishProfile()</code>	Supprime le profil actif, s'il existe plusieurs profils.

Méthode	Description
<code>document.deleteScene()</code>	Supprime la séquence actuelle ( <a href="#">Objet Timeline</a> ) et, si cette séquence n'était pas la dernière, définit la séquence suivante comme objet actif dans le scénario.
<code>document.deleteSelection()</code>	Supprime la sélection actuelle sur la scène.
<code>document.disableAllFilters()</code>	Désactive tous les filtres des objets sélectionnés.
<code>document.disableFilter()</code>	Dans la liste Filtres, désactive le filtre spécifié.
<code>document.disableOtherFilters()</code>	Dans la liste Filtres, désactive tous les filtres à l'exception de celui qui se trouve à la position spécifiée.
<code>document.distribute()</code>	Répartit la sélection.
<code>document.distributeToLayers()</code>	Effectue sur la sélection actuelle une opération de répartition vers les calques ; cette méthode est l'équivalent de l'option de menu Répartir vers les calques.
<code>document.documentHasData()</code>	Recherche dans le document des données persistantes ayant le nom spécifié.
<code>document.duplicatePublishProfile()</code>	Duplique le profil actuellement actif et donne le focus à ce duplicata.
<code>document.duplicateScene()</code>	Crée une copie de la séquence actuellement sélectionnée, en donnant à cette nouvelle séquence un nom unique et en en faisant la séquence active.
<code>document.duplicateSelection()</code>	Duplique la sélection sur la scène.
<code>document.editScene()</code>	Sélectionne la séquence spécifiée pour modification.
<code>document.enableAllFilters()</code>	Active tous les filtres de la liste Filtres pour les objets sélectionnés.
<code>document.enableFilter()</code>	Active le filtre spécifié pour le ou les objets sélectionnés.
<code>document.enterEditMode()</code>	Bascule l'environnement de programmation dans le mode de modification spécifié par le paramètre.
<code>document.exitEditMode()</code>	Quitte le mode Modification de symboles et renvoie le focus au niveau supérieur suivant.

Méthode	Description
<code>document.exportPNG()</code>	Exporte le document dans un ou plusieurs fichiers PNG.
<code>document.exportPublishProfile()</code>	Exporte le profil actif dans un fichier XML.
<code>document.exportSWF()</code>	Exporte le document au format Flash SWF.
<code>document.getAlignToDocument()</code>	Lit la valeur du bouton Sur la scène dans le panneau Aligner.
<code>document.getBlendMode()</code>	Renvoie une chaîne indiquant le mode de fondu pour les objets sélectionnés.
<code>document.getCustomFill()</code>	Lit l'objet de remplissage (Fill) de la forme sélectionnée ou, selon le paramètre, du panneau Outils et de l'inspecteur des propriétés.
<code>document.getCustomStroke()</code>	Renvoie l'objet trait (Stroke) de la forme sélectionnée ou, selon le paramètre, du panneau Outils et de l'inspecteur des propriétés.
<code>document.getDataFromDocument()</code>	Lit la valeur des données spécifiées.
<code>document.getElementProperty()</code>	Lit la propriété <code>Element</code> spécifiée pour la sélection.
<code>document.getElementTextAttr()</code>	Lit la propriété <code>TextAttrs</code> spécifiée pour les objets Texte sélectionnés.
<code>document.getFilters()</code>	Renvoie un tableau contenant la liste des filtres appliqués aux objets actuellement sélectionnés.
<code>document.getMetadata()</code>	Renvoie une chaîne contenant les métadonnées XML associées au document.
<code>document.getMobileSettings()</code>	Renvoie la chaîne transmise à <code>document.setMobileSettings()</code> .
<code>document.getPlayerVersion()</code>	Renvoie une chaîne représentant la version de Flash Player ciblée pour le document spécifié.
<code>document.getSelectionRect()</code>	Lit les coordonnées du rectangle de délimitation de la sélection.
<code>document.getTextString()</code>	Lit le texte sélectionné.
<code>document.getTimeline()</code>	Lit l' <a href="#">Objet Timeline</a> actif dans le document.
<code>document.getTransformationPoint()</code>	Lit l'emplacement du point de transformation de la sélection.
<code>document.group()</code>	Convertit la sélection en un groupe.
<code>document.importFile()</code>	Importe un fichier dans le document.

Méthode	Description
<code>document.importPublishProfile()</code>	Importe un profil à partir d'un fichier.
<code>document.importSWF()</code>	Importe un fichier SWF dans le document.
<code>document.intersect()</code>	Crée un objet de dessin Intersection à partir de tous les objets de dessin sélectionnés.
<code>document.match()</code>	Donne la même taille à tous les objets sélectionnés.
<code>document.mouseClick()</code>	Effectue un clic de souris à partir de l'outil Sélection.
<code>document.mouseDoubleClick()</code>	Effectue un double clic de souris à partir de l'outil Sélection.
<code>document.moveSelectedBezierPointsBy()</code>	Si la sélection contient au moins un trajet comptant au moins un point de courbe de Bézier, cette méthode déplace de la valeur spécifiée tous les points de courbe de Bézier de tous les trajets sélectionnés.
<code>document.moveSelectionBy()</code>	Déplace les objets sélectionnés de la distance spécifiée.
<code>document.optimizeCurves()</code>	Optimise le lissage de la sélection, le cas échéant en autorisant plusieurs passages pour obtenir un lissage optimal équivalent au choix de l'option Modifier > Forme > Optimiser.
<code>document.publish()</code>	Publie le document en fonction des paramètres de publication actuels (Fichier > Paramètres de publication) ; cette méthode est l'équivalent de l'option Fichier > Publier.
<code>document.punch()</code>	Sur la base de l'objet de dessin sélectionné le plus élevé, poinçonne tous les objets de dessin de niveau inférieur.
<code>document.removeAllFilters()</code>	Supprime tous les filtres du ou des objets sélectionnés.
<code>document.removeDataFromDocument()</code>	Supprime les données persistantes associées au document et correspondant au nom spécifié.
<code>document.removeDataFromSelection()</code>	Supprime les données persistantes associées à la sélection et correspondant au nom spécifié.
<code>document.removeFilter()</code>	Supprime le filtre spécifié de la liste Filtres pour les objets sélectionnés.
<code>document.renamePublishProfile()</code>	Renomme le profil actuel.

Méthode	Description
<code>document.renameScene()</code>	Renomme la séquence sélectionnée dans le panneau Séquences.
<code>document.reorderScene()</code>	Place la séquence spécifiée devant une autre séquence spécifiée.
<code>document.resetOvalObject()</code>	Dans l'inspecteur des propriétés, définit toutes les valeurs sur les paramètres par défaut de l'objet Oval.
<code>document.resetRectangleObject()</code>	Dans l'inspecteur des propriétés, définit toutes les valeurs sur les paramètres par défaut de l'objet Rectangle.
<code>document.resetTransformation()</code>	Réinitialise la matrice de transformation ; cette méthode est l'équivalent de l'option de menu Modification > Transformer > Supprimer la transformation.
<code>document.revert()</code>	Restaure la dernière version enregistrée du document spécifié ; cette méthode est l'équivalent de l'option de menu Fichier > Rétablir.
<code>document.revertToLastVersion()</code>	Remplace la version du document spécifié par celle qui est stockée sur le serveur Version Cue et consigne les éventuelles erreurs dans le panneau Sortie.
<code>document.rotateSelection()</code>	Effectue sur la sélection une rotation du nombre de degrés spécifié.
<code>document.save()</code>	Enregistre le document à son emplacement par défaut ; cette méthode est l'équivalent de l'option de menu Fichier > Enregistrer.
<code>document.saveAndCompact()</code>	Enregistre et compresse le fichier ; cette méthode est l'équivalent de l'option de menu Fichier > Enregistrer et compresser.
<code>document.saveAVersion()</code>	Enregistre une version du document spécifié sur le serveur Version Cue.
<code>document.scaleSelection()</code>	Effectue une mise à l'échelle de la sélection de la valeur spécifiée ; cette méthode est l'équivalent de l'outil Transformation libre utilisé pour modifier l'échelle de l'objet.

Méthode	Description
<code>document.selectAll()</code>	Sélectionne tous les éléments figurant sur la scène ; cette méthode est l'équivalent des raccourcis clavier Ctrl+A (Windows) ou Commande+A (Macintosh), ou de l'option de menu Edition > Sélectionner tout.
<code>document.selectNone()</code>	Désélectionne les éléments sélectionnés.
<code>document.setAlignToDocument()</code>	Définit les préférences afin d'activer <code>document.align()</code> , <code>document.distribute()</code> , <code>document.match()</code> et <code>document.space()</code> pour le document ; cette méthode est l'équivalent d'un clic sur le bouton Sur la scène dans le panneau Aligner.
<code>document.setBlendMode()</code>	Définit le mode de fondu pour les objets sélectionnés.
<code>document.setCustomFill()</code>	Définit les paramètres de remplissage pour le panneau Outils, l'inspecteur des propriétés et les formes sélectionnées, le cas échéant.
<code>document.setCustomStroke()</code>	Définit les paramètres de trait pour le panneau Outils, l'inspecteur des propriétés et les formes sélectionnées, le cas échéant.
<code>document.setElementProperty()</code>	Définit la propriété <code>Element</code> spécifiée pour les objets sélectionnés du document.
<code>document.setElementTextAttr()</code>	Définit selon la valeur spécifiée la propriété <code>TextAttrs</code> spécifiée pour les objets texte sélectionnés.
<code>document.setFillColor()</code>	Modifie la couleur de remplissage de la sélection avec la couleur spécifiée.
<code>document.setFilterProperty()</code>	Définit la propriété de filtre spécifiée pour le ou les objets sélectionnés.
<code>document.setFilters()</code>	Applique des filtres aux objets sélectionnés.
<code>document.setInstanceAlpha()</code>	Définit l'opacité de l'occurrence.
<code>document.setInstanceBrightness()</code>	Définit la luminosité de l'occurrence.
<code>document.setInstanceTint()</code>	Définit la teinte de l'occurrence.
<code>document.setMetadata()</code>	Définit les métadonnées XML du document spécifié, en remplacement des métadonnées existantes.

Méthode	Description
<code>document.setMobileSettings()</code>	Définit la valeur d'une chaîne de paramètres XML dans un fichier FLA mobile.
<code>document.setOvalObjectProperty()</code>	Spécifie une valeur pour une propriété spécifiée des objets Oval de type primitif.
<code>document.setPlayerVersion()</code>	Définit la version de Flash Player ciblée par le document spécifié.
<code>document.setRectangleObjectProperty()</code>	Spécifie une valeur pour une propriété spécifiée des objets Rectangle de type primitif.
<code>document.setSelectionBounds()</code>	Déplace et redimensionne la sélection en une seule opération.
<code>document.setSelectionRect()</code>	Trace un cadre de sélection rectangulaire sur la scène, à partir des coordonnées spécifiées.
<code>document.setStroke()</code>	Définit la couleur, la largeur et le style des traits sélectionnés.
<code>document.setStrokeColor()</code>	Modifie la couleur de trait de la sélection avec la couleur spécifiée.
<code>document.setStrokeSize()</code>	Modifie la taille de trait de la sélection avec la taille spécifiée.
<code>document.setStrokeStyle()</code>	Modifie le style de trait de la sélection avec le style spécifié.
<code>document.setTextRectangle()</code>	Modifie le rectangle de délimitation de l'élément texte sélectionné en lui donnant la taille spécifiée.
<code>document.setTextSelection()</code>	Définit la sélection de texte du champ texte sélectionné avec les valeurs spécifiées dans <i>startIndex</i> et <i>endIndex</i> .
<code>document.setTextString()</code>	Insère une chaîne de texte.
<code>document.setTransformationPoint()</code>	Déplace le point de transformation de la sélection.
<code>document.skewSelection()</code>	Effectue sur la sélection une inclinaison de la valeur spécifiée.
<code>document.smoothSelection()</code>	Lisse la courbe de chaque contour de remplissage ou segment incurvé sélectionné.
<code>document.space()</code>	Espace régulièrement les objets sélectionnés.

Méthode	Description
<code>document.straightenSelection()</code>	Redresse les traits sélectionnés ; cette méthode est l'équivalent du bouton Redresser dans le panneau Outils.
<code>document.swapElement()</code>	Intervertit la sélection actuelle avec la sélection spécifiée.
<code>document.swapStrokeAndFill()</code>	Permute les couleurs de trait et de remplissage
<code>document.synchronizeWithHeadVersion()</code>	Synchronise le document spécifié avec la version la plus récente trouvée sur le serveur Version Cue et consigne les éventuelles erreurs dans le panneau Sortie.
<code>document.testMovie()</code>	Exécute une opération Tester l'animation sur le document.
<code>document.testScene()</code>	Exécute une opération Tester la séquence sur la séquence active dans le document.
<code>document.traceBitmap()</code>	Convertit le bitmap sélectionné en graphique vectoriel ; cette méthode est l'équivalent de l'option de menu Modification > Bitmap > Tracer le bitmap.
<code>document.transformSelection()</code>	Effectue une transformation générale de la sélection, en appliquant la matrice passée en argument.
<code>document.unGroup()</code>	Dissocie la sélection.
<code>document.union()</code>	Combine toutes les formes sélectionnées sous forme d'un objet de dessin.
<code>document.unlockAllElements()</code>	Déverrouille tous les éléments verrouillés dans l'image sélectionnée.
<code>document.xmlPanel()</code>	Transmet une boîte de dialogue XMLUI.



# Résumé des propriétés pour l'objet Document

Vous pouvez utiliser les propriétés suivantes avec l'objet Document :

Propriété	Description
<code>document.accName</code>	Chaîne équivalente au champ Nom du panneau Accessibilité.
<code>document.as3AutoDeclare</code>	Valeur booléenne indiquant si les occurrences placées sur la scène sont automatiquement ajoutées aux classes Timeline (scénario) définies par l'utilisateur.
<code>document.as3Dialect</code>	Chaîne décrivant le « dialecte » d'ActionScript 3.0 utilisé dans le document spécifié.
<code>document.as3ExportFrame</code>	Entier spécifiant dans quelle image exporter les classes ActionScript 3.0.
<code>document.as3StrictMode</code>	Valeur booléenne spécifiant si le compilateur ActionScript 3.0 doit compiler avec l'option du mode strict activée ou non.
<code>document.as3WarningsMode</code>	Valeur booléenne spécifiant si le compilateur ActionScript 3.0 doit compiler avec l'option Mode avertissements activée ou non.
<code>document.asVersion</code>	Nombre entier spécifiant quelle version d'ActionScript est en cours d'utilisation dans le fichier spécifié.
<code>document.autoLabel</code>	Valeur booléenne équivalente à la case à cocher Etiquetage auto du panneau Accessibilité.
<code>document.backgroundColor</code>	Chaîne, valeur hexadécimale ou entier représentant la couleur d'arrière-plan.
<code>document.currentPublishProfile</code>	Chaîne spécifiant le nom du profil de publication actif pour le document spécifié.
<code>document.currentTimeline</code>	Nombre entier spécifiant l'index du scénario actif.
<code>document.description</code>	Chaîne équivalente au champ Description du panneau Accessibilité.
<code>document.docClass</code>	Spécifie la classe ActionScript 3.0 de niveau supérieur qui sera associée au document.
<code>document.forceSimple</code>	Valeur booléenne indiquant si les enfants de l'objet spécifié sont accessibles.

Propriété	Description
<code>document.frameRate</code>	Nombre en virgule flottante indiquant le nombre d'images affichées par seconde pendant la lecture du fichier SWF ; cette valeur est 12 par défaut.
<code>document.height</code>	Nombre entier indiquant la hauteur en pixels du document (scène).
<code>document.id</code>	Nombre entier unique (attribué automatiquement) qui identifie un document lors d'une session Flash.
<code>document.library</code>	Propriété en lecture seule ; <a href="#">Objet library</a> d'un document.
<code>document.livePreview</code>	Valeur booléenne indiquant si la fonction Aperçu en direct est activée.
<code>document.name</code>	Propriété en lecture seule ; chaîne représentant le nom d'un document (fichier FLA).
<code>document.path</code>	Propriété en lecture seule ; chaîne représentant le chemin d'accès d'un document (fichier FLA).
<code>document.publishProfiles</code>	Propriété en lecture seule ; tableau des noms des profils de publication du document.
<code>document.screenOutline</code>	Propriété en lecture seule ; <a href="#">Objet ScreenOutline</a> actif du document.
<code>document.selection</code>	Tableau des objets sélectionnés dans le document.
<code>document.silent</code>	Valeur booléenne indiquant si l'objet est accessible.
<code>document.timelines</code>	Propriété en lecture seule ; tableau des objets de scénario (voir <a href="#">Objet Timeline</a> ).
<code>document.viewMatrix</code>	Propriété en lecture seule ; <a href="#">Objet Matrix</a> .
<code>document.width</code>	Nombre entier indiquant la largeur en pixels du document (scène).
<code>document.zoomFactor</code>	Spécifie le pourcentage de zoom de la scène en contexte de création.

# document.accName

## Disponibilité

Flash MX 2004.

## Utilisation

`document.accName`

## Description

Propriété ; chaîne équivalente au champ Nom du panneau Accessibilité. Les logiciels de lecture vocale identifient l'objet en prononçant ce nom.

## Exemple

L'exemple suivant définit le nom d'accessibilité du document comme étant Main Movie :

```
fl.getDocumentDOM().accName = "Main Movie";
```

L'exemple suivant lit le nom d'accessibilité du document :

```
fl.trace(fl.getDocumentDOM().accName);
```

# document.addDataToDocument()

## Disponibilité

Flash MX 2004.

## Utilisation

`document.addDataToDocument(name, type, data)`

## Paramètres

*name* Chaîne spécifiant le nom des données à ajouter.

*type* Chaîne définissant le type de données à ajouter. Les valeurs possibles sont "integer", "integerArray", "double", "doubleArray", "string" et "byteArray".

*data* Valeur à ajouter. Les types valides dépendent du paramètre *type*.

## Valeur renvoyée

Aucune.

## Description

Méthode ; enregistre les données spécifiées avec un document. Les données sont écrites dans le fichier FLA et sont accessibles en JavaScript à la réouverture du fichier.

## Exemple

L'exemple suivant ajoute un entier ayant la valeur 12 au document actif :

```
fl.getDocumentDOM().addDataToDocument("myData", "integer", 12);
```

L'exemple suivant renvoie la valeur des données nommées "myData" et affiche le résultat dans le panneau Sortie :

```
fl.trace(fl.getDocumentDOM().getDataFromDocument("myData"));
```

## Voir aussi

[document.getDataFromDocument\(\)](#), [document.removeDataFromDocument\(\)](#)

# document.addDataToSelection()

## Disponibilité

Flash MX 2004.

## Utilisation

```
document.addDataToSelection(name, type, data)
```

## Paramètres

*name* Chaîne spécifiant le nom des données persistantes.

*type* Définit le type de données. Les valeurs possibles sont les suivantes : "integer", "integerArray", "double", "doubleArray", "string" et "byteArray".

*data* Valeur à ajouter. Les types valides dépendent du paramètre *type*.

## Valeur renvoyée

Aucune.

## Description

Méthode ; enregistre les données spécifiées avec le ou les objets sélectionnés. Les données sont écrites dans le fichier FLA et sont accessibles en JavaScript à la réouverture du fichier. Seuls les symboles et les bitmaps acceptent des données persistantes.

## Exemple

L'exemple suivant ajoute un entier ayant la valeur 12 à l'objet sélectionné :

```
fl.getDocumentDOM().addDataToSelection("myData", "integer", 12);
```

## Voir aussi

[document.removeDataFromSelection\(\)](#)

# document.addFilter()

## Disponibilité

Flash 8.

## Utilisation

```
document.addFilter(filterName)
```

## Paramètres

*filterName* Chaîne indiquant le filtre à ajouter à la liste Filtres et à activer pour les objets sélectionnés. Valeurs gérées : "adjustColorFilter", "bevelFilter", "blurFilter", "dropShadowFilter", "glowFilter", "gradientBevelFilter" et "gradientGlowFilter".

## Valeur renvoyée

Aucune.

## Description

Méthode qui applique un filtre aux objets sélectionnés et place ce dernier à la fin de la liste Filtres.

## Exemple

L'exemple suivant applique un effet néon aux objets sélectionnés :

```
fl.getDocumentDOM().addFilter("glowFilter");
```

## Voir aussi

```
document.changeFilterOrder(), document.disableFilter(),  
document.enableFilter(), document.getFilters(), document.removeFilter(),  
document.setBlendMode(), document.setFilterProperty()
```

# document.addItem()

## Disponibilité

Flash MX 2004.

## Utilisation

`document.addItem(position, item)`

## Paramètres

*position* Un point indiquant les coordonnées *x* et *y* de l'emplacement auquel le composant doit être ajouté. Ce point est le centre pour un symbole ou le coin supérieur gauche pour un bitmap ou une vidéo.

*item* Objet Item spécifiant l'élément à ajouter et la bibliothèque dont il est extrait (consultez la section [Objet Item](#)).

## Valeur renvoyée

Une valeur booléenne : `true` en cas de réussite, `false` sinon.

## Description

Méthode ; ajoute à l'objet Document spécifié un élément provenant d'un document ouvert ou d'une bibliothèque.

## Exemple

L'exemple suivant ajoute le premier élément de la bibliothèque au premier document, à l'emplacement indiqué pour le symbole, le bitmap ou la vidéo sélectionné(e) :

```
var item = fl.documents[0].library.items[0];
fl.documents[0].addItem({x:0,y:0}, item);
```

L'exemple suivant ajoute le symbole `myMovieClip` au document actif à partir de sa propre bibliothèque :

```
var itemIndex = fl.getDocumentDOM().library.findItemIndex("myMovieClip");
var theItem = fl.getDocumentDOM().library.items[itemIndex];
fl.getDocumentDOM().addItem({x:0,y:0}, theItem);
```

L'exemple suivant ajoute le symbole `myMovieClip` du deuxième document du tableau des documents au troisième document de ce tableau :

```
var itemIndex = fl.documents[1].library.findItemIndex("myMovieClip");
var theItem = fl.documents[1].library.items[itemIndex];
fl.documents[2].addItem({x:0,y:0}, theItem);
```

# document.addNewLine()

## Disponibilité

Flash MX 2004.

## Utilisation

`document.addNewLine(startPoint, endpoint)`

## Paramètres

*startPoint* Paire de nombres en virgule flottante indiquant les coordonnées *x* et *y* de départ de la ligne.

*endpoint* Paire de nombres en virgule flottante indiquant les coordonnées *x* et *y* de fin de la ligne.

## Valeur renvoyée

Aucune.

## Description

Méthode ; ajoute un nouveau trajet entre deux points. La méthode utilise les attributs de trait actuels du document et ajoute le trajet dans l'image et le calque actuels. Cette méthode est l'équivalent d'un clic sur l'outil Ligne et du traçage d'une ligne.

## Exemple

L'exemple suivant ajoute une ligne entre le point de début et le point de fin spécifiés :

```
fl.getDocumentDOM().addNewLine({x:216.7, y:122.3}, {x:366.8, y:165.8});
```

# document.addNewOval()

## Disponibilité

Flash MX 2004.

## Utilisation

```
document.addNewOval(boundingRectangle [, bSuppressFill] [, bSuppressStroke])
```

## Paramètres

*boundingRectangle* Rectangle délimitant l'ovale à ajouter. Pour plus d'informations sur le format de *boundingRectangle*, consultez la section [document.addNewRectangle\(\)](#).

*bSuppressFill* Valeur booléenne. Si elle est réglée sur *true*, la méthode crée la forme sans remplissage. La valeur par défaut est *false*. Ce paramètre est facultatif.

*bSuppressStroke* Valeur booléenne. Si elle est *true*, la méthode crée la forme sans trait. La valeur par défaut est *false*. Ce paramètre est facultatif.

## Valeur renvoyée

Aucune.

## Description

Méthode ; ajoute un nouvel objet Oval dans le rectangle de délimitation spécifié. Cette méthode effectue la même opération que l'outil Ovale. La méthode utilise les attributs de trait et de remplissage par défaut actuels du document et ajoute l'ovale dans l'image et le calque actuels. Si *bSuppressFill* est réglé sur *true*, l'ovale est tracé sans remplissage.

Si *bSuppressStroke* est à *true*, l'ovale est tracé sans trait. Si *bSuppressFill* et *bSuppressStroke* sont tous deux réglés sur *true*, la méthode est sans effet.

## Exemple

L'exemple suivant ajoute un ovale de 164 pixels de large sur 178 pixels de haut dans la zone définie par les coordonnées spécifiées :

```
flash.getDocumentDOM().addNewOval({left:72,top:50,right:236,bottom:228});
```

L'exemple suivant trace l'ovale sans remplissage :

```
flash.getDocumentDOM().addNewOval({left:72,top:50,right:236,bottom:228},  
    true);
```

L'exemple suivant trace l'ovale sans trait :

```
flash.getDocumentDOM().addNewOval({left:72,top:50,right:236,bottom:228},  
    false, true);
```



# document.addNewPublishProfile()

## Disponibilité

Flash MX 2004.

## Utilisation

```
document.addNewPublishProfile([profileName])
```

## Paramètres

*profileName* Nom unique du nouveau profil. Si vous ne spécifiez pas de nom, un nom par défaut est utilisé. Ce paramètre est facultatif.

## Valeur renvoyée

Un nombre entier qui est l'index du nouveau profil dans la liste des profils. Renvoie -1 s'il est impossible de créer un nouveau profil.

## Description

Méthode ; ajoute un nouveau profil de publication et l'active.

## Exemple

L'exemple suivant ajoute un nouveau profil de publication avec un nom par défaut, puis affiche le nom du profil dans le panneau Sortie :

```
fl.getDocumentDOM().addNewPublishProfile();  
fl.outputPanel.trace(fl.getDocumentDOM().currentPublishProfile);
```

L'exemple suivant ajoute un nouveau profil de publication avec le nom "my profile" :

```
fl.getDocumentDOM().addNewPublishProfile("my profile");
```

## Voir aussi

[document.deletePublishProfile\(\)](#)

# document.addNewRectangle()

## Disponibilité

Flash MX 2004.

## Utilisation

```
document.addNewRectangle(boundingRectangle, roundness  
[, bSuppressFill[, bSuppressStroke]])
```

## Paramètres

*boundingRectangle* Rectangle qui délimite la zone de création du nouveau rectangle, exprimé au format {left:value1,top:value2,right:value3,bottom:value4}.

Les valeurs *left* et *top* spécifient l'emplacement du coin supérieur gauche (exemple : *left:0,top:0* représente le coin supérieur gauche de la scène), et les valeurs *right* et *bottom* indiquent l'emplacement du coin inférieur droit. De ce fait, la largeur du rectangle correspond à l'écart entre *left* et *right*, et sa hauteur à l'écart entre *top* et *bottom*.

En d'autres termes, les limites du rectangle ne correspondent pas toutes aux valeurs indiquées dans l'inspecteur des propriétés. Les valeurs *left* et *top* correspondent respectivement aux valeurs *X* et *Y* de l'inspecteur des propriétés. Toutefois, les valeurs *right* et *bottom* ne correspondent pas aux valeurs *L* et *H* de l'inspecteur des propriétés. Par exemple, supposons un rectangle possédant les limites suivantes :

```
{left:10,top:10,right:50,bottom:100}
```

Ce rectangle posséderait les valeurs suivantes dans l'inspecteur des propriétés :

*X* = 10, *Y* = 10, *W* = 40, *H* = 90

*roundness* Une valeur entière comprise entre 0 et 999 qui représente le niveau d'arrondi des coins. La valeur est exprimée sous la forme d'un nombre de points. Plus elle est élevée, plus l'arrondi est important.

*bSuppressFill* Valeur booléenne. Si elle est réglée sur *true*, la méthode crée la forme sans remplissage. La valeur par défaut est *false*. Ce paramètre est facultatif.

*bSuppressStroke* Valeur booléenne. Si elle est *true*, la méthode crée le rectangle sans trait. La valeur par défaut est *false*. Ce paramètre est facultatif.

## Valeur renvoyée

Aucune.

## Description

Méthode ; ajoute un nouveau rectangle ou rectangle arrondi dans la délimitation spécifiée. Cette méthode effectue la même opération que l'outil Rectangle. La méthode utilise les attributs de trait et de remplissage par défaut actuels du document et ajoute le rectangle dans l'image et le calque actuels. Si le paramètre *bSuppressFill* est à *true*, le rectangle est tracé sans remplissage. Si le paramètre *bSuppressStroke* est à *true*, le rectangle est tracé sans trait. Si *bSuppressFill* et *bSuppressStroke* sont tous deux réglés sur *true*, la méthode est sans effet.

## Exemple

L'exemple suivant ajoute un rectangle sans angles arrondis aux coordonnées indiquées, de 100 pixels de large sur 100 pixels de haut :

```
flash.getDocumentDOM().addNewRectangle({left:0,top:0,right:100,bottom:100},0);
```

L'exemple suivant ajoute un rectangle, sans angles arrondis ni remplissage, de 100 pixels de large sur 200 pixels de haut :

```
flash.getDocumentDOM().addNewRectangle({left:10,top:10,right:110,bottom:210},0, true);
```

L'exemple suivant ajoute un rectangle, sans angles arrondis ni contour, de 200 pixels de large sur 100 pixels de haut :

```
flash.getDocumentDOM().addNewRectangle({left:20,top:20,right:220,bottom:120},0, false, true);
```

# document.addNewScene()

## Disponibilité

Flash MX 2004.

## Utilisation

```
document.addNewScene([name])
```

## Paramètres

*name* Spécifie le nom de la séquence. Si vous ne spécifiez pas de nom, un nouveau nom de séquence par défaut est créé.

## Valeur renvoyée

Une valeur booléenne : `true` en cas de réussite, `false` sinon.

## Description

Méthode ; ajoute une nouvelle séquence ([Objet Timeline](#)) après la séquence actuellement sélectionnée, puis fait de cette nouvelle séquence la séquence sélectionnée. Si le nom de séquence indiqué existe déjà, la séquence n'est pas ajoutée et la méthode renvoie une erreur.

## Exemple

L'exemple suivant ajoute une nouvelle séquence baptisée `myScene` après la séquence actuelle dans le document actif. La variable `success` est réglée sur `true` après la création de la séquence, sur `false` dans le cas contraire.

```
var success = flash.getDocumentDOM().addNewScene("myScene");
```

L'exemple suivant ajoute une nouvelle séquence en utilisant la convention de nom par défaut. S'il n'existe encore qu'une seule séquence, la nouvelle séquence sera nommée "Scene 2".

```
fl.getDocumentDOM().addNewScene();
```

## document.addNewText()

### Disponibilité

Flash MX 2004 ; paramètre *text* facultatif intégré à Flash CS3 Professional.

### Utilisation

```
document.addNewText(boundingRectangle [, text ])
```

### Paramètres

*boundingRectangle* Spécifie la taille et l'emplacement du champ texte. Pour plus d'informations sur le format de *boundingRectangle*, consultez la section [document.addNewRectangle\(\)](#).

*text* Chaîne facultative qui précise le texte à placer dans le champ. Si vous omettez ce paramètre, la sélection dans le panneau Outils bascule sur l'outil Texte. Par conséquent, si vous ne voulez pas changer d'outil, passez une valeur pour *text*.

### Valeur renvoyée

Aucune.

### Description

Méthode ; insère un nouveau champ texte et, le cas échéant, place un texte dans ce champ. Si vous omettez le paramètre *text*, vous pouvez appeler [document.setTextString\(\)](#) pour remplir le champ texte.

### Exemple

L'exemple suivant crée un nouveau champ texte dans le coin supérieur gauche de la scène et y insère la chaîne de texte "Hello World" :

```
fl.getDocumentDOM().addNewText({left:0, top:0, right:100, bottom:100},  
    "Hello World!" );  
fl.getDocumentDOM().setTextString('Hello World!');
```

### Voir aussi

[document.setTextString\(\)](#)

# document.align()

## Disponibilité

Flash MX 2004.

## Utilisation

```
document.align(alignmode [, bUseDocumentBounds])
```

## Paramètres

*alignmode* Chaîne indiquant comment aligner la sélection. Valeurs gérées : "left" (gauche), "right" (droite), "top" (haut), "bottom" (bas), "vertical center" (centrage vertical) et "horizontal center" (centrage horizontal).

*bUseDocumentBounds* Valeur booléenne. Si elle est true, la méthode aligne les objets sélectionnés sur les limites du document. Sinon, la méthode utilise les limites des objets sélectionnés. La valeur par défaut est false. Ce paramètre est facultatif.

## Valeur renvoyée

Aucune.

## Description

Méthode ; aligne la sélection.

## Exemple

L'exemple suivant aligne les objets à gauche et sur les limites de la scène. Cette commande est équivalente à l'activation de l'option Sur la scène dans le panneau Aligner, puis à un clic sur le bouton Aligner à gauche :

```
fl.getDocumentDOM().align("left", true);
```

## Voir aussi

```
document.distribute(), document.getAlignToDocument(),  
document.setAlignToDocument()
```

# document.allowScreens()

## Disponibilité

Flash MX 2004.

## Utilisation

`document.allowScreens()`

## Paramètres

Aucun.

## Valeur renvoyée

Une valeur booléenne : `true` si `document.screenOutline` peut être utilisé ; `false` sinon.

## Description

Méthode, utilisée avant la propriété `document.screenOutline`. Si la méthode renvoie la valeur `true`, il est possible d'accéder sans risque à la propriété `screenOutline`. Flash affiche une erreur si la propriété `screenOutline` est appelée dans un document ne comportant pas d'écrans.

## Exemple

L'exemple suivant détermine s'il est possible d'utiliser les méthodes `screens` dans le document actif :

```
if(fl.getDocumentDOM().allowScreens()) {  
    fl.trace("screen outline is available.");  
}  
else {  
    fl.trace("whoops, no screens.");  
}
```

## Voir aussi

[document.screenOutline](#)

# document.arrange()

## Disponibilité

Flash MX 2004.

## Utilisation

`document.arrange(arrangeMode)`

## Paramètres

*arrangeMode* Spécifie la direction dans laquelle la sélection doit être déplacée. Les valeurs gérées correspondent à "back", "backward", "forward" et "front". Elle offre les mêmes possibilités que les options équivalentes du menu Modification > Disposer.

## Valeur renvoyée

Aucune.

## Description

Méthode ; organise la sélection sur la scène. Cette méthode s'applique uniquement aux objets qui ne sont pas des formes.

## Exemple

L'exemple suivant déplace la sélection au premier plan :

```
fl.getDocumentDOM().arrange("front");
```

# document.as3AutoDeclare

## Disponibilité

Flash CS3 Professional.

## Utilisation

`document.as3AutoDeclare`

## Description

Propriété ; valeur booléenne indiquant si les occurrences placées sur la scène sont automatiquement ajoutées aux classes Timeline (scénario) définies par l'utilisateur. La valeur par défaut est `true`.

## Exemple

L'exemple suivant précise que les occurrences placées sur la scène dans le document actif doivent être ajoutées manuellement aux classes Timeline (scénario) définies par l'utilisateur.

```
fl.getDocumentDOM().as3AutoDeclare=false;
```

# document.as3Dialect

## Disponibilité

Flash CS3 Professional.

## Utilisation

`document.as3Dialect`

## Description

Propriété ; chaîne décrivant le « dialecte » d'ActionScript 3.0 utilisé dans le document spécifié. La valeur par défaut est "AS3". Si vous souhaitez autoriser des classes de prototype, telles qu'elles ont été admises dans les précédentes spécifications ECMAScript, définissez cette valeur sur "ES".

## Exemple

L'exemple suivant précise que le dialecte utilisé dans le document actif est ECMAScript :

```
fl.getDocumentDOM().as3Dialect="ES";
```

## Voir aussi

[document.asVersion](#)

# document.as3ExportFrame

## Disponibilité

Flash CS3 Professional.

## Utilisation

`document.as3ExportFrame`

## Description

Propriété ; Nombre entier spécifiant dans quelle image exporter les classes ActionScript 3.0. Par défaut, les classes sont exportées dans l'image 1.

## Exemple

L'exemple suivant change l'image dans laquelle les classes sont exportées, l'image 5 remplace l'image par défaut 1.

```
var myDocument = fl.getDocumentDOM();
fl.outputPanel.trace("'Export classes in frame:' value before modification
is " + myDocument.as3ExportFrame);
myDocument.as3ExportFrame = 5;
fl.outputPanel.trace("'Export classes in frame:' value after modification
is " + myDocument.as3ExportFrame);
```



# document.as3StrictMode

## Disponibilité

Flash CS3 Professional.

## Utilisation

`document.as3StrictMode`

## Description

Propriété ; valeur booléenne spécifiant si le compilateur ActionScript 3.0 doit compiler avec l'option Mode strict activée (`true`) ou désactivée (`false`). Ce mode déclenche des avertissements devant être signalés comme erreurs, la compilation échouera donc si de telles erreurs existent. La valeur par défaut est `true`.

## Exemple

L'exemple suivant désactive l'option Mode strict du compilateur.

```
var myDocument = fl.getDocumentDOM();
fl.outputPanel.trace("Strict Mode value before modification is " +
    myDocument.as3StrictMode);
myDocument.as3StrictMode = false;
fl.outputPanel.trace("Strict Mode value after modification is " +
    myDocument.as3StrictMode);
```

## Voir aussi

[document.as3WarningsMode](#)

# document.as3WarningsMode

## Disponibilité

Flash CS3 Professional.

## Utilisation

`document.as3WarningsMode`

## Description

Propriété ; valeur booléenne spécifiant si le compilateur ActionScript 3.0 doit compiler avec l'option Mode avertissements activée (`true`) ou désactivée (`false`). Le mode avertissements déclenche des alarmes supplémentaires devant être signalées, celles-ci sont utiles lors de la mise à jour du code ActionScript 2.0 vers ActionScript 3.0 pour repérer d'éventuelles incompatibilités. La valeur par défaut est `true`.

## Exemple

L'exemple suivant désactive l'option Mode avertissements du compilateur.

```
var myDocument = fl.getDocumentDOM();
fl.outputPanel.trace("Warnings Mode value before modification is " +
    myDocument.as3WarningsMode);
myDocument.as3WarningsMode = false;
fl.outputPanel.trace("Warnings Mode value after modification is " +
    myDocument.as3WarningsMode);
```

## Voir aussi

[document.as3StrictMode](#)

# document.asVersion

## Disponibilité

Flash CS3 Professional.

## Utilisation

`document.asVersion`

## Description

Propriété ; nombre entier spécifiant quelle version d'ActionScript est en cours d'utilisation dans le document spécifié. Les valeurs gérées sont 1, 2 et 3.

Pour déterminer la version de Flash Player ciblée pour le document spécifié, utilisez [document.getPlayerVersion\(\)](#). Cette méthode renvoie une chaîne, elle est donc utilisable par les lecteurs Flash Lite.

## Exemple

L'exemple suivant définit la version d'ActionScript dans le document actif sur ActionScript 2.0 si elle est actuellement réglée sur ActionScript 1.0.

```
if(fl.getDocumentDOM().asVersion == 1){
    fl.getDocumentDOM().asVersion = 2;
}
```

## Voir aussi

[document.as3Dialect](#), [document.getPlayerVersion\(\)](#)

# document.autoLabel

## Disponibilité

Flash MX 2004.

## Utilisation

`document.autoLabel`

## Description

Propriété ; valeur booléenne équivalente à la case à cocher Etiquetage auto du panneau Accessibilité. Cette propriété permet d'indiquer à Flash d'étiqueter automatiquement les objets sur la scène avec le texte qui leur est associé.

## Exemple

L'exemple suivant lit la valeur de la propriété `autoLabel` et affiche le résultat dans le panneau Sortie :

```
var isAutoLabel = fl.getDocumentDOM().autoLabel;  
fl.trace(isAutoLabel);
```

L'exemple suivant règle la propriété `autoLabel` sur `true`, indiquant ainsi à Flash que les objets présents sur la scène doivent être étiquetés automatiquement :

```
fl.getDocumentDOM().autoLabel = true;
```

# document.backgroundColor

## Disponibilité

Flash MX 2004.

## Utilisation

`document.backgroundColor`

## Description

Propriété ; couleur de l'arrière-plan, exprimée dans l'un des formats suivants :

- chaîne au format `"#RRVBBB"` ou `"#RRVVBBAA"` ;
- nombre hexadécimal au format `0xRRVBBB` ;
- entier représentant l'équivalent décimal d'un nombre hexadécimal.

## Exemple

L'exemple suivant définit la couleur d'arrière-plan comme Noir :

```
fl.getDocumentDOM().backgroundColor = '#000000';
```

# document.breakApart()

## Disponibilité

Flash MX 2004.

## Utilisation

```
document.breakApart()
```

## Paramètres

Aucun.

## Valeur renvoyée

Aucune.

## Description

Méthode ; effectue une opération de séparation sur la sélection en cours.

## Exemple

L'exemple suivant sépare la sélection :

```
fl.getDocumentDOM().breakApart();
```

# document.canEditSymbol()

## Disponibilité

Flash MX 2004.

## Utilisation

```
document.canEditSymbol()
```

## Paramètres

Aucun.

## Valeur renvoyée

Une valeur booléenne : `true` si le menu et la fonctionnalité Modifier les symboles sont disponibles, `false` sinon.

## Description

Méthode ; indique si le menu Modifier les symboles et ses fonctionnalités sont activés. Cette méthode n'indique pas s'il est possible de modifier la sélection. Elle ne doit pas être utilisée pour tester si l'appel de `fl.getDocumentDOM().enterEditMode()` est possible.

## Exemple

L'exemple suivant affiche dans le panneau Sortie l'état du menu Modifier les symboles et ses fonctionnalités :

```
fl.trace("fl.getDocumentDOM().canEditSymbol() returns: " +  
    fl.getDocumentDOM().canEditSymbol());
```

# document.canRevert()

## Disponibilité

Flash MX 2004.

## Utilisation

```
document.canRevert()
```

## Paramètres

Aucun.

## Valeur renvoyée

Une valeur booléenne : `true` s'il est possible d'utiliser les méthodes `document.revert()` ou `fl.revertDocument()`, `false` sinon.

## Description

Méthode ; détermine s'il est possible d'utiliser la méthode `document.revert()` ou `fl.revertDocument()`.

## Exemple

L'exemple suivant vérifie s'il est possible de restaurer le document actif avec sa dernière version enregistrée. Dans l'affirmative, `fl.getDocumentDOM().revert()` rétablit la dernière version enregistrée.

```
if(fl.getDocumentDOM().canRevert()){  
    fl.getDocumentDOM().revert();  
}
```

# document.canSaveAVersion()

## Disponibilité

Flash CS3 Professional.

## Utilisation

```
document.canSaveAVersion()
```

## Paramètres

Aucun.

## Valeur renvoyée

Une valeur booléenne `true` si une version du fichier peut être enregistrée sur le serveur Version Cue, sinon `false`.

## Description

Méthode ; détermine si une version du document spécifié peut être enregistrée sur le serveur Version Cue.

## Exemple

L'exemple suivant vérifie si `document.saveAVersion()` est utilisable. Si c'est le cas, la méthode est ensuite appelée.

```
if(fl.getDocumentDOM().canSaveAVersion()){  
    fl.getDocumentDOM().saveAVersion;  
}
```

## Voir aussi

[document.revertToLastVersion\(\)](#), [document.saveAVersion\(\)](#)

# document.canTestMovie()

## Disponibilité

Flash MX 2004.

## Utilisation

```
document.canTestMovie()
```

## Paramètres

Aucun.

## Valeur renvoyée

Une valeur booléenne : `true` s'il est possible d'utiliser la méthode [document.testMovie\(\)](#). `false` sinon.

## Description

Méthode ; détermine s'il est possible d'utiliser la méthode [document.testMovie\(\)](#).

## Exemple

L'exemple suivant teste si `fl.getDocumentDOM().testMovie()` peut être utilisé. Si c'est le cas, la méthode est ensuite appelée.

```
if(fl.getDocumentDOM().canTestMovie()){
    fl.getDocumentDOM().testMovie();
}
```

## Voir aussi

[document.canTestScene\(\)](#), [document.testScene\(\)](#)

# document.canTestScene()

## Disponibilité

Flash MX 2004.

## Utilisation

`document.canTestScene()`

## Paramètres

Aucun.

## Valeur renvoyée

Une valeur booléenne : `true` s'il est possible d'utiliser la méthode `document.testScene()`, `false` sinon.

## Description

Méthode ; détermine s'il est possible d'utiliser la méthode [document.testScene\(\)](#).

## Exemple

L'exemple suivant vérifie d'abord si `fl.getDocumentDOM().testScene()` est utilisable. Si c'est le cas, la méthode est ensuite appelée.

```
if(fl.getDocumentDOM().canTestScene()){
    fl.getDocumentDOM().testScene();
}
```

## Voir aussi

[document.canTestMovie\(\)](#), [document.testMovie\(\)](#)

# document.changeFilterOrder()

## Disponibilité

Flash 8.

## Utilisation

```
document.changeFilterOrder(oldIndex, newIndex)
```

## Paramètres

*oldIndex* Nombre entier représentant la position d'index (en base 0) du filtre à déplacer au sein de la liste Filtres.

*newIndex* Nombre entier représentant la nouvelle position du filtre dans la liste.

## Valeur renvoyée

Aucune.

## Description

Méthode ; modifie la position d'index du filtre dans la liste Filtres. Tout filtre qui suit ou précède *newIndex* est décalé en conséquence vers le haut ou vers le bas. Par exemple, avec les filtres indiqués ci-dessous, après la commande

```
fl.getDocumentDOM().changeFilterOrder(3, 0) les filtres sont réorganisés comme suit :
```

**Avant :** blurFilter, dropShadowFilter, glowFilter, gradientBevelFilter

**Après :** gradientBevelFilter, blurFilter, dropShadowFilter, glowFilter

Si vous appelez alors la commande `fl.getDocumentDOM().changeFilterOrder(0, 2)`, les filtres sont organisés comme suit :

**Avant :** gradientBevelFilter, blurFilter, dropShadowFilter, glowFilter

**Après :** blurFilter, dropShadowFilter, gradientBevelFilter, glowFilter

## Exemple

L'exemple suivant déplace le filtre qui est actuellement en seconde position dans la liste Filtres et le fait passer en première position :

```
fl.getDocumentDOM().changeFilterOrder(1,0);
```

## Voir aussi

[document.addFilter\(\)](#), [document.disableFilter\(\)](#), [document.enableFilter\(\)](#), [document.getFilters\(\)](#), [document.removeFilter\(\)](#), [Objet Filter](#)



# document.clipCopy()

## Disponibilité

Flash MX 2004.

## Utilisation

`document.clipCopy()`

## Paramètres

Aucun.

## Valeur renvoyée

Aucune.

## Description

Méthode ; copie la sélection actuelle du document dans le Presse-papiers.

Pour copier une chaîne dans le Presse-papiers, utilisez `fl.clipCopyString()`.

## Exemple

L'exemple suivant copie la sélection actuelle du document dans le Presse-papiers :

```
fl.getDocumentDOM().clipCopy();
```

## Voir aussi

`document.clipCut()`, `document.clipPaste()`

# document.clipCut()

## Disponibilité

Flash MX 2004.

## Utilisation

`document.clipCut()`

## Paramètres

Aucun.

## Valeur renvoyée

Aucune.

## Description

Méthode ; coupe la sélection actuelle du document et la place dans le Presse-papiers.

## Exemple

L'exemple suivant coupe la sélection actuelle du document et la place dans le Presse-papiers :

```
fl.getDocumentDOM().clipCut();
```

## Voir aussi

[document.clipCopy\(\)](#), [document.clipPaste\(\)](#), [fl.clipCopyString\(\)](#)

# document.clipPaste()

## Disponibilité

Flash MX 2004.

## Utilisation

```
document.clipPaste([bInPlace])
```

## Paramètres

*bInPlace* Valeur booléenne. Si elle est `true`, la méthode effectue une opération de Coller en place. La valeur par défaut est `false`. La méthode effectue dans ce cas une opération Coller dans le centre du document. Ce paramètre est facultatif.

## Valeur renvoyée

Aucune.

## Description

Méthode ; colle le contenu du Presse-papiers dans le document.

## Exemple

L'exemple suivant colle le contenu du Presse-papiers au centre du document :

```
fl.getDocumentDOM().clipPaste();
```

L'exemple suivant colle le contenu du Presse-papiers à sa place dans le document actif :

```
fl.getDocumentDOM().clipPaste(true);
```

## Voir aussi

[document.clipCopy\(\)](#), [document.clipCut\(\)](#), [fl.clipCopyString\(\)](#)

# document.close()

## Disponibilité

Flash MX 2004.

## Utilisation

```
document.close([bPromptToSaveChanges])
```

## Paramètres

*bPromptToSaveChanges* Valeur booléenne. Si elle est *true*, la méthode affiche une boîte de dialogue pour inviter l'utilisateur à enregistrer les modifications du document.

Si *bPromptToSaveChanges* est à *false*, l'utilisateur n'est pas invité à enregistrer les éventuelles modifications. La valeur par défaut est *true*. Ce paramètre est facultatif.

## Valeur renvoyée

Aucune.

## Description

Méthode ; ferme le document spécifié.

## Exemple

L'exemple suivant ferme le document actif et affiche une boîte de dialogue pour inviter l'utilisateur à enregistrer les modifications :

```
fl.getDocumentDOM().close();
```

L'exemple suivant ferme le document sans enregistrer les modifications :

```
fl.getDocumentDOM().close(false);
```

# document.convertLinesToFills()

## Disponibilité

Flash MX 2004.

## Utilisation

```
document.convertLinesToFills()
```

## Paramètres

Aucun.

## Valeur renvoyée

Aucune.

## Description

Méthode ; convertit des lignes en trame pour les objets sélectionnés.

## Exemple

L'exemple suivant convertit en trames les lignes sélectionnées :

```
fl.getDocumentDOM().convertLinesToFills();
```

# document.convertToSymbol()

## Disponibilité

Flash MX 2004.

## Utilisation

```
document.convertToSymbol(type, name, registrationPoint)
```

## Paramètres

*type* Chaîne spécifiant le type de symbole à créer. Les valeurs possibles sont "button" (bouton), "movie clip" (clip) et "graphic".

*nom* Chaîne spécifiant le nom du nouveau symbole. Ce nom doit être unique. Il est possible de transmettre une chaîne vide. La méthode crée alors un nom de symbole unique.

*registration point* Spécifie le point d'alignement, qui représente les coordonnées 0,0 du symbole. Les valeurs acceptables sont les suivantes : "top left" (haut gauche), "top center" (haut centre), "top right" (haut droite), "center left" (centre gauche), "center" (centre), "center right" (centre droite), "bottom left" (bas gauche), "bottom center" (bas centre) et "bottom right" (bas droite).

## Valeur renvoyée

Un objet pour le nouveau symbole, ou null s'il est impossible de créer le symbole.

## Description

Méthode ; convertit le ou les éléments sélectionnés dans la scène en un nouveau symbole.

Pour toutes informations sur la définition des liaisons et des propriétés des actifs partagés pour un symbole, consultez la section [Objet Item](#).

## Exemple

Les exemples suivants créent un symbole de clip avec le nom spécifié, un symbole de bouton avec le nom spécifié, et un symbole de clip avec un nom par défaut :

```
newMc = fl.getDocumentDOM().convertToSymbol("movie clip", "mcSymbolName",  
    "top left");  
newButton = fl.getDocumentDOM().convertToSymbol("button", "btnSymbolName",  
    "bottom right");  
newClipWithDefaultName = fl.getDocumentDOM().convertToSymbol("movie clip",  
    "", "top left");
```

# document.crop()

## Disponibilité

Flash 8.

## Utilisation

```
document.crop()
```

## Paramètres

Aucun.

## Valeur renvoyée

Valeur booléenne : `true` en cas de réussite, sinon `false`.

## Description

Méthode ; sur la base de l'objet de dessin sélectionné le plus élevé, recadre tous les objets de dessin de niveau inférieur. La méthode renvoie `false` s'il n'existe pas d'objets de dessin sélectionnés, ou si l'un des éléments sélectionnés n'est pas un objet de dessin.

## Exemple

L'exemple suivant recadre les objets actuellement sélectionnés :

```
fl.getDocumentDOM().crop();
```

## Voir aussi

[document.deleteEnvelope\(\)](#), [document.intersect\(\)](#), [document.punch\(\)](#),  
[document.union\(\)](#), [shape.isDrawingObject](#)

# document.currentPublishProfile

## Disponibilité

Flash MX 2004.

## Utilisation

`document.currentPublishProfile`

## Description

Propriété ; chaîne spécifiant le nom du profil de publication actif pour le document spécifié.

## Exemple

L'exemple suivant ajoute un nouveau profil de publication avec le nom par défaut, puis affiche le nom du profil dans le panneau Sortie :

```
fl.getDocumentDOM().addNewPublishProfile();  
fl.outputPanel.trace(fl.getDocumentDOM().currentPublishProfile);
```

L'exemple suivant remplace le profil sélectionné par le profil par défaut ("Default") :

```
fl.getDocumentDOM().currentPublishProfile = "Default";
```

# document.currentTimeline

## Disponibilité

Flash MX 2004.

## Utilisation

`document.currentTimeline`

## Description

Propriété ; nombre entier spécifiant l'index du scénario actif. Il est possible de définir le scénario actif en changeant la valeur de cette propriété ; l'effet est pratiquement identique à un appel de `document.editScene()`. La seule différence réside dans le fait qu'aucun message d'erreur n'est affiché si l'index du scénario n'est pas valide. La propriété n'est pas modifiée, ce qui provoque un échec silencieux.

## Exemple

L'exemple suivant affiche l'index du scénario actif :

```
var myCurrentTL = fl.getDocumentDOM().currentTimeline;  
fl.trace("The index of the current timeline is: "+ myCurrentTL);
```

L'exemple suivant change le scénario actif en remplaçant le scénario principal par une séquence nommée "myScene".

```
var i = 0;
var curTimelines = fl.getDocumentDOM().timelines;
while(i < fl.getDocumentDOM().timelines.length){
    if(curTimelines[i].name == "myScene"){
        fl.getDocumentDOM().currentTimeline = i;
    }
    ++i;
}
```

### Voir aussi

[document.getTimeline\(\)](#)

## document.deleteEnvelope()

### Disponibilité

Flash 8.

### Utilisation

```
document.deleteEnvelope();
```

### Paramètres

Aucun.

### Valeur renvoyée

Une valeur booléenne : `true` en cas de réussite, `false` sinon.

### Description

Méthode ; supprime l'enveloppe (cadre de sélection contenant un ou plusieurs objets) de l'objet sélectionné.

### Exemple

L'exemple suivant supprime l'enveloppe des objets sélectionnés :

```
fl.getDocumentDOM().deleteEnvelope();
```

### Voir aussi

[document.crop\(\)](#), [document.intersect\(\)](#), [document.punch\(\)](#), [document.union\(\)](#), [shape.isDrawingObject](#)

# document.deletePublishProfile()

## Disponibilité

Flash MX 2004.

## Utilisation

```
document.deletePublishProfile()
```

## Paramètres

Aucun.

## Valeur renvoyée

Un nombre entier qui représente l'index du nouveau profil actif. Si aucun nouveau profil n'est disponible, la méthode conserve le profil actuel et renvoie son index.

## Description

Méthode ; supprime le profil actif, s'il existe plusieurs profils. Il doit rester au moins un profil.

## Exemple

L'exemple suivant supprime le profil actif s'il existe plusieurs profils, et affiche l'index du nouveau profil actif :

```
alert(fl.getDocumentDOM().deletePublishProfile());
```

## Voir aussi

[document.addNewPublishProfile\(\)](#)

# document.deleteScene()

## Disponibilité

Flash MX 2004.

## Utilisation

```
document.deleteScene()
```

## Paramètres

Aucun.

## Valeur renvoyée

Une valeur booléenne : `true` en cas de réussite, `false` sinon.



## Description

Méthode ; supprime la séquence actuelle ([Objet Timeline](#)) et, si cette séquence n'était pas la dernière, définit la séquence suivante comme objet actif dans le scénario. Si la séquence supprimée était la dernière, le premier objet est défini comme objet actif dans le scénario. S'il n'existe qu'un seul objet Timeline (séquence), la méthode renvoie la valeur `false`.

## Exemple

En supposant qu'il existe trois séquences (Scene0, Scene1 et Scene2) dans le document actif, l'exemple suivant active la séquence Scene2 puis la supprime :

```
fl.getDocumentDOM().editScene(2);  
var success = fl.getDocumentDOM().deleteScene();
```

# document.deleteSelection()

## Disponibilité

Flash MX 2004.

## Utilisation

```
document.deleteSelection()
```

## Paramètres

Aucun.

## Valeur renvoyée

Aucune.

## Description

Méthode ; supprime la sélection actuelle sur la scène. Un message d'erreur est affiché s'il n'existe pas de sélection.

## Exemple

L'exemple suivant supprime la sélection du document :

```
fl.getDocumentDOM().deleteSelection();
```

# document.description

## Disponibilité

Flash MX 2004.

## Utilisation

`document.description`

## Description

Propriété ; chaîne équivalente au champ Description du panneau Accessibilité.  
Cette description est lue par les logiciels de lecture vocale.

## Exemple

L'exemple suivant définit la description du document :

```
fl.getDocumentDOM().description= "This is the main movie";
```

L'exemple suivant lit la description du document et l'affiche dans le panneau Sortie :

```
fl.trace(fl.getDocumentDOM().description);
```

# document.disableAllFilters()

## Disponibilité

Flash 8.

## Utilisation

`document.disableAllFilters()`

## Paramètres

Aucun.

## Valeur renvoyée

Aucune.

## Description

Méthode ; désactive tous les filtres des objets sélectionnés.

## Exemple

L'exemple suivant désactive tous les filtres des objets sélectionnés :

```
fl.getDocumentDOM().disableAllFilters();
```

### Voir aussi

[document.addFilter\(\)](#), [document.changeFilterOrder\(\)](#), [document.disableFilter\(\)](#), [document.disableOtherFilters\(\)](#), [document.enableAllFilters\(\)](#), [document.getFilters\(\)](#), [document.removeAllFilters\(\)](#), [Objet Filter](#)

## document.disableFilter()

### Disponibilité

Flash 8.

### Utilisation

`document.disableFilter(filterIndex)`

### Paramètres

*filterIndex* Nombre entier représentant l'index (en base 0) du filtre dans la liste.

### Valeur renvoyée

Aucune.

### Description

Méthode ; dans la liste Filtres, désactive le filtre spécifié.

### Exemple

L'exemple suivant désactive des objets sélectionnés les premier et troisième filtres (valeur d'index 0 et 2) de la liste Filtres.

```
fl.getDocumentDOM().disableFilter(0);  
fl.getDocumentDOM().disableFilter(2);
```

### Voir aussi

[document.addFilter\(\)](#), [document.changeFilterOrder\(\)](#), [document.disableAllFilters\(\)](#), [document.disableOtherFilters\(\)](#), [document.enableFilter\(\)](#), [document.getFilters\(\)](#), [document.removeFilter\(\)](#), [Objet Filter](#)

# document.disableOtherFilters()

## Disponibilité

Flash 8.

## Utilisation

```
document.disableOtherFilters(enabledFilterIndex)
```

## Paramètres

*enabledFilterIndex* Nombre entier représentant l'index (en base 0) du filtre qui doit rester activé après que tous les autres filtres ont été désactivés.

## Valeur renvoyée

Aucune.

## Description

Méthode ; dans la liste Filtres, désactive tous les filtres à l'exception de celui qui se trouve à la position spécifiée.

## Exemple

L'exemple suivant désactive tous les filtres, à l'exception du deuxième filtre de la liste (index 1).

```
fl.getDocumentDom().disableOtherFilters(1);
```

## Voir aussi

[document.addFilter\(\)](#), [document.changeFilterOrder\(\)](#),  
[document.disableAllFilters\(\)](#), [document.disableFilter\(\)](#),  
[document.enableFilter\(\)](#), [document.getFilters\(\)](#), [document.removeFilter\(\)](#), [Objet Filter](#)

# document.distribute()

## Disponibilité

Flash MX 2004.

## Utilisation

```
document.distribute(distributemode [, bUseDocumentBounds])
```

## Paramètres

*distributemode* Chaîne spécifiant où répartir l'objet sélectionné. Valeurs gérées : "left edge", "horizontal center", "right edge", "top edge", "vertical center" et "bottom edge".

*bUseDocumentBounds* Valeur booléenne. Si elle est true, la méthode répartit les objets sélectionnés sur les limites du document. Sinon, la méthode utilise les limites des objets sélectionnés. La valeur par défaut est false.

## Valeur renvoyée

Aucune.

## Description

Méthode ; répartit la sélection.

## Exemple

L'exemple suivant répartit les objets sélectionnés sur le bord supérieur :

```
fl.getDocumentDOM().distribute("top edge");
```

L'exemple suivant répartit les objets sélectionnés sur le bord supérieur et définit explicitement le paramètre *bUseDocumentBounds* :

```
fl.getDocumentDOM().distribute("top edge", false);
```

L'exemple suivant répartit les objets sélectionnés sur leurs bords supérieurs, en les alignant sur les limites du document :

```
fl.getDocumentDOM().distribute("top edge", true);
```

## Voir aussi

[document.getAlignToDocument\(\)](#), [document.setAlignToDocument\(\)](#)

# document.distributeToLayers()

## Disponibilité

Flash MX 2004.

## Utilisation

```
document.distributeToLayers()
```

## Paramètres

Aucun.

## Valeur renvoyée

Aucune.

## Description

Méthode ; effectue sur la sélection actuelle une opération de répartition vers les calques ; cette méthode est l'équivalent de l'option de menu Répartir vers les calques. Un message d'erreur est affiché s'il n'existe pas de sélection.

## Exemple

L'exemple suivant répartit la sélection dans les calques :

```
fl.getDocumentDOM().distributeToLayers();
```

# document.docClass

## Disponibilité

Flash CS3 Professional.

## Utilisation

```
document.docClass
```

## Description

Propriété ; chaîne spécifiant la classe ActionScript 3.0 de niveau supérieur qui sera associée au document. Si le document n'est pas configuré pour utiliser ActionScript 3.0, cette propriété est ignorée.

## Exemple

L'exemple suivant précise que la classe `ActionScript 3.0` associée au document est `com.mycompany.ManagerClass` et qu'elle est définie dans `com/mycompany/ManagerClass` comme :

```
var myDocument = fl.getDocumentDOM();
// définir la propriété
myDocument.docClass = "com.mycompany.ManagerClass";
// lire la propriété
fl.outputPanel.trace("document.docClass has been set to " +
    myDocument.docClass);
```

## Voir aussi

[item.linkageBaseClass](#)

# document.documentHasData()

## Disponibilité

Flash MX 2004.

## Utilisation

`document.documentHasData(name)`

## Paramètres

*name* Chaîne spécifiant le nom des données à vérifier.

## Valeur renvoyée

Une valeur booléenne : `true` si le document comporte des données persistantes, `false` sinon.

## Description

Méthode ; recherche dans le document des données persistantes ayant le nom spécifié.

## Exemple

L'exemple suivant recherche dans le document des données persistantes ayant le nom `"myData"`:

```
var hasData = fl.getDocumentDOM().documentHasData("myData");
```

## Voir aussi

[document.addDataToDocument\(\)](#), [document.getDataFromDocument\(\)](#),  
[document.removeDataFromDocument\(\)](#)

# document.duplicatePublishProfile()

## Disponibilité

Flash MX 2004.

## Utilisation

```
document.duplicatePublishProfile([profileName])
```

## Paramètres

*profileName* Chaîne spécifiant le nom unique du profil dupliqué. Si vous ne spécifiez pas de nom, le nom par défaut est utilisé. Ce paramètre est facultatif.

## Valeur renvoyée

Un nombre entier qui est l'index du nouveau profil dans la liste des profils. Renvoie -1 s'il est impossible de dupliquer le profil.

## Description

Méthode ; duplique le profil actuellement actif et donne le focus à ce duplicata.

## Exemple

L'exemple suivant duplique le profil actuellement actif et affiche l'index du nouveau profil dans le panneau Sortie :

```
fl.trace(fl.getDocumentDOM().duplicatePublishProfile("dup profile"));
```

# document.duplicateScene()

## Disponibilité

Flash MX 2004.

## Utilisation

```
document.duplicateScene()
```

## Paramètres

Aucun.

## Valeur renvoyée

Une valeur booléenne : `true` en cas de réussite, `false` sinon.

## Description

Méthode ; crée une copie de la séquence actuellement sélectionnée, en donnant à cette nouvelle séquence un nom unique et en en faisant la séquence active.



### Exemple

L'exemple suivant duplique la seconde séquence du document actif :

```
fl.getDocumentDOM().editScene(1); //Définir la séquence du milieu
                                //comme séquence active.
var success = fl.getDocumentDOM().duplicateScene();
```

## document.duplicateSelection()

### Disponibilité

Flash MX 2004.

### Utilisation

```
document.duplicateSelection()
```

### Paramètres

Aucun.

### Valeur renvoyée

Aucune.

### Description

Méthode ; duplique la sélection sur la scène.

### Exemple

L'exemple suivant duplique la sélection. Cette opération est semblable à un appui sur la touche Alt suivi d'un clic et d'un glisser-déposer de l'objet :

```
fl.getDocumentDOM().duplicateSelection();
```

## document.editScene()

### Disponibilité

Flash MX 2004.

### Utilisation

```
document.editScene(index)
```

### Paramètres

*index* Nombre entier en base zéro indiquant la séquence à modifier.

### Valeur renvoyée

Aucune.

## Description

Méthode ; sélectionne la séquence spécifiée pour modification.

## Exemple

En supposant qu'il existe trois séquences (Scene0, Scene1 et Scene2) dans le document actif, l'exemple suivant active la séquence Scene2 puis la supprime :

```
fl.getDocumentDOM().editScene(2);  
fl.getDocumentDOM().deleteScene();
```

# document.enableAllFilters()

## Disponibilité

Flash 8.

## Utilisation

```
document.enableAllFilters()
```

## Paramètres

Aucun.

## Valeur renvoyée

Aucune.

## Description

Méthode ; active tous les filtres de la liste Filtres pour les objets sélectionnés.

## Exemple

L'exemple suivant active tous les filtres de la liste Filtres pour les objets sélectionnés :

```
fl.getDocumentDOM().enableAllFilters()
```

## Voir aussi

[document.addFilter\(\)](#), [document.changeFilterOrder\(\)](#),  
[document.disableAllFilters\(\)](#), [document.enableFilter\(\)](#), [document.getFilters\(\)](#),  
[document.removeAllFilters\(\)](#), [Objet Filter](#)

# document.enableFilter()

## Disponibilité

Flash 8.

## Utilisation

```
document.enableFilter(filterIndex)
```

## Paramètres

*filterIndex* Nombre entier indiquant l'index (en base 0) dans la liste Filtres du filtre à activer.

## Valeur renvoyée

Aucune.

## Description

Méthode ; active le filtre spécifié pour le ou les objets sélectionnés.

## Exemple

L'exemple suivant active le second filtre des objets sélectionnés :

```
fl.getDocumentDOM().enableFilter(1);
```

## Voir aussi

[document.addFilter\(\)](#), [document.changeFilterOrder\(\)](#), [document.disableFilter\(\)](#), [document.enableAllFilters\(\)](#), [document.getFilters\(\)](#), [document.removeFilter\(\)](#), [Objet Filter](#)

# document.enterEditMode()

## Disponibilité

Flash MX 2004.

## Utilisation

```
document.enterEditMode([editMode])
```

## Paramètres

*editMode* Chaîne qui spécifie le mode de modification. Les valeurs gérées sont "inPlace" ou "newWindow". Si aucun paramètre n'est spécifié, le mode par défaut est le mode de modification de symbole. Ce paramètre est facultatif.

## Valeur renvoyée

Aucune.

## Description

Méthode ; bascule l'environnement de programmation dans le mode de modification spécifié par le paramètre. Si aucun paramètre n'est spécifié, le mode par défaut est le mode Modification de symbole. Le résultat est semblable à un clic droit sur le symbole pour afficher le menu contextuel, puis au choix de l'option Modifier.

## Exemple

L'exemple suivant met Flash en mode Modification en place pour le symbole sélectionné :

```
fl.getDocumentDOM().enterEditMode('inPlace');
```

L'exemple suivant met Flash en mode Modifier dans une nouvelle fenêtre pour le symbole sélectionné :

```
fl.getDocumentDOM().enterEditMode('newWindow');
```

## Voir aussi

[document.exitEditMode\(\)](#)

# document.exitEditMode()

## Disponibilité

Flash MX 2004.

## Utilisation

```
document.exitEditMode()
```

## Paramètres

Aucun.

## Valeur renvoyée

Aucune.

## Description

Méthode ; quitte le mode Modification de symboles et renvoie le focus au niveau supérieur suivant. Par exemple, si vous modifiez un symbole inséré dans un autre symbole, cette méthode remonte d'un niveau à partir du symbole en cours de modification, soit au niveau du symbole parent.

## Exemple

L'exemple suivant quitte le mode Modification de symbole :

```
fl.getDocumentDOM().exitEditMode();
```

## Voir aussi

[document.enterEditMode\(\)](#)

# document.exportPNG()

## Disponibilité

Flash 8.

## Utilisation

```
document.exportPNG([fileURI [, bCurrentPNGSettings [, bCurrentFrame]])
```

## Paramètres

*fileURI* Chaîne de type « fichier:/// URI » qui spécifie le nom du fichier à exporter.

Si *fileURI* est une chaîne vide ou non spécifiée, Flash affiche la boîte de dialogue Exporter l'animation.

*bCurrentPNGSettings* Une valeur booléenne indiquant s'il faut utiliser les paramètres de publication PNG actuels (*true*) ou bien afficher la boîte de dialogue Exporter (*false*).

Ce paramètre est facultatif. La valeur par défaut est *false*.

*bCurrentFrame* Une valeur booléenne indiquant s'il faut exporter uniquement l'image actuelle (*true*) ou bien exporter toutes les images, chacune dans un fichier PNG distinct (*false*). Ce paramètre est facultatif. La valeur par défaut est *false*.

## Valeur renvoyée

Une valeur booléenne *true* si le fichier a été exporté sous la forme d'un fichier PNG ;

*false* sinon.

## Description

Méthode qui exporte le document dans un ou plusieurs fichiers PNG. Si *fileURI* est spécifié et que le fichier existe déjà, il est écrasé sans avertissement.

## Exemple

L'exemple suivant exporte l'image actuelle du document actuel vers *myFile.png*, en utilisant les paramètres de publication PNG actuels :

```
fl.getDocumentDOM().exportPNG("file:///C:/myProject/myFile.png", true,  
    true);
```

## document.exportPublishProfile()

### Disponibilité

Flash MX 2004.

### Utilisation

```
document.exportPublishProfile(fileURI)
```

### Paramètres

*fileURI* Chaîne de type « fichier:/// URI » qui spécifie le chemin d'accès du fichier XML dans lequel le profil doit être exporté.

### Valeur renvoyée

Aucune.

### Description

Méthode, qui exporte le profil actif dans un fichier XML.

### Exemple

L'exemple suivant exporte le profil actif dans le fichier profile.xml, qui réside dans le dossier /Documents and Settings/utilisateur/Desktop sur le lecteur C :

```
fl.getDocumentDOM().exportPublishProfile('file:///C:/Documents and  
Settings/username/Desktop/profile.xml');
```

## document.exportSWF()

### Disponibilité

Flash MX 2004.

### Utilisation

```
document.exportSWF([fileURI [, bCurrentSettings]])
```

### Paramètres

*fileURI* Chaîne de type « fichier:/// URI » qui spécifie le nom du fichier à exporter.

Si *fileURI* est vide ou non spécifié, Flash affiche la boîte de dialogue Exporter l'animation.

Ce paramètre est facultatif.

*bCurrentSettings* Valeur booléenne. Si elle est *true*, Flash utilise les paramètres de publication en SWF actuels. Sinon, Flash affiche la boîte de dialogue Exporter au format Flash Player. La valeur par défaut est *false*. Ce paramètre est facultatif.

## Valeur renvoyée

Aucune.

## Description

Méthode, qui exporte le document au format Flash SWF.

## Exemple

L'exemple suivant exporte le document à l'emplacement spécifié avec les paramètres de publication actuels :

```
fl.getDocumentDOM().exportSWF("file:///C:/Documents and Settings/joe_user/Desktop/qwerty.swf");
```

L'exemple suivant affiche les boîtes de dialogue Exporter l'animation, puis Exporter au format Flash Player, et exporte le document selon les paramètres spécifiés :

```
fl.getDocumentDOM().exportSWF("", true);
```

L'exemple suivant affiche la boîte de dialogue Exporter l'animation, puis exporte le document selon les paramètres spécifiés :

```
fl.getDocumentDOM().exportSWF();
```

# document.forceSimple

## Disponibilité

Flash MX 2004.

## Utilisation

```
document.forceSimple
```

## Description

Propriété ; valeur booléenne indiquant si les enfants de l'objet spécifié sont accessibles.

Cette propriété est équivalente à l'inverse du paramètre Rendre les objets enfant accessibles du panneau Accessibilité. Ainsi, si `forceSimple` est `true`, son effet est similaire à la désactivation de l'option Rendre les objets enfant accessibles. Si `forceSimple` est `false`, son effet est similaire à l'activation de l'option Rendre les objets enfant accessibles.

## Exemple

L'exemple suivant fixe la valeur de la variable `areChildrenAccessible` sur celle de la propriété `forceSimple` : La valeur `false` signifie que les enfants sont accessibles.

```
var areChildrenAccessible = fl.getDocumentDOM().forceSimple;
```

L'exemple suivant modifie la propriété `forceSimple` pour rendre les enfants du document accessibles :

```
fl.getDocumentDOM().forceSimple = false;
```

# document.frameRate

## Disponibilité

Flash MX 2004.

## Utilisation

```
document.frameRate
```

## Description

Propriété ; nombre en virgule flottante indiquant le nombre d'images affichées par seconde pendant la lecture du fichier SWF ; la valeur par défaut est 12. Cette propriété équivaut à définir la cadence dans la boîte de dialogue Propriétés du document (Modification > Document) dans le fichier FLA.

## Exemple

L'exemple suivant définit la cadence de 25,5 images par seconde :

```
fl.getDocumentDOM().frameRate = 25.5;
```

# document.getAlignToDocument()

## Disponibilité

Flash MX 2004.

## Utilisation

```
document.getAlignToDocument()
```

## Paramètres

Aucun.



### Valeur renvoyée

Une valeur booléenne : `true` si l'alignement des objets par rapport à la scène est défini dans les préférences ; `false` sinon.

### Description

Méthode ; effet identique à la lecture de la valeur du bouton Sur la scène dans le panneau Aligner. La préférence ainsi lue peut être utilisée sur ce document pour les méthodes `document.align()`, `document.distribute()`, `document.match()` et `document.space()`.

### Exemple

L'exemple suivant lit la valeur du bouton Sur la scène dans le panneau Aligner. Si la valeur renvoyée est `true`, le bouton Sur la scène est activé, sinon, il est désactivé.

```
var isAlignToDoc = fl.getDocumentDOM().getAlignToDocument();
fl.getDocumentDOM().align("left", isAlignToDoc);
```

### Voir aussi

`document.setAlignToDocument()`

## document.getBlendMode()

### Disponibilité

Flash 8.

### Utilisation

`document.getBlendMode()`

### Paramètres

Aucun.

### Valeur renvoyée

Renvoie une chaîne indiquant le mode de fondu pour les objets sélectionnés. Si plusieurs objets aux modes de fondu distincts sont sélectionnés, la chaîne reflète le mode de fondu de l'objet à la profondeur la plus élevée.

REMARQUE

La valeur renvoyée est imprévisible si certains objets sélectionnés ne gèrent pas les modes de fondu ou si leur mode de fondu est réglé sur la valeur `"normal"`.

## Description

Méthode ; renvoie une chaîne indiquant le mode de fondu pour les objets sélectionnés.

## Exemple

L'exemple suivant affiche le nom du mode de fondu dans le panneau Sortie :

```
fl.trace(fl.getDocumentDom().getBlendMode());
```

# document.getCustomFill()

## Disponibilité

Flash MX 2004.

## Utilisation

```
document.getCustomFill([objectToFill])
```

## Paramètres

*objectToFill* Chaîne indiquant l'emplacement de l'objet Fill (remplissage). Les valeurs suivantes sont possibles :

- "toolbar" renvoie l'objet Fill (remplissage) du panneau Outils et de l'inspecteur des propriétés.
- "selection" renvoie l'objet Fill (remplissage) de la sélection.

Si vous omettez ce paramètre, la valeur par défaut est "selection". S'il n'y a aucune sélection, la méthode renvoie *undefined*. Ce paramètre est facultatif.

## Valeur renvoyée

L'[Objet Fill](#) spécifié par le paramètre *objectToFill* en cas de succès, ou *undefined* en cas d'échec.

## Description

Méthode ; lit l'objet Fill (remplissage) de la forme sélectionnée ou, selon le paramètre, du panneau Outils et de l'inspecteur des propriétés.

## Exemple

L'exemple suivant lit l'objet Fill (remplissage) de la sélection, puis change la couleur de la sélection en blanc :

```
var fill = fl.getDocumentDOM().getCustomFill();  
fill.color = '#FFFFFF';  
fill.style = "solid";  
fl.getDocumentDOM().setCustomFill(fill);
```

L'exemple suivant renvoie l'objet de remplissage du panneau Outils et de l'inspecteur des propriétés, puis active un dégradé linéaire dans le nuancier :

```
var fill = fl.getDocumentDOM().getCustomFill("toolbar");
fill.style = "linearGradient";
fill.colorArray = [ 0x00ff00, 0xff0000, 0x0000ff ];
fill.posArray = [0, 100, 200];
fl.getDocumentDOM().setCustomFill( fill );
```

### Voir aussi

[document.setCustomFill\(\)](#)

## document.getCustomStroke()

### Disponibilité

Flash MX 2004.

### Utilisation

```
document.getCustomStroke([ locationOfStroke ])
```

### Paramètres

*locationOfStroke* Chaîne indiquant l'emplacement de l'objet trait. Les valeurs suivantes sont possibles :

- "toolbar" renvoie l'objet trait du panneau Outils et de l'inspecteur des propriétés, si elle est définie.
- "selection" renvoie l'objet trait de la sélection, si elle est définie.

Si vous omettez ce paramètre, la valeur par défaut est "selection". S'il n'y a aucune sélection, la méthode renvoie `undefined`. Ce paramètre est facultatif.

### Valeur renvoyée

L'[Objet Stroke](#) spécifié par le paramètre *locationOfStroke* en cas de succès, ou `undefined` en cas d'échec.

### Description

Renvoie l'objet trait de la forme sélectionnée ou, selon le paramètre, du panneau Outils et de l'inspecteur des propriétés.

## Exemple

L'exemple suivant les paramètres de trait actuels de la sélection et change l'épaisseur de trait à 2 :

```
var stroke = fl.getDocumentDOM().getCustomStroke("selection");
stroke.thickness = 2;
fl.getDocumentDOM().setCustomStroke(stroke);
```

L'exemple suivant renvoie les paramètres de trait actuels du panneau Outils et de l'inspecteur des propriétés, puis change la couleur de trait en rouge :

```
var stroke = fl.getDocumentDOM().getCustomStroke("toolbar");
stroke.color = "#FF0000";
fl.getDocumentDOM().setCustomStroke(stroke);
```

## Voir aussi

[document.setCustomStroke\(\)](#)

# document.getDataFromDocument()

## Disponibilité

Flash MX 2004.

## Utilisation

`document.getDataFromDocument(name)`

## Paramètres

*name* Chaîne spécifiant le nom des données à renvoyer.

## Valeur renvoyée

Les données spécifiées.

## Description

Méthode ; lit la valeur des données spécifiées. Le type renvoyé dépend du type de données enregistrées.

## Exemple

L'exemple suivant ajoute une valeur entière de 12 au document actif et appelle cette méthode pour afficher la valeur dans le panneau Sortie :

```
fl.getDocumentDOM().addDataToDocument("myData", "integer", 12);
fl.trace(fl.getDocumentDOM().getDataFromDocument("myData"));
```

## Voir aussi

[document.addDataToDocument\(\)](#), [document.documentHasData\(\)](#),  
[document.removeDataFromDocument\(\)](#)

# document.getElementProperty()

## Disponibilité

Flash MX 2004.

## Utilisation

```
document.getElementProperty(propertyName)
```

## Paramètres

*propertyName* Chaîne spécifiant le nom de la propriété Element dont la valeur est recherchée.

## Valeur renvoyée

La valeur de la propriété spécifiée. Renvoie `null` si la propriété est dans un état indéterminé, par exemple lorsque plusieurs éléments possédant des valeurs différentes pour cette propriété sont sélectionnés. Renvoie `undefined` si la propriété n'est pas une propriété valide de l'élément sélectionné.

## Description

Méthode ; lit la propriété Element spécifiée pour la sélection. La liste des valeurs gérées est indiquée dans la section « [Résumé des propriétés de l'objet Element](#) », à la page 226.

## Exemple

L'exemple suivant lit le nom (name) de la propriété Element de la sélection :

```
// elementName = nom d'occurrence de l'objet sélectionné.  
var elementName = fl.getDocumentDOM().getElementProperty("name");
```

## Voir aussi

[document.setElementProperty\(\)](#)

# document.getElementByIdTextAttr()

## Disponibilité

Flash MX 2004.

## Utilisation

```
document.getElementByIdTextAttr(attrName [, startIndex [, endIndex]])
```

## Paramètres

*attrName* Chaîne spécifiant le nom de la propriété `TextAttrs` à renvoyer. La liste des noms de propriétés et de leurs valeurs possibles figure dans la section « [Résumé des propriétés de l'objet TextAttrs](#) », à la page 532.

*startIndex* Nombre entier spécifiant l'index du premier caractère, 0 (zéro) étant l'index de la première position. Ce paramètre est facultatif.

*endIndex* Nombre entier qui spécifie l'index du dernier caractère. Ce paramètre est facultatif.

## Valeur renvoyée

Si un champ texte est sélectionné, la propriété est renvoyée si une seule valeur est utilisée pour tout le texte pour cette propriété. Renvoie `undefined` si le champ texte comporte plusieurs valeurs pour cette propriété. Si plusieurs champs texte sont sélectionnés, et si toutes les valeurs d'alignement de texte sont égales, la méthode renvoie cette valeur. Si plusieurs champs texte sont sélectionnés, mais que toutes les valeurs d'alignement de texte ne sont pas égales, la méthode renvoie `undefined`. Si les arguments facultatifs ne sont pas transmis, ces règles s'appliquent à la plage de texte sélectionnée, ou à tout le champ texte si le texte n'est pas en cours de modification. Si seul le paramètre *startIndex* est transmis, la propriété du caractère situé à droite de l'index est renvoyée, sous réserve que tous les objets `Text` sélectionnés aient une valeur identique. Si *startIndex* et *endIndex* sont transmis, la valeur renvoyée correspond à toute la plage de caractères comprise entre *startIndex* et le dernier caractère avant *endIndex*.

## Description

Méthode ; lit la propriété `TextAttrs` spécifiée pour les objets `Text` sélectionnés. Les objets sélectionnés qui ne sont pas des champs texte sont ignorés. La liste des noms de propriétés et de leurs valeurs possibles figure dans la section « [Résumé des propriétés de l'objet TextAttrs](#) », à la page 532. Voir aussi `document.setElementTextAttr()`.

### Exemple

L'exemple suivant lit la taille des champs texte sélectionnés :

```
fl.getDocumentDOM().getElementTextAttr("size");
```

L'exemple suivant lit la couleur du caractère d'indice 3 des champs texte sélectionnés :

```
fl.getDocumentDOM().getElementTextAttr("fillColor", 3);
```

L'exemple suivant lit le nom de la police de caractères du texte à partir de l'index 2 et jusqu'à l'index 10 (non inclus) des champs texte sélectionnés :

```
fl.getDocumentDOM().getElementTextAttr("face", 2, 10);
```

## document.getFilters()

### Disponibilité

Flash 8.

### Utilisation

```
document.getFilters()
```

### Paramètres

Aucun.

### Valeur renvoyée

Tableau contenant la liste des filtres appliqués aux objets actuellement sélectionnés.

### Description

Méthode ; renvoie un tableau contenant la liste des filtres appliqués aux objets actuellement sélectionnés. Si plusieurs objets sont sélectionnés et que leurs filtres ne sont pas identiques, cette méthode renvoie la liste des filtres appliqués au premier objet sélectionné.

### Exemple

Voir la section [document.setFilters\(\)](#).

### Voir aussi

[document.addFilter\(\)](#), [document.changeFilterOrder\(\)](#), [document.setFilters\(\)](#),  
[Objet Filter](#)

# document.getMetadata()

## Disponibilité

Flash 8.

## Utilisation

```
document.getMetadata()
```

## Paramètres

Aucun.

## Valeur renvoyée

Chaîne contenant les métadonnées XML associées au document, ou une chaîne vide s'il n'existe pas de métadonnées.

## Description

Méthode ; renvoie une chaîne contenant les métadonnées XML associées au document, ou une chaîne vide s'il n'existe pas de métadonnées.

## Exemple

L'exemple suivant affiche dans le panneau Sortie les métadonnées XML du document actif :

```
fl.trace("XML Metadata is :" + fl.getDocumentDOM().getMetadata());
```

## Voir aussi

```
document.setMetadata()
```

# document.getMobileSettings()

## Disponibilité

Flash CS3 Professional.

## Utilisation

```
document.getMobileSettings()
```

## Paramètres

Aucun.

## Valeur renvoyée

Chaîne représentant les paramètres XML pour le document. Si aucune valeur n'est spécifiée, une chaîne vide est renvoyée.



## Description

Méthode ; renvoie les paramètres XML mobiles pour le document.

## Exemple

L'exemple suivant affiche la chaîne des paramètres XML pour le document actif :

```
fl.trace(fl.getDocumentDOM().getMobileSettings());  
//trace une chaîne comme celle qui suit  
"<? xml version="1.0" encoding="UTF-16" standalone="no"  
?><mobileSettings> <contentType id="standalonePlayer" name="Standalone  
Player"/> <testDevices> <testDevice id="1170" name="Generic Phone"  
selected="yes"/> </testDevices> <outputMsgFiltering info="no" trace="yes"  
warning="yes"/> <testWindowState height="496" splitterClosed="No"  
splitterXPos="400" width="907"/> </mobileSettings>"
```

## Voir aussi

[document.setMobileSettings\(\)](#)

# document.getPlayerVersion()

## Disponibilité

Flash CS3 Professional.

## Utilisation

[document.getPlayerVersion\(\)](#)

## Paramètres

Aucun.

## Valeur renvoyée

Chaîne représentant la version de Flash Player spécifiée par l'intermédiaire de [document.setPlayerVersion\(\)](#). Si aucune valeur n'est définie, renvoie la valeur spécifiée dans la boîte de dialogue Paramètres de publication.

## Description

Méthode ; renvoie une chaîne représentant la version de Flash Player ciblée pour le document spécifié.

Pour déterminer quelle version d'ActionScript est actuellement ciblée dans le fichier spécifié, utilisez [document.asVersion](#).

## Exemple

L'exemple suivant illustre le ciblage des versions de Flash Player qui sont spécifiées pour le document actif, puis la récupération de ces valeurs :

```
fl.getDocumentDOM().setPlayerVersion("6");
var version = fl.getDocumentDOM().getPlayerVersion();
fl.trace(version) // affiche "6"
fl.getDocumentDOM().setPlayerVersion("FlashLite20");
var version = fl.getDocumentDOM().getPlayerVersion();
fl.trace(version) // affiche « "FlashLite20" »
```

## Voir aussi

[document.setPlayerVersion\(\)](#)

# document.getSelectionRect()

## Disponibilité

Flash MX 2004.

## Utilisation

`document.getSelectionRect()`

## Paramètres

Aucun.

## Valeur renvoyée

Le rectangle de délimitation de la sélection actuelle, ou 0 s'il n'existe pas de sélection.

Pour plus d'informations sur le format de la valeur renvoyée, consultez la section

[document.addNewRectangle\(\)](#).

## Description

Méthode ; lit les coordonnées du rectangle de délimitation de la sélection. Pour une sélection non rectangulaire, c'est le plus petit rectangle encadrant la sélection entière qui est renvoyé.

Le rectangle est basé sur l'espace du document ou, en mode de modification, sur le point d'alignement (également *point d'origine* ou *point zéro*) du symbole en cours de modification.

## Exemple

L'exemple suivant lit le rectangle de délimitation de la sélection actuelle, puis affiche ses propriétés :

```
var newRect = fl.getDocumentDOM().getSelectionRect();
var outputStr = "left: " + newRect.left + " top: " + newRect.top + " right: "
    + newRect.right + " bottom: " + newRect.bottom;
alert(outputStr);
```

## Voir aussi

`document.selection`, `document.setSelectionRect()`

# document.getString()

## Disponibilité

Flash MX 2004.

## Utilisation

```
document.getString([startIndex [, endIndex]])
```

## Paramètres

*startIndex* Nombre entier qui spécifie l'index du premier caractère voulu. Ce paramètre est facultatif.

*endIndex* Nombre entier qui spécifie l'index du dernier caractère voulu. Ce paramètre est facultatif.

## Valeur renvoyée

Une chaîne qui contient le texte sélectionné.

## Description

Méthode ; lit le texte sélectionné. Si les paramètres facultatifs ne sont pas transmis, la sélection de texte actuelle est utilisée. Si le texte n'est pas en cours de modification, toute la chaîne de texte est renvoyée. Si seul le paramètre *startIndex* est transmis, la méthode renvoie la chaîne qui commence à cet index et se termine à la fin du champ. Si *startIndex* et *endIndex* sont transmis, la valeur renvoyée correspond à toute la plage de caractères comprise entre *startIndex* et le dernier caractère avant *endIndex*.

Si plusieurs champs texte sont sélectionnés, c'est une concaténation de toutes les chaînes qui est renvoyée.

## Exemple

L'exemple suivant lit le contenu des champs texte sélectionnés :

```
fl.getDocumentDOM().getString();
```

L'exemple suivant lit le contenu des champs texte sélectionnés à partir du caractère d'indice 5 :

```
fl.getDocumentDOM().getString(5);
```

L'exemple suivant lit la chaîne à partir du caractère d'index 2 et jusqu'au caractère d'index 10 non compris :

```
fl.getDocumentDOM().getString(2, 10);
```

## Voir aussi

[document.setTextString\(\)](#)

# document.getTimeline()

## Disponibilité

Flash MX 2004.

## Utilisation

```
document.getTimeline()
```

## Paramètres

Aucun.

## Valeur renvoyée

L'objet Timeline actuel.

## Description

Méthode ; lit l'[Objet Timeline](#) actif dans le document. Le scénario actuel peut être la séquence actuelle, le symbole en cours de modification, ou l'écran actuel.

## Exemple

L'exemple suivant lit l'objet Timeline et renvoie le nombre d'images du calque le plus long :

```
var longestLayer = fl.getDocumentDOM().getTimeline().frameCount;  
fl.trace("The longest layer has" + longestLayer + "frames");
```

L'exemple suivant passe au mode de modification en place pour le symbole sélectionné sur la scène et insère une image dans le scénario du symbole.

```
fl.getDocumentDOM().enterEditMode("inPlace");  
fl.getDocumentDOM().getTimeline().insertFrames();
```

L'exemple suivant lit l'objet Timeline et affiche son nom :

```
var timeline = fl.getDocumentDOM().getTimeline();  
alert(timeline.name);
```

## Voir aussi

[document.currentTimeline](#), [document.timelines](#), [symbolItem.timeline](#)

# document.getTransformationPoint()

## Disponibilité

Flash MX 2004.

## Utilisation

```
document.getTransformationPoint()
```

## Paramètres

Aucun.

## Valeur renvoyée

Un point (par exemple, {x:10, y:20}), sachant que x et y sont des nombres à virgule flottante) qui précise la position du point de transformation (également *point d'origine* ou *point zéro*) dans le système de coordonnées de l'élément sélectionné.

## Description

Méthode ; lit l'emplacement du point de transformation de la sélection. Le point de transformation peut être utilisé pour des permutations, par exemple rotation et inclinaison.

### REMARQUE

Les points de transformation sont définis par rapport à différents emplacements, en fonction du type d'élément sélectionné. Pour plus d'informations, consultez la section [document.setTransformationPoint\(\)](#).

## Exemple

L'exemple suivant lit l'emplacement du point de transformation de la sélection. La propriété `transPoint.x` donne la coordonnée x du point de transformation. La propriété `transPoint.y` donne la coordonnée y du point de transformation.

```
var transPoint = fl.getDocumentDOM().getTransformationPoint();
```

## Voir aussi

[document.setTransformationPoint\(\)](#), [element.getTransformationPoint\(\)](#)

# document.group()

## Disponibilité

Flash MX 2004.

## Utilisation

`document.group()`

## Paramètres

Aucun.

## Valeur renvoyée

Aucune.

## Description

Méthode ; convertit la sélection en un groupe.

## Exemple

L'exemple suivant convertit en groupe les objets de la sélection actuelle :

```
fl.getDocumentDOM().group();
```

## Voir aussi

[document.unGroup\(\)](#)

# document.height

## Disponibilité

Flash MX 2004.

## Utilisation

`document.height`

## Description

Propriété ; nombre entier indiquant la hauteur en pixels du document (scène).

## Exemple

L'exemple suivant fixe la hauteur de la scène à 400 pixels :

```
fl.getDocumentDOM().height = 400;
```

## Voir aussi

[document.width](#)

# document.id

## Disponibilité

Flash CS3 Professional.

## Utilisation

`document.id`

## Description

Propriété en lecture seule ; nombre entier unique (attribué automatiquement) qui identifie un document lors d'une session Flash. Utilisez cette propriété conjointement avec `fl.findDocumentDOM()` afin de spécifier un document particulier pour une action.

## Exemple

L'exemple suivant affiche l'ID de document du document actif :

```
fl.trace("Current doc's internal ID is: " + fl.getDocumentDOM().id);
```

## Voir aussi

`fl.findDocumentDOM()`

# document.importFile()

## Disponibilité

Flash 8.

## Utilisation

```
document.importFile(fileURI [, importToLibrary])
```

## Paramètres

*fileURI* Chaîne de type « fichier:/// URI » qui spécifie le nom du fichier à importer.

*importToLibrary* Une valeur booléenne indiquant s'il faut importer le fichier uniquement dans la bibliothèque du document (`true`) ou bien en placer également une copie sur la scène (`false`). La valeur par défaut est `false`.

## Valeur renvoyée

Une valeur booléenne indiquant si le fichier a été importé avec succès.

## Description

Méthode ; importe un fichier dans un document. Cette méthode a le même effet que la commande Importer dans la bibliothèque ou Importer vers la scène. Pour importer un profil de publication, utilisez `document.importPublishProfile()`.

## Exemple

L'exemple suivant permet à l'utilisateur de rechercher le fichier qu'il veut importer sur la scène :

```
var dom = fl.getDocumentDOM();
var URI = fl.browseForFileURL("select", "Import File");
dom.importFile(URI);
```

## Voir aussi

`document.importSWF()`, `fl.browseForFileURL()`

# document.importPublishProfile()

## Disponibilité

Flash MX 2004.

## Utilisation

`document.importPublishProfile( fileURI )`

## Paramètres

*fileURI* Chaîne de type « fichier:/// URI » qui spécifie le chemin d'accès du fichier XML définissant le profil à importer.

## Valeur renvoyée

Un nombre entier qui est l'index du profil importé dans la liste des profils. Renvoie -1 s'il est impossible d'importer le profil.

## Description

Méthode ; importe un profil à partir d'un fichier.

## Exemple

L'exemple suivant importe le profil contenu dans le fichier profile.xml et affiche son index dans la liste des profils :

```
alert(fl.getDocumentDOM().importPublishProfile('file:///C:/Documents and
Settings/janeUser/Desktop/profile.xml'));
```



# document.importSWF()

## Disponibilité

Flash MX 2004.

## Utilisation

```
document.importSWF(fileURI)
```

## Paramètres

*fileURI* Chaîne de type « fichier:/// URI » qui spécifie le nom du fichier SWF à exporter.

## Valeur renvoyée

Aucune.

## Description

Méthode ; importe un fichier SWF dans le document. Cette méthode effectue une opération identique à l'utilisation de la commande Importer pour spécifier un fichier SWF. Dans Flash 8 et les versions ultérieures, vous pouvez également utiliser `document.importFile()` pour importer un fichier SWF (ainsi que d'autres types de fichiers).

## Exemple

L'exemple suivant importe le fichier "mySwf.swf" depuis le dossier de configuration de Flash :

```
fl.getDocumentDOM().importSWF(fl.configURI+"mySwf.swf");
```

## Voir aussi

[document.importFile\(\)](#)

# document.intersect()

## Disponibilité

Flash 8.

## Utilisation

```
document.intersect();
```

## Paramètres

Aucun.

## Valeur renvoyée

Valeur booléenne : `true` en cas de réussite, sinon `false`.

## Description

Méthode ; crée un objet de dessin Intersection à partir de tous les objets de dessin sélectionnés. La méthode renvoie `false` s'il n'existe pas d'objets de dessin sélectionnés, ou si l'un des éléments sélectionnés n'est pas un objet de dessin.

## Exemple

L'exemple suivant crée un objet de dessin Intersection à partir de tous les objets de dessin sélectionnés :

```
fl.getDocumentDOM().intersect();
```

## Voir aussi

[document.crop\(\)](#), [document.deleteEnvelope\(\)](#), [document.punch\(\)](#),  
[document.union\(\)](#), [shape.isDrawingObject](#)

# document.library

## Disponibilité

Flash MX 2004.

## Utilisation

```
document.library
```

## Description

Propriété en lecture seule ; [Objet library](#) d'un document.

## Exemple

L'exemple suivant lit la bibliothèque du document actif.

```
var myCurrentLib = fl.getDocumentDOM().library;
```

En supposant que le document actuellement actif n'est pas `fl.documents[1]`, l'exemple suivant lit la bibliothèque non active ou une bibliothèque ouverte à l'aide de la commande Fichier > Ouvrir comme bibliothèque externe :

```
var externalLib = fl.documents[1].library;
```

# document.livePreview

## Disponibilité

Flash MX 2004.

## Utilisation

`document.livePreview`

## Description

Propriété ; valeur booléenne indiquant si la fonction Aperçu en direct est activée. Si cette propriété est `true`, les composants apparaissent sur la scène tels qu'ils apparaîtront dans le contenu Flash publié, avec leur taille approximative. Si elle est `false`, seuls les contours des composants apparaissent. La valeur par défaut est `true`.

## Exemple

L'exemple suivant définit la fonction Aperçu en direct sur `false` :

```
fl.getDocumentDOM().livePreview = false;
```

# document.match()

## Disponibilité

Flash MX 2004.

## Utilisation

`document.match(bWidth, bHeight [, bUseDocumentBounds])`

## Paramètres

*bWidth* Valeur booléenne. Si elle est `true`, la méthode donne une largeur identique à tous les éléments sélectionnés.

*bHeight* Valeur booléenne. Si elle est `true`, la méthode donne une hauteur identique à tous les éléments sélectionnés.

*bUseDocumentBounds* Valeur booléenne. Si elle est `true`, la méthode aligne la taille des objets sur les limites du document. Sinon, la méthode utilise les limites de l'objet ayant la plus grande taille. La valeur par défaut est `false`. Ce paramètre est facultatif.

## Valeur renvoyée

Aucune.

## Description

Méthode ; donne la même taille à tous les objets sélectionnés.

## Exemple

L'exemple suivant donne une largeur identique à tous les éléments sélectionnés :

```
fl.getDocumentDOM().match(true,false);
```

L'exemple suivant donne une hauteur identique à tous les éléments sélectionnés :

```
fl.getDocumentDOM().match(false,true);
```

L'exemple suivant donne une largeur identique correspondant aux limites du document :

```
fl.getDocumentDOM().match(true,false,true);
```

## Voir aussi

[document.getAlignToDocument\(\)](#), [document.setAlignToDocument\(\)](#)

# document.mouseClick()

## Disponibilité

Flash MX 2004.

## Utilisation

```
document.mouseClick(position, bToggleSel, bShiftSel)
```

## Paramètres

*position* Paire de valeurs numériques à virgule flottante indiquant les coordonnées *x* et *y* du clic, en pixels.

*bToggleSel* Valeur booléenne indiquant l'état de la touche Maj : *true* si la touche est enfoncée, *false* si elle ne l'est pas.

*bShiftSel* Valeur booléenne indiquant l'état de la préférence Sélection avec la touche Maj : *true* si la préférence est activée, *false* si elle ne l'est pas.

## Valeur renvoyée

Aucune.

## Description

Méthode ; effectue un clic de souris à l'aide de l'outil Sélection.

## Exemple

L'exemple suivant simule un clic de souris à l'emplacement indiqué :

```
fl.getDocumentDOM().mouseClick({x:300, y:200}, false, false);
```

## Voir aussi

[document.mouseDoubleClick\(\)](#)

# document.mouseDbClick()

## Disponibilité

Flash MX 2004.

## Utilisation

`document.mouseDbClick(position, bAltDown, bShiftDown, bShiftSelect)`

## Paramètres

*position* Paire de valeurs numériques à virgule flottante indiquant les coordonnées *x* et *y* du clic, en pixels.

*bAltDown* Valeur booléenne qui indique si la touche Alt est enfoncée au moment de l'événement : *true* si la touche est enfoncée, *false* si elle ne l'est pas.

*bShiftDown* Valeur booléenne qui indique si la touche Maj est enfoncée au moment de l'événement : *true* si la touche est enfoncée, *false* si elle ne l'est pas.

*bShiftSelect* Valeur booléenne indiquant l'état de la préférence Sélection avec la touche Maj : *true* si la préférence est activée, *false* si elle ne l'est pas.

## Valeur renvoyée

Aucune.

## Description

Méthode ; effectue un double-clic de souris à l'aide de l'outil Sélection.

## Exemple

L'exemple suivant simule un double-clic de souris à l'emplacement indiqué :

```
fl.getDocumentDOM().mouseDbClick({x:392.9, y:73}, false, false, true);
```

## Voir aussi

[document.mouseClick\(\)](#)

# document.moveSelectedBezierPointsBy()

## Disponibilité

Flash MX 2004.

## Utilisation

`document.moveSelectedBezierPointsBy(delta)`

## Paramètres

*delta* Paire de nombres en virgule flottante indiquant les coordonnées  $x$  et  $y$  de déplacement, en pixels, des points de courbe de Bézier sélectionnés. Par exemple, les paramètres  $(\{x:1,y:2\})$  spécifient un emplacement situé à droite (de un pixel) et en-dessous (de deux pixels) de l'emplacement actuel.

## Valeur renvoyée

Aucune.

## Description

Méthode ; si la sélection contient au moins un trajet comptant au moins un point de courbe de Bézier, cette méthode déplace de la valeur spécifiée tous les points de courbe de Bézier de tous les trajets sélectionnés.

## Exemple

L'exemple suivant déplace les points de courbe de Bézier sélectionnés de 10 pixels vers la droite et 5 pixels vers le bas :

```
fl.getDocumentDOM().moveSelectedBezierPointsBy({x:10, y:5});
```

# document.moveSelectionBy()

## Disponibilité

Flash MX 2004.

## Utilisation

```
document.moveSelectionBy(distanceToMove)
```

## Paramètres

*distanceToMove* Paire de nombres en virgule flottante indiquant les valeurs de déplacement  $x$  et  $y$  de la sélection. Par exemple, les paramètres  $(\{x:1,y:2\})$  spécifient un emplacement situé à droite (de un pixel) et en-dessous (de deux pixels) de l'emplacement actuel.

## Valeur renvoyée

Aucune.

## Description

Méthode ; déplace les objets sélectionnés de la distance spécifiée.

### REMARQUE

Lorsque l'utilisateur utilise les touches de direction (flèches) pour déplacer un élément, le panneau Historique combine tous les appuis de touches fléchées en un seul déplacement. Lorsque l'utilisateur appuie à répétition sur les touches fléchées, plutôt que d'enregistrer plusieurs étapes dans le panneau Historique, la méthode effectue tous les déplacements en une seule étape, et les arguments sont actualisés pour reproduire les actions répétées sur les touches fléchées.

Pour plus d'informations sur les sélections, consultez les sections

[document.setSelectionRect\(\)](#), [document.mouseClick\(\)](#), [document.mouseDbClick\(\)](#) et [Objet Element](#).

## Exemple

L'exemple suivant déplace l'élément sélectionné de 62 pixels vers la droite et 84 pixels vers le bas :

```
flash.getDocumentDOM().moveSelectionBy({x:62, y:84});
```

# document.name

## Disponibilité

Flash MX 2004.

## Utilisation

`document.name`

## Description

Propriété en lecture seule ; chaîne représentant le nom d'un document (fichier FLA).

## Exemple

L'exemple suivant donne à la variable `fileName` le nom de fichier du premier document du tableau des documents :

```
var fileName = flash.documents[0].name;
```

L'exemple suivant affiche dans le panneau Sortie les noms de tous les documents ouverts :

```
var openDocs = fl.documents;
for(var i=0; i < openDocs.length; i++){
    fl.trace(i + " " + openDocs[i].name + "\n");
}
```

# document.optimizeCurves()

## Disponibilité

Flash MX 2004.

## Utilisation

`document.optimizeCurves(smoothing, bUseMultiplePasses)`

## Paramètres

*smoothing* Nombre entier compris entre 0 et 100, avec 0 signifiant pas de lissage, et 100 signifiant un lissage maximum.

*bUseMultiplePasses* Valeur booléenne. Si elle est `true`, la méthode doit utiliser plusieurs passages. Cette option est plus lente mais produit de meilleurs résultats. Ce paramètre a le même effet qu'un clic sur le bouton Appliquer plusieurs passages dans la boîte de dialogue Optimiser les courbes.

## Valeur renvoyée

Aucune.

## Description

Méthode ; optimise le lissage de la sélection, le cas échéant en autorisant plusieurs passages pour obtenir un lissage optimal. Cette méthode est l'équivalent de l'option de menu Modification > Forme > Optimiser.

## Exemple

L'exemple suivant optimise la courbe de la sélection à 50° de lissage en plusieurs passages :

```
fl.getDocumentDOM().optimizeCurves(50, true);
```

# document.path

## Disponibilité

Flash MX 2004.

## Utilisation

`document.path`

## Description

Propriété en lecture seule ; chaîne représentant le chemin d'accès du document dans un format spécifique à la plate-forme. Si le document n'a pas encore été enregistré, cette propriété renvoie `undefined`.



## Exemple

L'exemple suivant affiche dans le panneau Sortie le chemin d'accès du premier document du tableau des documents.

```
var filePath = flash.documents[0].path;  
fl.trace(filePath);
```

## document.publish()

### Disponibilité

Flash MX 2004.

### Utilisation

```
document.publish()
```

### Paramètres

Aucun.

### Valeur renvoyée

Aucune.

### Description

Méthode ; publie le document en fonction des paramètres de publication actuels (Fichier > Paramètres de publication). Cette méthode est l'équivalent de l'option de menu Fichier > Publier.

## Exemple

L'exemple suivant publie le document actif :

```
fl.getDocumentDOM().publish();
```

## document.publishProfiles

### Disponibilité

Flash MX 2004.

### Utilisation

```
document.publishProfiles
```

### Description

Propriété en lecture seule ; tableau des noms des profils de publication du document.

## Exemple

L'exemple suivant affiche les noms des profils de publication du document :

```
var myPubProfiles = fl.getDocumentDOM().publishProfiles;
for (var i=0; i < myPubProfiles.length; i++){
    fl.trace(myPubProfiles[i]);
}
```

## document.punch()

### Disponibilité

Flash 8.

### Utilisation

`document.punch()`

### Paramètres

Aucun.

### Valeur renvoyée

Une valeur booléenne : `true` en cas de réussite, `false` sinon.

### Description

Méthode ; sur la base de l'objet de dessin sélectionné le plus élevé, poinçonne tous les objets de dessin de niveau inférieur. La méthode renvoie `false` s'il n'existe pas d'objets de dessin sélectionnés, ou si l'un des éléments sélectionnés n'est pas un objet de dessin.

### Exemple

L'exemple suivant poinçonne tous les objets de dessin de niveau inférieur à l'objet de dessin sélectionné :

```
fl.getDocumentDOM().punch();
```

### Voir aussi

[document.crop\(\)](#), [document.deleteEnvelope\(\)](#), [document.intersect\(\)](#),  
[document.union\(\)](#), [shape.isDrawingObject](#)

# document.removeAllFilters()

## Disponibilité

Flash 8.

## Utilisation

```
document.removeAllFilters()
```

## Paramètres

Aucun.

## Valeur renvoyée

Aucune.

## Description

Méthode ; supprime tous les filtres des objets sélectionnés.

## Exemple

L'exemple suivant supprime tous les filtres des objets sélectionnés :

```
fl.getDocumentDOM().removeAllFilters();
```

## Voir aussi

[document.addFilter\(\)](#), [document.changeFilterOrder\(\)](#),  
[document.disableAllFilters\(\)](#), [document.getFilters\(\)](#), [document.removeFilter\(\)](#),  
[Objet Filter](#)

# document.removeDataFromDocument()

## Disponibilité

Flash MX 2004.

## Utilisation

```
document.removeDataFromDocument(name)
```

## Paramètres

*name* Chaîne spécifiant le nom des données à supprimer.

## Valeur renvoyée

Aucune.

## Description

Méthode ; supprime les données persistantes associées au document et correspondant au nom spécifié.

## Exemple

L'exemple suivant supprime du document les données persistantes ayant le nom "myData" :

```
fl.getDocumentDOM().removeDataFromDocument("myData");
```

## Voir aussi

[document.addDataToDocument\(\)](#), [document.documentHasData\(\)](#),  
[document.getDataFromDocument\(\)](#)

# document.removeDataFromSelection()

## Disponibilité

Flash MX 2004.

## Utilisation

```
document.removeDataFromSelection(name)
```

## Paramètres

*name* Chaîne spécifiant le nom des données persistantes à supprimer.

## Valeur renvoyée

Aucune.

## Description

Méthode ; supprime les données persistantes associées à la sélection et correspondant au nom spécifié.

## Exemple

L'exemple suivant supprime de la sélection les données persistantes ayant le nom "myData" :

```
fl.getDocumentDOM().removeDataFromSelection("myData");
```

## Voir aussi

[document.addDataToSelection\(\)](#)

# document.removeFilter()

## Disponibilité

Flash 8.

## Utilisation

```
document.removeFilter(filterIndex)
```

## Paramètres

*filterIndex* Nombre entier indiquant l'index (en base 0) dans la liste Filtres du filtre à supprimer des objets sélectionnés.

## Valeur renvoyée

Aucune.

## Description

Méthode ; supprime le filtre spécifié de la liste Filtres pour les objets sélectionnés.

## Exemple

L'exemple suivant supprime le premier filtre (index 0) de la liste Filtres pour les objets sélectionnés :

```
fl.getDocumentDOM().removeFilter(0);
```

## Voir aussi

[document.addFilter\(\)](#), [document.changeFilterOrder\(\)](#), [document.disableFilter\(\)](#), [document.getFilters\(\)](#), [document.removeAllFilters\(\)](#), [Objet Filter](#)

# document.renamePublishProfile()

## Disponibilité

Flash MX 2004.

## Utilisation

```
document.renamePublishProfile([profileNewName])
```

## Paramètres

*profileNewName* Paramètre facultatif qui indique le nouveau nom du profil. Ce nouveau nom doit être unique. Si vous ne spécifiez pas de nom, un nom par défaut est utilisé.

## Valeur renvoyée

Une valeur booléenne : `true` en cas de réussite, `false` sinon.

## Description

Méthode ; renomme le profil actuel.

## Exemple

L'exemple suivant renomme le profil actuel avec un nom par défaut qui est ensuite affiché :

```
alert(fl.getDocumentDOM().renamePublishProfile());
```

# document.renameScene()

## Disponibilité

Flash MX 2004.

## Utilisation

```
document.renameScene(name)
```

## Paramètres

*nom* Chaîne spécifiant le nouveau nom de la séquence.

## Valeur renvoyée

Une valeur booléenne : *true* en cas de réussite, *false* sinon. Si le nouveau nom n'est pas unique, par exemple, la méthode renvoie *false*.

## Description

Méthode ; renomme la séquence sélectionnée dans le panneau Séquences. Le nouveau nom de la séquence doit être unique.

## Exemple

L'exemple suivant renomme la séquence active avec le nom "new name" :

```
var success = fl.getDocumentDOM().renameScene("new name");
```

# document.reorderScene()

## Disponibilité

Flash MX 2004.

## Utilisation

```
document.reorderScene(sceneToMove, sceneToPutItBefore)
```

## Paramètres

*sceneToMove* Nombre entier spécifiant l'index de la séquence à déplacer, 0 (zéro) étant la première séquence.

*sceneToPutItBefore* Nombre entier spécifiant la séquence devant laquelle la séquence définie par *sceneToMove* doit être placée. Indiquez 0 (zéro) pour la première séquence. Par exemple, si vous spécifiez 1 pour *sceneToMove* et 0 pour *sceneToPutItBefore*, la seconde séquence est placée devant la première. Spécifiez -1 pour placer la séquence à la fin.

## Valeur renvoyée

Aucune.

## Description

Méthode ; place la séquence spécifiée devant une autre séquence spécifiée.

## Exemple

L'exemple suivant place la seconde séquence devant la première :

```
fl.getDocumentDOM().reorderScene(1, 0);
```

# document.resetOvalObject()

## Disponibilité

Flash CS3 Professional.

## Utilisation

```
document.resetOvalObject()
```

## Paramètres

Aucun.

## Valeur renvoyée

Aucune.

## Description

Méthode ; dans l'inspecteur des propriétés, définit toutes les valeurs sur les paramètres par défaut de l'objet Oval. Si des objets Oval sont sélectionnés, leurs propriétés sont également redéfinies sur les valeurs par défaut.

## Exemple

L'exemple suivant redéfinit les propriétés de l'objet Oval du document actif sur les valeurs par défaut :

```
fl.getDocumentDOM().resetOvalObject();
```

## Voir aussi

[document.resetRectangleObject\(\)](#)

# document.resetRectangleObject()

## Disponibilité

Flash CS3 Professional.

## Utilisation

```
document.resetRectangleObject()
```

## Paramètres

Aucun.

## Valeur renvoyée

Aucune.

## Description

Méthode ; dans l'inspecteur des propriétés, définit toutes les valeurs sur les paramètres par défaut de l'objet Rectangle. Si des objets Rectangle sont sélectionnés, leurs propriétés sont également redéfinies sur les valeurs par défaut.

## Exemple

L'exemple suivant redéfinit les propriétés de l'objet Rectangle du document actif sur les valeurs par défaut :

```
fl.getDocumentDOM().resetRectangleObject();
```

## Voir aussi

[document.resetOvalObject\(\)](#)



# document.resetTransformation()

## Disponibilité

Flash MX 2004.

## Utilisation

```
document.resetTransformation()
```

## Paramètres

Aucun.

## Valeur renvoyée

Aucune.

## Description

Méthode ; réinitialise la matrice de transformation. Cette méthode est l'équivalent de l'option de menu Modification > Transformer > Supprimer la transformation.

## Exemple

L'exemple suivant réinitialise la matrice de transformation de la sélection.

```
fl.getDocumentDOM().resetTransformation();
```

# document.revert()

## Disponibilité

Flash MX 2004.

## Utilisation

```
document.revert()
```

## Paramètres

Aucun.

## Valeur renvoyée

Aucune.

## Description

Méthode ; restaure la dernière version enregistrée du document spécifié. Cette méthode est l'équivalent de l'option de menu Fichier > Rétablir.

## Exemple

L'exemple suivant restaure la dernière version enregistrée du document actif.

```
fl.getDocumentDOM().revert();
```

## Voir aussi

[document.canRevert\(\)](#), [fl.revertDocument\(\)](#)

# document.revertToLastVersion()

## Disponibilité

Flash CS3 Professional.

## Utilisation

```
document.revertToLastVersion()
```

## Paramètres

Aucun.

## Valeur renvoyée

Valeur booléenne `true` si le rétablissement du document est réussi, `false` dans le cas contraire.

## Description

Méthode ; s'il est possible de rétablir le fichier, affiche une boîte de dialogue permettant à l'utilisateur de confirmer cette restauration. En cas de confirmation de l'utilisateur, cette méthode remplace la version du fichier par celle stockée sur le serveur Version Cue et consigne les éventuelles erreurs dans le panneau Sortie.

## Exemple

L'exemple suivant remplace la version du document actif par celle qui est stockée sur le serveur Version Cue :

```
fl.getDocumentDOM().revertToLastVersion();
```

## Voir aussi

[document.canSaveAVersion\(\)](#), [document.saveAVersion\(\)](#),  
[document.synchronizeWithHeadVersion\(\)](#), [fl.revertDocumentToLastVersion\(\)](#)

# document.rotateSelection()

## Disponibilité

Flash MX 2004.

## Utilisation

`document.rotateSelection(angle [, rotationPoint])`

## Paramètres

*angle* Valeur en virgule flottante spécifiant l'angle de rotation.

*rotationPoint* Chaîne spécifiant le côté du rectangle de délimitation sur lequel doit s'effectuer la rotation. Les valeurs gérées sont "top right" (haut droite), "top left" (haut gauche), "bottom right" (bas droite), "bottom left" (bas gauche), "top center" (centre haut), "right center" (centre droite), "bottom center" (centre bas) et "left center" (centre gauche). Si ce paramètre n'est pas spécifié, la méthode utilise le point de transformation. Ce paramètre est facultatif.

## Valeur renvoyée

Aucune.

## Description

Méthode ; effectue sur la sélection une rotation du nombre de degrés spécifié. L'effet est identique à celui de l'outil Transformation libre utilisé pour une rotation de l'objet.

## Exemple

L'exemple suivant effectue une rotation de 45° de la sélection autour de son point de transformation :

```
flash.getDocumentDOM().rotateSelection(45);
```

L'exemple suivant effectue une rotation de 45 degrés de la sélection autour de son coin inférieur gauche :

```
fl.getDocumentDOM().rotateSelection(45, "bottom left");
```

# document.save()

## Disponibilité

Flash MX 2004.

## Utilisation

```
document.save([bOkToSaveAs])
```

## Paramètres

*bOkToSaveAs* Paramètre facultatif. S'il est défini sur `true` ou omis, et si le fichier n'a jamais été enregistré, la boîte de dialogue Enregistrer sous apparaît. S'il est défini sur `false`, le fichier n'est pas sauvegardé même s'il n'a jamais été enregistré auparavant.

## Valeur renvoyée

Une valeur booléenne : `true` en cas de réussite de l'enregistrement, `false` sinon.

## Description

Méthode ; enregistre le document à son emplacement par défaut. Cette méthode est l'équivalent de l'option de menu Fichier > Enregistrer.

Pour donner un autre nom au fichier (au lieu de l'enregistrer avec un nom identique), utilisez [fl.saveDocument\(\)](#).

### REMARQUE

Si le fichier est nouveau et qu'il n'a été ni enregistré ni modifié, ou s'il existe et n'a pas été modifié depuis son dernier enregistrement, cette méthode n'a aucun effet et la valeur `false` est renvoyée. Pour permettre l'enregistrement d'un fichier qui n'a pas été enregistré ni modifié, utilisez [document.saveAndCompact\(\)](#) ou [fl.saveDocumentAs\(\)](#).

## Exemple

L'exemple suivant enregistre le document actif à son emplacement par défaut :

```
fl.getDocumentDOM().save();
```

## Voir aussi

[document.saveAndCompact\(\)](#), [fl.saveAll\(\)](#), [fl.saveDocument\(\)](#),  
[fl.saveDocumentAs\(\)](#)

# document.saveAndCompact()

## Disponibilité

Flash MX 2004.

## Utilisation

```
document.saveAndCompact([bOkToSaveAs])
```

## Paramètres

*bOkToSaveAs* Paramètre facultatif. S'il est `true` ou s'il est omis, et si le fichier n'a jamais été enregistré, la boîte de dialogue Enregistrer sous apparaît. S'il est `false`, le fichier n'est pas enregistré, même s'il n'a jamais été enregistré. La valeur par défaut est `true`.

## Valeur renvoyée

Une valeur booléenne : `true` en cas de réussite de l'enregistrement avec compression, `false` sinon.

## Description

Méthode ; enregistre et compresse le fichier. Cette méthode est l'équivalent de l'option de menu Fichier > Enregistrer et compresser.

### REMARQUE

Si le fichier n'a jamais été enregistré, cette méthode renvoie `true` même si l'utilisateur annule la boîte de dialogue Enregistrer sous. Pour déterminer précisément si le fichier a été enregistré ou non, utilisez [fl.saveDocumentAs\(\)](#).

## Exemple

L'exemple suivant enregistre et compresse le document actif :

```
fl.getDocumentDOM().saveAndCompact();
```

## Voir aussi

[document.save\(\)](#), [fl.saveDocumentAs\(\)](#), [fl.saveDocument\(\)](#), [fl.saveAll\(\)](#)

# document.saveAVersion()

## Disponibilité

Flash CS3 Professional.

## Utilisation

```
document.saveAVersion()
```

## Paramètres

Aucun.

## Valeur renvoyée

Valeur booléenne `true` si une version du document est correctement enregistrée sur le serveur Version Cue, sinon `false`.

## Description

Méthode ; si le fichier peut être enregistré sur le serveur Version Cue, affiche une boîte de dialogue permettant à l'utilisateur d'entrer des commentaires sur la version, puis enregistre une version du document spécifié sur le serveur et consigne les éventuelles erreurs dans le panneau Sortie.

### REMARQUE

Si l'enregistrement du fichier par Flash est impossible à cause d'une mise en cache des informations d'identification du serveur dans la session de l'application active, un message d'erreur signalant l'échec de l'authentification s'affiche dans le panneau Sortie. Si cette erreur se produit, l'utilisateur doit se servir de la boîte de dialogue Fichier > Ouvrir pour ouvrir l'espace de travail de Version Cue avec les informations d'identification appropriées. Les appels ultérieurs de l'API JavaScript à ce serveur aboutiront ensuite sans problème.

## Exemple

See [document.canSaveAVersion\(\)](#).

## Voir aussi

[document.canSaveAVersion\(\)](#), [document.revertToLastVersion\(\)](#),  
[document.synchronizeWithHeadVersion\(\)](#)

# document.scaleSelection()

## Disponibilité

Flash MX 2004.

## Utilisation

`document.scaleSelection(xScale, yScale [, whichCorner])`

## Paramètres

*xScale* Valeur en virgule flottante spécifiant la valeur *x* de mise à l'échelle horizontale.

*yScale* Valeur en virgule flottante spécifiant la valeur *y* de mise à l'échelle verticale.

*whichCorner* Valeur chaîne indiquant le coin à partir duquel est appliquée la transformation. Si ce paramètre n'est pas spécifié, la méthode utilise le point de transformation. Les valeurs acceptables sont les suivantes : "bottom left" (bas gauche), "bottom right" (bas droite), "top right" (haut droite), "top left" (haut gauche), "top center" (centre haut), "right center" (centre droite), "bottom center" (centre bas) et "left center" (centre gauche). Ce paramètre est facultatif.

## Valeur renvoyée

Aucune.

## Description

Méthode ; effectue sur la sélection une mise à l'échelle de la valeur spécifiée. L'effet est identique à celui de l'outil Transformation libre utilisé pour une mise à l'échelle de l'objet.

## Exemple

L'exemple suivant double la largeur originale de la sélection et réduit sa hauteur de moitié :

```
flash.getDocumentDOM().scaleSelection(2.0, 0.5);
```

L'exemple suivant fait pivoter la sélection sur l'axe vertical :

```
fl.getDocumentDOM().scaleSelection(1, -1);
```

L'exemple suivant fait pivoter la sélection sur l'axe horizontal :

```
fl.getDocumentDOM().scaleSelection(-1, 1);
```

L'exemple suivant effectue une mise à l'échelle verticale de la sélection, avec un facteur de 1,9 à partir du centre haut :

```
fl.getDocumentDOM().scaleSelection(1, 1.90, 'top center');
```

# document.screenOutline

## Disponibilité

Flash MX 2004.

## Utilisation

`document.screenOutline`

## Description

Propriété en lecture seule ; objet ScreenOutline actuel du document. Avant d'accéder à l'objet pour la première fois, il est conseillé d'utiliser `document.allowScreens()` pour vérifier l'existence de la propriété.

## Exemple

L'exemple suivant affiche un tableau des valeurs de la propriété `screenOutline` :

```
var myArray = new Array();
for(var i in fl.getDocumentDOM().screenOutline) {
    myArray.push(" "+i+" : "+fl.getDocumentDOM().screenOutline[i]) ;
}
fl.trace("Here is the property dump for screenOutline: "+myArray);
```

## Voir aussi

[document.allowScreens\(\)](#), [Objet ScreenOutline](#)

# document.selectAll()

## Disponibilité

Flash MX 2004.

## Utilisation

`document.selectAll()`

## Paramètres

Aucun.

## Valeur renvoyée

Aucune.



## Description

Méthode ; sélectionne tous les éléments sur la scène. Cette méthode est l'équivalent des raccourcis clavier Ctrl+A (Windows) ou Commande+A (Macintosh), ou de l'option de menu Edition > Sélectionner tout.

## Exemple

L'exemple suivant sélectionne tous les éléments actuellement visibles pour l'utilisateur :

```
fl.getDocumentDOM().selectAll();
```

## Voir aussi

[document.selection](#), [document.selectNone\(\)](#)

# document.selection

## Disponibilité

Flash MX 2004.

## Utilisation

`document.selection`

## Description

Propriété ; tableau des objets sélectionnés dans le document. Si rien n'est sélectionné, la propriété renvoie un tableau de longueur zéro. Si aucun document n'est ouvert, elle renvoie `null`.

Pour ajouter des objets au tableau, vous devez d'abord les sélectionner à l'aide de l'une des techniques suivantes :

- Sélectionnez manuellement les objets sur la scène.
- Utilisez l'une des méthodes de sélection, telles que `document.setSelectionRect()`, `document.setSelectionBounds()`, `document.mouseClick()`, `document.mouseDbClick()` ou `document.selectAll()`.
- Sélectionnez manuellement une ou plusieurs images.
- Utilisez l'une des méthodes de l'[Objet Timeline](#) pour sélectionner une ou plusieurs images, telles que `timeline.getSelectedFrames()`, `timeline.setSelectedFrames()` ou `timeline.selectAllFrames()`, comme indiqué ci-après.

```
fl.getDocumentDOM().selection =  
    fl.getDocumentDOM().getTimeline().layers[0].frames[0].elements[0];
```

- Spécifiez tous les éléments dans une image particulière (voir [Objet Element](#)). Reportez-vous au premier exemple ci-dessous.
- Créez un tableau d'un ou de plusieurs éléments, puis attribuez ce tableau au tableau `document.selection`. Reportez-vous au troisième exemple ci-dessous.

## Exemple

L'exemple suivant assigne tous les éléments de l'image 11 à la sélection actuelle (n'oubliez pas que les valeurs d'index diffèrent des valeurs de numéro d'image) :

```
fl.getDocumentDOM().getTimeline().currentFrame = 10;
fl.getDocumentDOM().selection =
    fl.getDocumentDOM().getTimeline().layers[0].frames[10].elements;
```

L'exemple suivant crée un rectangle dans le coin supérieur gauche de la scène, puis une chaîne de texte sous ce rectangle : ces deux objets sont ensuite sélectionnés à l'aide de `document.setSelectionRect()` et ajoutés au tableau `document.selection`. Enfin, le contenu de `document.selection` est affiché dans le panneau Sortie.

```
fl.getDocumentDOM().addNewRectangle({left:0, top:0, right:99, bottom:99},
    0);
fl.getDocumentDOM().addNewText({left:-1, top:117.3, right:9.2,
    bottom:134.6});
fl.getDocumentDOM().setTextString('Hello World');
fl.getDocumentDOM().setSelectionRect({left:-28, top:-22, right:156.0,
    bottom:163});

var theSelectionArray = fl.getDocumentDOM().selection;

for(var i=0;i<theSelectionArray.length;i++){
    fl.trace("fl.getDocumentDOM().selection["+i+"] = " +
        theSelectionArray[i]);
}
```

L'exemple suivant est plus complexe. Il montre comment parcourir en boucle le tableau des calques et le tableau des éléments pour retrouver les occurrences d'un symbole particulier et les sélectionner. Cet exemple pourrait être étendu en y ajoutant des boucles analysant les images ou les séquences. Cet exemple affecte à la sélection active toutes les occurrences du clip `myMovieClip` figurant dans la première image :

```
// Affecte le tableau des calques à la variable "theLayers".
var theLayers = fl.getDocumentDOM().getTimeline().layers;
// Crée un tableau pour contenir tous les éléments
// qui sont des occurrences de "myMovieClip".
var myArray = new Array();
// Variable de compteur
var x = 0;
// Lancer une boucle dans tous les calques.
```

```

for (var i = 0; i < theLayers.length; i++) {
    // Récupère le tableau d'éléments dans l'image 1
    // et l'attribue au tableau "theElems".
    var theElems = theLayers[i].frames[0].elements;
    // Lancer une boucle dans tous les éléments d'un calque.
    for (var c = 0; c < theElems.length; c++) {
        // Vérifie si l'élément est de type "instance" (occurrence).
        if (theElems[c].elementType == "instance") {
            // Si l'élément est une occurrence, vérifie
            // s'il s'agit d'une occurrence de "myMovieClip".
            if (theElems[c].libraryItem.name == "myMovieClip") {
                // Affecte les éléments qui sont des occurrences
                // de "myMovieClip" à "myArray".
                myArray[x] = theElems[c];
                // Incrémente la variable de compteur.
                x++;
            }
        }
    }
}
// Maintenant que toutes les occurrences de "myMovieClip" sont affectées
// à "myArray", définir le tableau document.selection
// égal à myArray. Cette opération sélectionne les objets sur la scène.
fl.getDocumentDOM().selection = myArray;

```

## document.selectNone()

### Disponibilité

Flash MX 2004.

### Utilisation

document.selectNone()

### Paramètres

Aucun.

### Valeur renvoyée

Aucune.

### Description

Méthode ; désélectionne les éléments sélectionnés.

### Exemple

L'exemple suivant désélectionne tous les éléments sélectionnés :

```
fl.getDocumentDOM().selectNone();
```

## Voir aussi

[document.selectAll\(\)](#), [document.selection](#)

# document.setAlignToDocument()

## Disponibilité

Flash MX 2004.

## Utilisation

```
document.setAlignToDocument(bToStage)
```

## Paramètres

*bToStage* Valeur booléenne. Si elle est `true`, la méthode aligne les objets sur la scène. Si elle est `false`, elle ne les aligne pas.

## Valeur renvoyée

Aucune.

## Description

Méthode ; définit les préférences afin d'activer [document.align\(\)](#), [document.distribute\(\)](#), [document.match\(\)](#) et [document.space\(\)](#) pour le document. Cette commande est équivalente à l'activation de l'option Sur la scène dans le panneau Aligner.

## Exemple

L'exemple suivant active le bouton Sur la scène dans le panneau Aligner, afin d'aligner les objets sur la scène :

```
fl.getDocumentDOM().setAlignToDocument(true);
```

## Voir aussi

[document.getAlignToDocument\(\)](#)

# document.setBlendMode()

## Disponibilité

Flash 8.

## Utilisation

```
document.setBlendMode(mode)
```

## Paramètres

*mode* Chaîne représentant le mode de fondu désiré pour les objets sélectionnés. Les valeurs gérées sont "normal", "layer", "multiply", "screen", "overlay", "hardlight", "lighten", "darken", "difference", "add", "subtract", "invert", "alpha" et "erase".

## Valeur renvoyée

Aucune.

## Description

Méthode ; définit le mode de fondu pour les objets sélectionnés.

## Exemple

L'exemple suivant règle sur "add" le mode de fondu pour l'objet sélectionné.

```
fl.getDocumentDOM().setBlendMode("add");
```

## Voir aussi

[document.addFilter\(\)](#), [document.setFilterProperty\(\)](#), [symbolInstance.blendMode](#)

# document.setCustomFill()

## Disponibilité

Flash MX 2004.

## Utilisation

```
document.setCustomFill(fill)
```

## Paramètres

*fill* Objet Fill qui spécifie les paramètres de remplissages requis. Voir la section [Objet Fill](#).

## Valeur renvoyée

Aucune.

## Description

Méthode ; définit les paramètres de remplissage pour le panneau Outils, l'inspecteur des propriétés et les formes sélectionnées, le cas échéant. Cette opération permet au script de définir les paramètres de remplissage avant de tracer un objet, ce qui est plus élégant que dessiner l'objet, le sélectionner et modifier ses paramètres de remplissage. Elle permet aussi de modifier par script les paramètres de remplissage du panneau Outils et de l'inspecteur des propriétés.

## Exemple

L'exemple suivant définit le blanc comme couleur active dans la palette des couleurs de remplissage du panneau Outils, de l'inspecteur des propriétés et des éventuelles formes sélectionnées :

```
var fill = fl.getDocumentDOM().getCustomFill();
fill.color = "#FFFFFF";
fill.style = "solid";
fl.getDocumentDOM().setCustomFill(fill);
```

## Voir aussi

[document.getCustomFill\(\)](#)

# document.setCustomStroke()

## Disponibilité

Flash MX 2004.

## Utilisation

`document.setCustomStroke(stroke)`

## Paramètres

*stroke* Un [Objet Stroke](#).

## Valeur renvoyée

Aucune.

## Description

Méthode ; définit les paramètres de trait pour le panneau Outils, l'inspecteur des propriétés et les formes sélectionnées, le cas échéant. Cette opération permet au script de définir les paramètres de trait avant de tracer un objet, ce qui est plus élégant que dessiner l'objet, le sélectionner et modifier ses paramètres de trait. Elle permet aussi de modifier par script les paramètres de trait du panneau Outils et de l'inspecteur des propriétés.

## Exemple

L'exemple suivant modifie les paramètres d'épaisseur de trait du panneau Outils, de l'inspecteur des propriétés et des formes sélectionnées, le cas échéant :

```
var stroke = fl.getDocumentDOM().getCustomStroke();
stroke.thickness += 2;
fl.getDocumentDOM().setCustomStroke(stroke);
```

## Voir aussi

[document.getCustomStroke\(\)](#)

# document.setElementProperty()

## Disponibilité

Flash MX 2004.

## Utilisation

```
document.setElementProperty(property, value)
```

## Paramètres

*property* Chaîne spécifiant le nom de la propriété `Element` à définir. La liste complète des propriétés et de leurs valeurs figure dans la section « [Résumé des propriétés de l'objet Element](#) », à la page 226.

REMARQUE

Cette méthode ne permet pas de modifier les propriétés en lecture seule, telles que `element.elementType`, `element.top` et `element.left`.

*value* Nombre entier spécifiant la valeur à attribuer à la propriété `Element` spécifiée.

## Valeur renvoyée

Aucune.

## Description

Méthode ; définit la propriété `Element` spécifiée pour les objets sélectionnés du document. La méthode est inopérante s'il n'existe pas de sélection.

## Exemple

L'exemple suivant définit la largeur de tous les éléments sélectionnés à 100 et leur hauteur à 50 :

```
f1.getDocumentDOM().setElementProperty("width", 100);  
f1.getDocumentDOM().setElementProperty("height", 50);
```

# document.setElementTextAttr()

## Disponibilité

Flash MX 2004.

## Utilisation

```
document.setElementTextAttr(attrName, attrValue [, startIndex  
[, endIndex]])
```

## Paramètres

*attrName* Chaîne spécifiant le nom de la propriété `TextAttrs` à modifier.

*attrValue* Valeur à affecter à la propriété `TextAttrs` désirée. La liste des noms de propriétés et de leurs valeurs possibles figure dans la section « [Résumé des propriétés de l'objet TextAttrs](#) », à la page 532.

*startIndex* Nombre entier spécifiant l'index du premier caractère concerné. Ce paramètre est facultatif.

*endIndex* Nombre entier spécifiant l'index du dernier caractère concerné. Ce paramètre est facultatif.

## Valeur renvoyée

Une valeur booléenne : `true` si au moins une propriété d'attribut de texte est modifiée, `false` sinon.

## Description

Méthode ; définit selon la valeur spécifiée la propriété `TextAttrs` spécifiée pour les objets texte sélectionnés. La liste des noms de propriétés et de leurs valeurs possibles figure dans la section « [Résumé des propriétés de l'objet TextAttrs](#) », à la page 532. Si les paramètres facultatifs ne sont pas transmis, la méthode définit le style de la plage de texte sélectionnée, ou de tout le champ texte si aucun texte n'est sélectionné. Si seul le paramètre *startIndex* est transmis, la méthode définit les attributs de ce caractère. Si *startIndex* et *endIndex* sont transmis, la méthode définit les attributs de toute la plage de caractères comprise entre *startIndex* et le dernier caractère avant *endIndex*. Si des styles de paragraphe sont spécifiés, tous les paragraphes de la plage sélectionnée sont modifiés.

## Exemple

Les exemples suivants affectent les attributs de texte `fillColor`, `italic` et `bold` aux éléments de texte sélectionnés :

```
var success = fl.getDocumentDOM().setElementTextAttr("fillColor",  
    "#00ff00");  
var pass = fl.getDocumentDOM().setElementTextAttr("italic", true, 10);  
var ok = fl.getDocumentDOM().setElementTextAttr("bold", true, 5, 15);
```



# document.setFillColor()

## Disponibilité

Flash MX 2004.

## Utilisation

```
document.setFillColor(color)
```

## Paramètres

*color* Couleur du remplissage, exprimée dans l'un des formats suivants :

- chaîne au format "#RRRVBB" ou "#RRVBBAA" ;
- nombre hexadécimal au format 0xRRRVBB ;
- entier représentant l'équivalent décimal d'un nombre hexadécimal.

Si la valeur du paramètre est `null`, aucune couleur de remplissage n'est définie, ce qui est l'équivalent du choix de l'option « Pas de remplissage » dans la palette des couleurs de remplissage.

## Valeur renvoyée

Aucune.

## Description

Méthode ; modifie la couleur de remplissage de la sélection avec la couleur spécifiée. Pour plus d'informations sur la modification de la couleur de remplissage dans le panneau Outils ou dans l'inspecteur Propriétés, consultez la section [document.setCustomFill\(\)](#).

## Exemple

Dans l'exemple suivant, les trois premières instructions définissent la couleur de remplissage à l'aide des trois différents formats de définition de la couleur. La quatrième instruction annule le remplissage.

```
flash.getDocumentDOM().setFillColor("#cc00cc");  
flash.getDocumentDOM().setFillColor(0xcc00cc);  
flash.getDocumentDOM().setFillColor(120000);  
flash.getDocumentDOM().setFillColor(null);
```

# document.setFilterProperty()

## Disponibilité

Flash 8.

## Utilisation

```
document.setFilterProperty(property, filterIndex, value)
```

## Paramètres

*property* Chaîne définissant la propriété à modifier. Les valeurs gérées sont "blurX", "blurY", "quality", "angle", "distance", "strength", "knockout", "inner", "bevelType", "color", "shadowColor" et "highlightColor".

*filterIndex* Nombre entier spécifiant l'index (en base 0) du filtre dans la liste Filtres.

*value* Nombre ou chaîne spécifiant la valeur à attribuer à la propriété de filtre spécifiée. Les valeurs acceptables dépendent de la propriété et du filtre à définir.

## Valeur renvoyée

Aucune.

## Description

Méthode ; définit la propriété de filtre spécifiée pour les objets sélectionnés qui gèrent la propriété de filtre.

## Exemple

L'exemple suivant affecte à la propriété `quality` la valeur 2 pour le second filtre (index 1) de la liste Filtres des objets sélectionnés, puis définit la propriété `shadowColor` du premier filtre de la liste Filtres pour les objets sélectionnés :

```
fl.getDocumentDOM().setFilterProperty("quality", 1, 2);  
fl.getDocumentDOM().setFilterProperty("shadowColor", 0, "#FF00FF");
```

## Voir aussi

[document.addFilter\(\)](#), [document.getFilters\(\)](#), [document.setBlendMode\(\)](#), [document.setFilters\(\)](#), [Objet Filter](#)

# document.setFilters()

## Disponibilité

Flash 8.

## Utilisation

`document.setFilters(filterArray)`

## Paramètres

*filterArray* Tableau de filtres spécifié.

## Valeur renvoyée

Aucune.

## Description

Méthode ; applique un filtre aux objets sélectionnés. Utilisez cette méthode après avoir appelé `document.getFilters()` et effectué toute modification requise dans les filtres.

## Exemple

L'exemple suivant lit la liste des filtres appliqués à l'objet sélectionné et règle la propriété `blurX` de tous les filtres Blur sur 50 :

```
var myFilters = fl.getDocumentDOM().getFilters();
for (i=0; i < myFilters.length; i++) {
    if (myFilters[i].name == "blurFilter"){
        myFilters[i].blurX = 50;
    }
}
fl.getDocumentDOM().setFilters(myFilters);
```

## Voir aussi

[document.addFilter\(\)](#), [document.getFilters\(\)](#), [document.setFilterProperty\(\)](#),  
[Objet Filter](#)

# document.setInstanceAlpha()

## Disponibilité

Flash MX 2004.

## Utilisation

`document.setInstanceAlpha(opacity)`

### Paramètres

*opacity* Nombre entier compris entre 0 (transparent) et 100 (opacité complète) qui définit la transparence de l'occurrence.

### Valeur renvoyée

Aucune.

### Description

Méthode ; définit l'opacité de l'occurrence.

### Exemple

L'exemple suivant fixe l'opacité de la teinte à 50 :

```
fl.getDocumentDOM().setInstanceAlpha(50);
```

## document.setInstanceBrightness()

### Disponibilité

Flash MX 2004.

### Utilisation

```
document.setInstanceBrightness(brightness)
```

### Paramètres

*brightness* Nombre entier définissant la luminosité, avec une valeur comprise entre -100 (noir) et 100 (blanc).

### Valeur renvoyée

Aucune.

### Description

Méthode ; définit la luminosité de l'occurrence.

### Exemple

L'exemple suivant fixe la luminosité de l'occurrence à 50 :

```
fl.getDocumentDOM().setInstanceBrightness(50);
```

# document.setInstanceTint()

## Disponibilité

Flash MX 2004.

## Utilisation

`document.setInstanceTint( color, strength )`

## Paramètres

*color* Couleur de la teinte, exprimée dans l'un des formats suivants :

- chaîne au format "#RRVVBB" ou "#RRVVBBA" ;
- nombre hexadécimal au format 0xRRVVBB ;
- entier représentant l'équivalent décimal d'un nombre hexadécimal.

Ce paramètre équivaut à sélectionner la valeur Couleur - Teinte d'un symbole dans l'inspecteur des propriétés.

*strength* Entier compris entre 0 et 100 qui définit l'opacité de la teinte.

## Valeur renvoyée

Aucune.

## Description

Méthode ; définit la teinte de l'occurrence.

## Exemple

L'exemple suivant définit comme rouge la teinte de l'occurrence sélectionnée, avec une opacité de 50 :

```
fl.getDocumentDOM().setInstanceTint(0xff0000, 50);
```

# document.setMetadata()

## Disponibilité

Flash 8.

## Utilisation

`document.setMetadata(strMetadata)`

## Paramètres

*strMetadata* Chaîne contenant les métadonnées XML associées au document. Pour plus d'informations, consultez la description ci-dessous.

## Valeur renvoyée

Une valeur booléenne : `true` en cas de réussite, `false` sinon.

## Description

Méthode ; définit les métadonnées XML du document spécifié, en remplacement des métadonnées existantes. Avant d'être enregistré, le code XML transmis dans `strMetadata` fait l'objet d'une validation et est susceptible d'être réécrit. S'il est impossible de le valider comme XML standard ou s'il enfreint des règles spécifiques, les métadonnées XML ne sont pas appliquées et la méthode renvoie `false`. (Si la valeur `false` est renvoyée, il n'est pas possible d'obtenir plus d'informations sur l'erreur.)

### REMARQUE

Même si la méthode renvoie `true`, il est possible que le code XML appliqué ne soit pas exactement identique à la chaîne transmise. Pour extraire la valeur exacte du code XML, appelez `document.getMetadata()`.

Les métadonnées possèdent un format RDF conforme à la spécification XMP. Pour plus d'informations sur RDF et XMP, consultez les sources suivantes :

- le guide RDF à l'adresse [www.w3.org/TR/rdf-primer/](http://www.w3.org/TR/rdf-primer/) ;
- la spécification RDF à l'adresse [www.w3.org/TR/1999/REC-rdf-syntax-19990222/](http://www.w3.org/TR/1999/REC-rdf-syntax-19990222/) ;
- la page d'accueil de XMP à l'adresse [www.adobe.com/products/xmp/](http://www.adobe.com/products/xmp/).

## Exemple

Les exemples suivants illustrent diverses méthodes autorisées de représentation de données identiques. Dans tous les cas de figure, à l'exception du deuxième, si les données étaient transmises à `Document.setMetadata()`, elles ne seraient pas réécrites (mis à part la suppression des sauts de ligne).

Dans le premier exemple, les métadonnées sont imbriquées dans les balises, les différents schémas étant placés dans des balises `rdf:Description` distinctes :

```
<rdf:RDF xmlns:rdf='http://www.w3.org/1999/02/22-rdf-syntax-ns#'>
  <rdf:Description rdf:about='' xmlns:dc='http://purl.org/dc/1.1/'>
    <dc:title>Simple title</dc:title>
    <dc:description>Simple description</dc:description>
  </rdf:Description>
  <rdf:Description rdf:about='' xmlns:xmp='http://ns.adobe.com/xap/1.0/'>
    <xmp:CreateDate>2004-10-12T10:29-07:00</xmp:CreateDate>
    <xmp:CreatorTool>Flash Authoring WIN 8,0,0,215</xmp:CreatorTool>
  </rdf:Description>
</rdf:RDF>
```

Dans le deuxième exemple, les métadonnées sont imbriquées dans les balises, mais tous les schémas sont regroupés dans la balise `rdf:Description`. Cet exemple comprend également des commentaires, qui ne seront pas pris en compte et seront effacés par

`Document.setMetadata()` :

```
<rdf:RDF xmlns:rdf='http://www.w3.org/1999/02/22-rdf-syntax-ns#'>
  <!-- Avant la première balise rdf:Description -->
  <rdf:Description rdf:about='' xmlns:dc='http://purl.org/dc/1.1/'>
    <dc:title>Simple title</dc:title>
    <dc:description>Simple description</dc:description>
  </rdf:Description>
  <!-- Entre les deux balises rdf:Description -->
  <rdf:Description rdf:about='' xmlns:xmp='http://ns.adobe.com/xap/1.0/'>
    <xmp:CreateDate>2004-10-12T10:29-07:00</xmp:CreateDate>
    <xmp:CreatorTool>Flash Authoring WIN 8,0,0,215</xmp:CreatorTool>
  </rdf:Description>
  <!-- Après la seconde balise rdf:Description -->
</rdf:RDF>
```

Dans le troisième exemple, les métadonnées sont imbriquées dans des attributs et tous les schémas sont regroupés dans la balise `rdf:Description` :

```
<rdf:RDF xmlns:rdf='http://www.w3.org/1999/02/22-rdf-syntax-ns#'>
  <rdf:Description rdf:about='' xmlns:dc='http://purl.org/dc/1.1/'
    dc:title='Simple title'
    dc:description='Simple description' />
  <rdf:Description rdf:about='' xmlns:xmp='http://ns.adobe.com/xap/1.0/'
    xmp:CreateDate='2004-10-12T10:29-07:00' xmp:CreatorTool='Flash Authoring
    WIN 8,0,0,215' />
</rdf:RDF>
```

## Voir aussi

[document.getMetadata\(\)](#)

# document.setMobileSettings()

## Disponibilité

Flash CS3 Professional.

## Utilisation

`document.setMobileSettings(xmlString)`

## Paramètres

*xmlString* Chaîne décrivant les paramètres XML dans un fichier FLA mobile.

## Valeur renvoyée

Valeur `true` si les paramètres ont été correctement définis, `false` dans le cas contraire.

## Description

Méthode ; définit la valeur d'une chaîne de paramètres XML dans un fichier FLA mobile. (La plupart des fichiers FLA mobiles sont pourvus d'une chaîne XML servant à décrire les paramètres dans le document.)

## Exemple

L'exemple suivant définit la chaîne de paramètres XML pour un fichier FLA mobile.

Vous remarquerez que l'exemple ci-dessous représente une seule ligne de code.

```
fl.getDocumentDOM().setMobileSettings("<? xml version='1.0' encoding='UTF-16' standalone='no' ?> <mobileSettings> <contentType id='standalonePlayer' name='Standalone Player'/> <testDevices> <testDevice id='1170' name='Generic Phone' selected='yes'/> </testDevices> <outputMsgFiltering info='no' trace='yes' warning='yes'/> <testWindowState height='496' splitterClosed='No' splitterXPos='400' width='907'/> </mobileSettings>");
```

## Voir aussi

[document.getMobileSettings\(\)](#)

# document.setOvalObjectProperty()

## Disponibilité

Flash CS3 Professional.

## Utilisation

`document.setOvalObjectProperty(propertyName, value)`

## Paramètres

*propertyName* Chaîne précisant la propriété à définir. Pour connaître les valeurs gérées, consultez la section « [Résumé des propriétés de l'objet Oval](#) », à la page 403.

*value* Valeur à affecter à la propriété. Les valeurs gérées varient en fonction de la propriété que vous spécifiez dans *propertyName*.

## Valeur renvoyée

Aucune.



## Description

Méthode ; spécifie une valeur pour une propriété spécifiée des objets Oval de type primitif.

## Exemple

Reportez-vous aux propriétés individuelles de l'[Objet Oval](#) pour les exemples.

## Voir aussi

[Objet Oval](#), `shape.isOvalObject`

# document.setPlayerVersion()

## Disponibilité

Flash CS3 Professional.

## Utilisation

```
document.setPlayerVersion(version)
```

## Paramètres

*version* Chaîne représentant la version de Flash Player ciblée par le document spécifié. Les valeurs gérées sont les suivantes : "FlashLite", "FlashLite11", "FlashLite20", "1", "2", "3", "4", "5", "6", "7", "8", et "9". Ces valeurs correspondent à la liste déroulante Version située dans la boîte de dialogue Paramètres de publication.

## Valeur renvoyée

Valeur `true` si la version de lecteur a été correctement définie, `false` dans le cas contraire.

## Description

Méthode ; définit la version de Flash Player ciblée par le document spécifié. Il s'agit de la même valeur que celle qui a été définie dans la boîte de dialogue Paramètres de publication.

## Exemple

L'exemple suivant cible Flash Lite 1.1 comme version de lecteur pour le document actif :

```
fl.getDocumentDOM().setPlayerVersion("FlashLite11");
```

## Voir aussi

`document.getPlayerVersion()`

# document.setRectangleObjectProperty()

## Disponibilité

Flash CS3 Professional.

## Utilisation

```
document.setRectangleObjectProperty(propertyName, value)
```

## Paramètres

*propertyName* Chaîne précisant la propriété à définir. Pour connaître les valeurs gérées, consultez la section « [Résumé des propriétés de l'objet Rectangle](#) », à la page 435.

*value* Valeur à affecter à la propriété. Les valeurs gérées varient en fonction de la propriété que vous spécifiez dans *propertyName*.

## Valeur renvoyée

Aucune.

## Description

Méthode ; spécifie une valeur pour une propriété spécifiée des objets Rectangle de type primitif.

## Exemple

Reportez-vous aux propriétés individuelles de l'[Objet Rectangle](#) pour les exemples.

## Voir aussi

[Objet Rectangle](#), `shape.isRectangleObject`

# document.setSelectionBounds()

## Disponibilité

Flash MX 2004 ; paramètre *bContactSensitiveSelection* intégré à Flash 8.

## Utilisation

```
document.setSelectionBounds(boundingRectangle [,  
    bContactSensitiveSelection])
```

## Paramètres

*boundingRectangle* Rectangle délimitant le nouvel emplacement et la nouvelle taille de la sélection. Pour plus d'informations sur le format de *boundingRectangle*, consultez la section [document.addNewRectangle\(\)](#).

*bContactSensitiveSelection* Une valeur booléenne indiquant si le mode de sélection Activer au contact est activé (*true*) ou non (*false*) lors de la sélection d'un objet. La valeur par défaut est *false*.

## Valeur renvoyée

Aucune.

## Description

Méthode ; déplace et redimensionne la sélection en une seule opération.

Si une valeur est transmise pour *bContactSensitiveSelection*, elle n'est valide que pour cette méthode et n'affecte pas le mode de sélection Activer au contact du document (voir [fl.contactSensitiveSelection](#)).

## Exemple

L'exemple suivant déplace la sélection aux coordonnées 10, 20 et la redimensionne à 100 sur 200 pixels :

```
var l = 10;
var t = 20;
fl.getDocumentDOM().setSelectionBounds({left:l, top:t, right:(100+l),
    bottom:(200+t)});
```

## Voir aussi

[document.selection](#), [document.setSelectionRect\(\)](#)

# document.setSelectionRect()

## Disponibilité

Flash MX 2004 ; paramètre *bContactSensitiveSelection* intégré à Flash 8.

## Utilisation

```
document.setSelectionRect(rect [, bReplaceCurrentSelection
    [, bContactSensitiveSelection]])
```

## Paramètres

*rect* Objet rectangle à définir comme sélectionné. Pour plus d'informations sur le format de *rect*, consultez la section [document.addNewRectangle\(\)](#).

*bReplaceCurrentSelection* Valeur booléenne qui indique si la méthode remplace la sélection actuelle (*true*) ou ajoute l'élément à cette dernière (*false*). La valeur par défaut est *true*.

*bContactSensitiveSelection* Une valeur booléenne indiquant si le mode de sélection Activer au contact est activé (*true*) ou non (*false*) lors de la sélection d'un objet. La valeur par défaut est *false*.

## Valeur renvoyée

Aucune.

## Description

Méthode ; trace un cadre de sélection rectangulaire sur la scène, à partir des coordonnées spécifiées. Cette opération est différente de `document.getSelectionRect()`, avec laquelle l'emplacement du rectangle est relatif à l'objet en cours de modification.

Cette méthode revient à faire glisser un rectangle à l'aide de l'outil Sélection. Pour être sélectionnée, une occurrence doit être entièrement inscrite dans ce rectangle.

Si une valeur est transmise pour *bContactSensitiveSelection*, elle n'est valide que pour cette méthode et n'affecte pas le mode de sélection Activer au contact du document (voir [fl.contactSensitiveSelection](#)).

### REMARQUE

La répétition de `setSelectionRect()` à l'aide du panneau ou de l'élément de menu Historique répète l'étape qui précède l'opération `setSelectionRect()`.

## Exemple

Dans l'exemple suivant, la seconde sélection remplace la première :

```
fl.getDocumentDOM().setSelectionRect({left:1, top:1, right:200,
  bottom:200});
fl.getDocumentDOM().setSelectionRect({left:364.0, top:203.0, right:508.0,
  bottom:434.0}, true);
```

Dans l'exemple suivant, la seconde sélection est ajoutée à la première. Cette opération est identique à un appui maintenu sur la touche Maj tout en sélectionnant un second objet.

```
fl.getDocumentDOM().setSelectionRect({left:1, top:1, right:200,
  bottom:200});
fl.getDocumentDOM().setSelectionRect({left:364.0, top:203.0, right:508.0,
  bottom:434.0}, false);
```

## Voir aussi

[document.getSelectionRect\(\)](#), [document.selection](#),  
[document.setSelectionBounds\(\)](#)

# document.setStroke()

## Disponibilité

Flash MX 2004.

## Utilisation

`document.setStroke(color, size, strokeType)`

## Paramètres

*color* Couleur du trait, exprimée dans l'un des formats suivants :

- chaîne au format "#RRVVBB" ou "#RRVVBBA" ;
- nombre hexadécimal au format 0xRRVVBB ;
- entier représentant l'équivalent décimal d'un nombre hexadécimal.

*size* Valeur en virgule flottante spécifiant la nouvelle taille de trait de la sélection.

*strokeType* Chaîne spécifiant le nouveau type de trait de la sélection. Les valeurs gérées sont les suivantes : "hairline" (filet), "solid" (uni), "dashed" (discontinu), "dotted" (pointillé), "ragged" (irrégulier), "stipple" (pointillé fin) et "hatched" (hachuré).

## Valeur renvoyée

Aucune.

## Description

Méthode ; définit la couleur, la largeur et le style des traits sélectionnés. Pour plus d'informations sur la modification du trait dans le panneau Outils ou dans l'inspecteur Propriétés, consultez la section [document.setCustomStroke\(\)](#).

## Exemple

L'exemple suivant modifie le trait en lui donnant la couleur rouge, une taille de 3,25 et le type pointillé :

```
fl.getDocumentDOM().setStroke("#ff0000", 3.25, "dashed");
```

# document.setStrokeColor()

## Disponibilité

Flash MX 2004.

## Utilisation

`document.setStrokeColor(color)`

## Paramètres

*color* Couleur du trait, exprimée dans l'un des formats suivants :

- chaîne au format "#RRRVBB" ou "#RRRVBBAA" ;
- nombre hexadécimal au format 0xRRRVBB ;
- entier représentant l'équivalent décimal d'un nombre hexadécimal.

## Valeur renvoyée

Aucune.

## Description

Méthode ; modifie la couleur de trait de la sélection avec la couleur spécifiée. Pour plus d'informations sur la modification du trait dans le panneau Outils ou dans l'inspecteur des propriétés, consultez la section [document.setCustomStroke\(\)](#).

## Exemple

Dans l'exemple suivant, les trois instructions définissent la couleur de trait à l'aide des trois différents formats de définition de la couleur.

```
flash.getDocumentDOM().setStrokeColor("#cc00cc");  
flash.getDocumentDOM().setStrokeColor(0xcc00cc);  
flash.getDocumentDOM().setStrokeColor(120000);
```

# document.setStrokeSize()

## Disponibilité

Flash MX 2004.

## Utilisation

`document.setStrokeSize(size)`

## Paramètres

*size* Valeur en virgule flottante comprise entre 0,25 et 10 qui détermine la taille du trait. La méthode ignore toute précision supérieure à deux décimales.

## Valeur renvoyée

Aucune.

## Description

Méthode ; modifie la taille de trait de la sélection avec la taille spécifiée. Pour plus d'informations sur la modification du trait dans le panneau Outils ou dans l'inspecteur des propriétés, consultez la section [document.setStrokeSize\(\)](#).

## Exemple

L'exemple suivant définit à 5 points la taille de trait de la sélection :

```
fl.getDocumentDOM().setStrokeSize(5);
```

# document.setStrokeStyle()

## Disponibilité

Flash MX 2004.

## Utilisation

```
document.setStrokeStyle(strokeType)
```

## Paramètres

*strokeType* Chaîne spécifiant le style de trait de la sélection. Les valeurs gérées sont les suivantes : "hairline" (filet), "solid" (uni), "dashed" (discontinu), "dotted" (pointillé), "ragged" (irrégulier), "stipple" (pointillé fin) et "hatched" (hachuré).

## Valeur renvoyée

Aucune.

## Description

Méthode ; modifie le style de trait de la sélection avec le style spécifié. Pour plus d'informations sur la modification du trait dans le panneau Outils ou dans l'inspecteur des propriétés, consultez la section [document.setStrokeStyle\(\)](#).

## Exemple

L'exemple suivant définit un style de trait discontinu ("dashed") pour la sélection :

```
fl.getDocumentDOM().setStrokeStyle("dashed");
```

# document.setTextRectangle()

## Disponibilité

Flash MX 2004.

## Utilisation

`document.setTextRectangle(boundingRectangle)`

## Paramètres

*boundingRectangle* Objet rectangle de texte qui spécifie la nouvelle taille dans laquelle l'élément texte doit s'insérer. Pour plus d'informations sur le format de *boundingRectangle*, consultez la section [document.addNewRectangle\(\)](#).

## Valeur renvoyée

Une valeur booléenne : `true` si la taille d'au moins un champ texte est modifiée, `false` sinon.

## Description

Méthode ; modifie le rectangle de délimitation de l'élément texte sélectionné en lui donnant la taille spécifiée. Cette méthode provoque un reformatage du texte dans le nouveau rectangle ; l'élément texte n'est ni mis à l'échelle ni transformé. Les valeurs transmises dans *boundingRectangle* sont utilisées comme suit :

- Si le texte est horizontal et statique, la méthode ne prend en compte que la largeur transmise dans *boundingRectangle*. La hauteur est automatiquement calculée pour inclure la totalité du texte.
- Si le texte est vertical, donc statique, la méthode ne prend en compte que la hauteur transmise dans *boundingRectangle*. La largeur est automatiquement calculée pour inclure la totalité du texte.
- Si le texte est dynamique ou saisi, la méthode prend en compte la largeur et la hauteur transmises dans *boundingRectangle*, et la taille du rectangle généré risque d'être supérieure à la taille requise par le texte. Cependant, si les paramètres spécifient une taille trop réduite pour le texte, la méthode ne prend en compte que la largeur transmise dans *boundingRectangle*, la hauteur étant automatiquement calculée pour inclure la totalité du texte.

## Exemple

L'exemple suivant adapte la taille du rectangle de délimitation de texte aux dimensions spécifiées :

```
fl.getDocumentDOM().setTextRectangle({left:0, top:0, right:50, bottom:200})
```



# document.setTextSelection()

## Disponibilité

Flash MX 2004.

## Utilisation

```
document.setTextSelection(startIndex, endIndex)
```

## Paramètres

*startIndex* Nombre entier spécifiant la position du premier caractère à sélectionner.  
La position du premier caractère est zéro.

*endIndex* Nombre entier spécifiant la position du dernier caractère de la sélection,  
*endIndex* non compris. La position du premier caractère est zéro.

## Valeur renvoyée

Une valeur booléenne : `true` en cas de réussite de la sélection de texte, `false` sinon.

## Description

Méthode ; définit la sélection de texte du champ texte sélectionné avec les valeurs spécifiées dans *startIndex* et *endIndex*. Le mode de modification de texte est activé s'il ne l'était pas déjà.

## Exemple

L'exemple suivant sélectionne le texte du 6e au 25e caractère :

```
fl.document.setTextSelection(5, 25);
```

# document.setTextString()

## Disponibilité

Flash MX 2004.

## Utilisation

```
document.setTextString(text [, startIndex [, endIndex]])
```

## Paramètres

*text* Chaîne de caractères à insérer dans le champ texte.

*startIndex* Nombre entier spécifiant la position du premier caractère à remplacer.  
La position du premier caractère est zéro. Ce paramètre est facultatif.

*endIndex* Nombre entier spécifiant la position du dernier caractère à remplacer.  
Ce paramètre est facultatif.

## Valeur renvoyée

Une valeur booléenne : `true` si le texte d'au moins un champ texte est modifié, `false` sinon.

## Description

Méthode ; insère une chaîne de texte. Si les paramètres facultatifs ne sont pas transmis, la méthode remplace la sélection de texte existante. Si l'objet `Text` n'est pas en cours de modification, toute la chaîne de texte est remplacée. Si seul le paramètre `startIndex` est transmis, la chaîne transmise est insérée à cette position. Si `startIndex` et `endIndex` sont transmis, la chaîne transmise remplace le segment de texte compris entre `startIndex` et le dernier caractère avant `endIndex`.

## Exemple

L'exemple suivant remplace la sélection de texte actuelle par la chaîne « Hello World » :

```
var success = fl.getDocumentDOM().setTextString("Hello World!");
```

L'exemple suivant insère « hello » à la position 6 de la sélection de texte actuelle :

```
var pass = fl.getDocumentDOM().setTextString("hello", 6);
```

L'exemple suivant insère « Howdy » entre la position 2 et la position 7 (non comprise) de la sélection de texte actuelle :

```
var ok = fl.getDocumentDOM().setTextString("Howdy", 2, 7);
```

## Voir aussi

[document.getTextString\(\)](#)

# document.setTransformationPoint()

## Disponibilité

Flash MX 2004.

## Utilisation

```
document.setTransformationPoint( transformationPoint )
```

## Paramètres

*transformationPoint* Point (par exemple, {x:10, y:20}, sachant que x et y sont des nombres à virgule flottante) spécifiant des valeurs pour le point de transformation de chacun des éléments sélectionnés :

- **Formes** : *transformationPoint* est défini par rapport au document (0,0 étant le coin supérieur gauche de la scène).

- Symboles : *transformationPoint* est défini par rapport au point d'alignement du symbole (0,0 étant situé au point d'alignement).
- Texte : *transformationPoint* est défini par rapport au champ texte (0,0 étant le coin supérieur gauche du champ texte).
- Bitmaps/vidéos : *transformationPoint* est défini par rapport au bitmap/vidéo (0,0 étant le coin supérieur gauche du bitmap ou de la vidéo).
- Objets de dessin, ovales et rectangles de type primitif, groupes : *transformationPoint* est défini par rapport au document (0,0 étant le coin supérieur gauche de la scène). Pour définir *transformationPoint* par rapport au point central de l'objet, de la primitive ou du groupe, utilisez `element.setTransformationPoint()`.

### Valeur renvoyée

Aucune.

### Description

Méthode ; définit la position du point de transformation de la sélection actuelle.

### Exemple

L'exemple suivant définit l'emplacement du point de transformation de la sélection aux coordonnées 100, 200 :

```
fl.getDocumentDOM().setTransformationPoint({x:100, y:200});
```

### Voir aussi

`document.getTransformationPoint()`, `element.setTransformationPoint()`

## document.silent

### Disponibilité

Flash MX 2004.

### Utilisation

`document.silent`

### Description

Propriété ; valeur booléenne indiquant si l'objet est accessible. Cette propriété est équivalente à l'inverse du paramètre Rendre l'animation accessible du panneau Accessibilité. Ainsi, si `document.silent` est `true`, son effet est similaire à la désactivation de l'option Rendre l'animation accessible. Si le paramètre est `false`, son effet est similaire à l'activation de l'option Rendre l'animation accessible.

## Exemple

L'exemple suivant fixe la valeur de la variable `isSilent` avec la valeur de la propriété `silent` :

```
var isSilent = fl.getDocumentDOM().silent;
```

L'exemple suivant règle la propriété `silent` sur `false` indiquant ainsi que le document est accessible :

```
fl.getDocumentDOM().silent = false;
```

# document.skewSelection()

## Disponibilité

Flash MX 2004.

## Utilisation

```
document.skewSelection(xSkew, ySkew [, whichEdge])
```

## Paramètres

*xSkew* Valeur en virgule flottante spécifiant la valeur *x* d'inclinaison, exprimée en degrés.

*ySkew* Valeur en virgule flottante spécifiant la valeur *y* d'inclinaison, exprimée en degrés.

*whichEdge* Chaîne indiquant le côté sur lequel la transformation doit être appliquée.

Si ce paramètre est omis, l'inclinaison est appliquée au niveau du point de transformation.

Les valeurs gérées sont "top center" (centre haut), "right center" (centre droite), "bottom center" (centre bas) et "left center" (centre gauche). Ce paramètre est facultatif.

## Valeur renvoyée

Aucune.

## Description

Méthode ; effectue sur la sélection une inclinaison de la valeur spécifiée. L'effet est identique à celui de l'outil Transformation libre utilisé pour une inclinaison de l'objet.

## Exemple

Les exemples suivants inclinent l'objet sélectionné de 2° verticalement et 1,5° horizontalement. Le second exemple transforme l'objet sur le bord centre haut :

```
flash.getDocumentDOM().skewSelection(2.0, 1.5);  
flash.getDocumentDOM().skewSelection(2.0, 1.5, "top center");
```

# document.smoothSelection()

## Disponibilité

Flash MX 2004.

## Utilisation

```
document.smoothSelection()
```

## Paramètres

Aucun.

## Valeur renvoyée

Aucune.

## Description

Méthode ; lisse la courbe de chaque contour de remplissage ou segment incurvé sélectionné. Cette méthode effectue la même opération que le bouton Lisser dans le panneau Outils.

## Exemple

L'exemple suivant lisse la courbe de la sélection :

```
fl.getDocumentDOM().smoothSelection();
```

# document.space()

## Disponibilité

Flash MX 2004.

## Utilisation

```
document.space(direction [, bUseDocumentBounds])
```

## Paramètres

*direction* Chaîne spécifiant la direction dans laquelle doivent être espacés les objets de la sélection. Les valeurs gérées sont "horizontal" ou "vertical".

*bUseDocumentBounds* Valeur booléenne. Si elle est *true*, la méthode espace les objets sélectionnés sur les limites du document. Sinon, la méthode utilise les limites des objets sélectionnés. La valeur par défaut est *false*. Ce paramètre est facultatif.

## Valeur renvoyée

Aucune.

## Description

Méthode ; espace régulièrement les objets sélectionnés.

## Exemple

L'exemple suivant espace les objets horizontalement par rapport aux limites de la scène :

```
fl.getDocumentDOM().space("horizontal",true);
```

L'exemple suivant espace les objets entre eux sur le plan horizontal :

```
fl.getDocumentDOM().space("horizontal");
```

L'exemple suivant espace les objets horizontalement les uns par rapport aux autres, mais avec *useDocumentBounds* réglé explicitement sur *false* :

```
fl.getDocumentDOM().space("horizontal",false);
```

## Voir aussi

[document.getAlignToDocument\(\)](#), [document.setAlignToDocument\(\)](#)

# document.straightenSelection()

## Disponibilité

Flash MX 2004.

## Utilisation

```
document.straightenSelection()
```

## Paramètres

Aucun.

## Valeur renvoyée

Aucune.

## Description

Méthode ; redresse les traits sélectionnés. Cette méthode est l'équivalent du bouton Redresser dans le panneau Outils.

## Exemple

L'exemple suivant redresse la courbe de la sélection :

```
fl.getDocumentDOM().straightenSelection();
```

# document.swapElement()

## Disponibilité

Flash MX 2004.

## Utilisation

```
document.swapElement(name)
```

## Paramètres

*name* Chaîne spécifiant le nom de l'élément de bibliothèque à utiliser.

## Valeur renvoyée

Aucune.

## Description

méthode ; intervertit la sélection actuelle avec la sélection spécifiée. La sélection doit contenir un graphisme, un bouton, un clip, une vidéo ou un bitmap. Un message d'erreur est affiché s'il n'existe pas de sélection ou si l'objet indiqué est introuvable.

## Exemple

L'exemple suivant intervertit la sélection avec l'objet `Symbol 1` de la bibliothèque :

```
fl.getDocumentDOM().swapElement('Symbol 1');
```

# document.swapStrokeAndFill()

## Disponibilité

Flash 8.

## Utilisation

```
document.swapStrokeAndFill();
```

## Paramètres

Aucun.

## Valeur renvoyée

Aucune.

## Description

Méthode ; permute les couleurs de trait et de remplissage.

## Exemple

L'exemple suivant permute les couleurs de trait et de remplissage dans le document actuel :

```
fl.getDocumentDOM().swapStrokeAndFill();
```

# document.synchronizeWithHeadVersion()

## Disponibilité

Flash CS3 Professional.

## Utilisation

```
fl.document.getDocumentDOM().synchronizeWithHeadVersion().swapStrokeAndFill();
```

## Paramètres

Aucun.

## Valeur renvoyée

Valeur booléenne `true` si la synchronisation du fichier spécifié s'est déroulée correctement sur le serveur Version Cue, sinon `false`.

## Description

Méthode ; synchronise le document spécifié avec la version la plus récente trouvée sur le serveur Version Cue et consigne les éventuelles erreurs dans le panneau Sortie.

Cette méthode fonctionne uniquement avec les documents qui sont ouverts à ce moment-là. Pour récupérer la version la plus récente d'un fichier qui n'est pas actuellement ouvert, utilisez [fl.downloadLatestVersion\(\)](#).

## Exemple

L'exemple suivant synchronise le document actif avec la version stockée sur le serveur Version Cue :

```
fl.getDocumentDOM().synchronizeWithHeadVersion();
```

## Voir aussi

```
document.canSaveAVersion(), fl.downloadLatestVersion(),  
document.revertToLastVersion(), document.saveAVersion(),  
fl.synchronizeDocumentWithHeadVersion()
```



# document.testMovie()

## Disponibilité

Flash MX 2004.

## Utilisation

```
document.testMovie()
```

## Paramètres

Aucun.

## Valeur renvoyée

Aucune.

## Description

Méthode ; exécute une opération Tester l'animation sur le document.

## Exemple

L'exemple suivant teste l'animation du document actif :

```
fl.getDocumentDOM().testMovie();
```

## Voir aussi

[document.canTestMovie\(\)](#), [document.testScene\(\)](#)

# document.testScene()

## Disponibilité

Flash MX 2004.

## Utilisation

```
document.testScene()
```

## Paramètres

Aucun.

## Valeur renvoyée

Aucune.

## Description

Méthode ; exécute une opération Tester la séquence sur la séquence active dans le document.

## Exemple

L'exemple suivant teste la séquence active du document :

```
fl.getDocumentDOM().testScene();
```

## Voir aussi

[document.canTestScene\(\)](#), [document.testMovie\(\)](#)

# document.timelines

## Disponibilité

Flash MX 2004.

## Utilisation

`document.timelines`

## Description

Propriété en lecture seule ; tableau des objets Timeline (scénario) (voir [Objet Timeline](#)).

## Exemple

L'exemple suivant lit le tableau des objets de scénario du document actif et affiche leurs noms dans le panneau Sortie :

```
var i = 0;
var curTimelines = fl.getDocumentDOM().timelines;
while(i < fl.getDocumentDOM().timelines.length){
    alert(curTimelines[i].name);
    ++i;
}
```

## Voir aussi

[document.currentTimeline](#), [document.getTimeline\(\)](#)

# document.traceBitmap()

## Disponibilité

Flash MX 2004.

## Utilisation

`document.traceBitmap(threshold, minimumArea, curveFit, cornerThreshold)`

## Paramètres

*threshold* Nombre entier limitant le nombre de couleurs du bitmap faisant l'objet du tracer. Les valeurs gérées sont des entiers compris entre 0 et 500.

*minimumArea* Nombre entier spécifiant le rayon, indiqué en pixels. Les valeurs gérées sont des entiers compris entre 1 et 1 000.

*curveFit* Chaîne spécifiant le niveau de lissage des contours. Les valeurs gérées sont "pixels", "very tight" (très serré), "tight" (serré), "normal", "smooth" (lisse) et "very smooth" (très lisse).

*cornerThreshold* Chaîne dont l'effet est similaire à celui de *curveFit*, mais concernant les angles de l'image bitmap. Les valeurs gérées sont "many corners" (nombreux angles), "normal" et "few corners" (peu d'angles).

## Valeur renvoyée

Aucune.

## Description

Méthode ; effectue une opération de tracer de bitmap sur la sélection en cours. Cette méthode est l'équivalent de l'option de menu Modification > Bitmap > Tracer le bitmap.

## Exemple

L'exemple suivant effectue un tracer du bitmap sélectionné sur la base des paramètres spécifiés :

```
fl.getDocumentDOM().traceBitmap(0, 500, 'normal', 'normal');
```

# document.transformSelection()

## Disponibilité

Flash MX 2004.

## Utilisation

```
document.transformSelection(a, b, c, d)
```

## Paramètres

- a* Nombre en virgule flottante spécifiant l'élément (0,0) de la matrice de transformation.
- b* Nombre en virgule flottante spécifiant l'élément (0,1) de la matrice de transformation.
- c* Nombre en virgule flottante spécifiant l'élément (1,0) de la matrice de transformation.
- d* Nombre en virgule flottante spécifiant l'élément (1,1) de la matrice de transformation.

## Valeur renvoyée

Aucune.

## Description

Méthode ; effectue une transformation générale de la sélection, en appliquant la matrice passée en argument. Pour plus d'informations, consultez la propriété [element.matrix](#).

## Exemple

L'exemple suivant étire la sélection d'un facteur 2 dans la direction x :

```
fl.getDocumentDOM().transformSelection(2.0, 0.0, 0.0, 1.0);
```

# document.unGroup()

## Disponibilité

Flash MX 2004.

## Utilisation

```
document.unGroup()
```

## Paramètres

Aucun.

## Valeur renvoyée

Aucune.

## Description

Méthode ; dissocie la sélection.

## Exemple

L'exemple suivant dissocie les éléments de la sélection :

```
fl.getDocumentDOM().unGroup();
```

## Voir aussi

[document.group\(\)](#)

# document.union()

## Disponibilité

Flash 8.

## Utilisation

```
document.union()
```

## Paramètres

Aucun.

## Valeur renvoyée

Une valeur booléenne : `true` en cas de réussite, `false` sinon.

## Description

Méthode ; combine toutes les formes sélectionnées sous forme d'un objet de dessin.

## Exemple

L'exemple suivant combine toutes les formes sélectionnées sous forme d'un objet de dessin :

```
fl.getDocumentDOM().union();
```

## Voir aussi

[document.crop\(\)](#), [document.deleteEnvelope\(\)](#), [document.intersect\(\)](#),  
[document.punch\(\)](#), [shape.isDrawingObject](#)

# document.unlockAllElements()

## Disponibilité

Flash MX 2004.

## Utilisation

```
document.unlockAllElements()
```

## Paramètres

Aucun.

## Valeur renvoyée

Aucune.

## Description

Méthode ; déverrouille tous les éléments verrouillés dans l'image sélectionnée.

## Exemple

L'exemple suivant déverrouille tous les éléments verrouillés dans l'image active :

```
fl.getDocumentDOM().unlockAllElements();
```

## Voir aussi

[element.locked](#)

# document.viewMatrix

## Disponibilité

Flash MX 2004.

## Utilisation

```
document.viewMatrix
```

## Description

Propriété en lecture seule ; objet Matrix. La propriété `viewMatrix` permet de passer de l'espace de l'objet à l'espace du document lorsque celui-ci est en mode de modification. Les coordonnées de la souris, telles qu'elles sont reçues par un outil, sont relatives à l'objet en cours de modification. Voir la section [Objet Matrix](#).

Par exemple, si vous créez un symbole, double-cliquez dessus pour le modifier et dessinez à l'aide de l'outil Polygone, le point (0,0) se trouve sur le point d'alignement du symbole. Toutefois, l'objet `drawingLayer` attend des valeurs situées dans l'espace du document, donc si vous tracez une ligne à partir des coordonnées (0,0) avec `drawingLayer`, ces coordonnées seront relatives au coin supérieur gauche de la scène. La propriété `viewMatrix` permet de passer de l'espace de l'objet en cours de modification à l'espace du document.

### Exemple

L'exemple suivant lit la valeur de la propriété `viewMatrix` :

```
var mat = fl.getDocumentDOM().viewMatrix;
```

## document.width

### Disponibilité

Flash MX 2004.

### Utilisation

```
document.width
```

### Description

Propriété ; nombre entier indiquant la largeur en pixels du document (scène).

### Exemple

L'exemple suivant fixe la largeur de la scène à 400 pixels :

```
fl.getDocumentDOM().width= 400;
```

### Voir aussi

[document.height](#)

## document.xmlPanel()

### Disponibilité

Flash MX 2004.

### Utilisation

```
document.xmlPanel(fileURI)
```

### Paramètres

*fileURI* Chaîne de type « fichier:/// URI » qui spécifie le chemin d'accès du fichier XML définissant les contrôles apparaissant dans le panneau. Le chemin complet est nécessaire.

## Valeur renvoyée

Un objet dont les propriétés sont définies pour tous les contrôles définis dans le fichier XML. Toutes les propriétés sont renvoyées sous forme de chaînes. L'objet renvoyé comporte une propriété prédéfinie appelée "dismiss" qui possède la valeur de chaîne "accept" ou "cancel".

## Description

Méthode ; transmet une boîte de dialogue XMLUI. Voir la section [fl.xmlui](#).

## Exemple

L'exemple suivant charge le fichier Test.xml et affiche chacune de ses propriétés :

```
var obj = fl.getDocumentDOM().xmlPanel(fl.configURI + "Commands/Test.xml");
for (var prop in obj) {
    fl.trace("property " + prop + " = " + obj[prop]);
}
```

# document.zoomFactor

## Disponibilité

Flash 8.

## Utilisation

`document.zoomFactor`

## Description

Propriété ; spécifie le pourcentage de zoom de la scène en contexte de création. La valeur 1 signifie zoom à 100 %, 8 signifie 800 %, 0,5 signifie 50 %, etc.

## Exemple

L'exemple suivant règle le facteur de zoom de la scène sur 200 %.

```
fl.getDocumentDOM().zoomFactor = 2;
```



# Objet drawingLayer

## Disponibilité

Flash MX 2004.

## Description

L'objet drawingLayer est accessible en JavaScript sous forme d'un enfant de l'objet flash. L'objet drawingLayer est utilisé par les outils extensibles lorsque l'utilisateur dessine un tracé temporaire à l'aide de la souris, par exemple pour créer un rectangle de sélection. Il est conseillé d'appeler `drawingLayer.beginFrame()` avant d'appeler toute autre méthode de drawingLayer.

## Résumé des méthodes de l'objet drawingLayer

Les méthodes suivantes sont disponibles avec l'objet drawingLayer :

Méthodes	Description
<code>drawingLayer.beginDraw()</code>	Active le mode Dessin de Flash.
<code>drawingLayer.beginFrame()</code>	Efface ce qui a été dessiné précédemment avec drawingLayer et prépare le programme pour d'autres commandes de dessin.
<code>drawingLayer.cubicCurveTo()</code>	Dessine une courbe cubique à partir de l'emplacement actuel du curseur, en utilisant les paramètres comme coordonnées du segment cubique.
<code>drawingLayer.curveTo()</code>	Dessine un segment de courbe quadratique commençant à l'emplacement actuel du curseur et se terminant au point spécifié.
<code>drawingLayer.drawPath()</code>	Dessine le trajet spécifié.
<code>drawingLayer.endDraw()</code>	Désactive le mode Dessin.
<code>drawingLayer.endFrame()</code>	Signale la fin d'un groupe de commandes de dessin.
<code>drawingLayer.lineTo()</code>	Dessine une ligne à partir de la position du curseur et jusqu'au point (x,y).
<code>drawingLayer.moveTo()</code>	Définit la position de dessin actuelle.
<code>drawingLayer.newPath()</code>	Renvoie un nouvel <a href="#">Objet Path</a> .
<code>drawingLayer.setColor()</code>	Définit la couleur des objets dessin à venir.

# drawingLayer.beginDraw()

## Disponibilité

Flash MX 2004.

## Utilisation

`drawingLayer.beginDraw([persistentDraw])`

## Paramètres

*persistentDraw* Valeur Booléenne (facultative). Si sa valeur est `true`, elle indique que le dessin dans le dernier cadre reste sur la scène jusqu'au prochain appel de `beginDraw()` ou `beginFrame()`. (Dans ce contexte, *frame* fait référence à l'espace du dessin, pas aux images dans le scénario.) Par exemple, lorsque l'utilisateur trace un rectangle, il peut avoir un aperçu du contour de cette forme pendant qu'il déplace la souris. Si vous souhaitez que ce contour reste visible lorsque l'utilisateur relâche le bouton de la souris, donnez au paramètre *persistentDraw* la valeur `true`.

## Valeur renvoyée

Aucune.

## Description

Méthode ; active le mode Dessin de Flash. Le mode dessin est utilisé pour les tracés temporaires avec le bouton de la souris. En général, cette méthode n'est utilisée que lors de la création d'outils extensibles.

## Exemple

L'exemple suivant active le mode Dessin de Flash :

```
fl.drawingLayer.beginDraw();
```

# drawingLayer.beginFrame()

## Disponibilité

Flash MX 2004.

## Utilisation

`drawingLayer.beginFrame()`

## Paramètres

Aucun.

## Valeur renvoyée

Aucune.

## Description

Méthode ; efface ce qui a été dessiné précédemment avec `drawingLayer` et prépare le programme pour d'autres commandes de dessin. Cette méthode doit être appelée après `drawingLayer.beginDraw()`. Tout ce qui est tracé entre un appel de `drawingLayer.beginFrame()` et un appel de `drawingLayer.endFrame()` reste sur la scène jusqu'aux appels suivants de `beginFrame()` et `endFrame()`. (Dans ce contexte, *frame* fait référence à l'espace du dessin, pas aux images dans le scénario.) En général, cette méthode n'est utilisée que lors de la création d'outils extensibles. Voir [drawingLayer.beginDraw\(\)](#).

# drawingLayer.cubicCurveTo()

## Disponibilité

Flash MX 2004.

## Utilisation

```
drawingLayer.cubicCurveTo(x1Ctrl, y1Ctrl, x2Ctrl, y2Ctrl, xEnd, yEnd)
```

## Paramètres

*x1Ctrl* Valeur en virgule flottante indiquant la coordonnée *x* du premier point de contrôle.

*y1Ctrl* Valeur en virgule flottante indiquant la coordonnée *y* du premier point de contrôle.

*x2Ctrl* Valeur en virgule flottante indiquant la coordonnée *x* du point de contrôle central.

*y2Ctrl* Valeur en virgule flottante indiquant la coordonnée *y* du point de contrôle central.

*xEnd* Valeur en virgule flottante indiquant la coordonnée *x* du point de contrôle final.

*yEnd* Valeur en virgule flottante indiquant la coordonnée *y* du point de contrôle final.

## Valeur renvoyée

Aucune.

## Description

Méthode ; dessine une courbe cubique à partir de l'emplacement actuel du curseur, en utilisant les paramètres comme coordonnées du segment cubique. En général, cette méthode est uniquement utilisée lors de la création d'outils extensibles.

## Exemple

L'exemple suivant trace une courbe cubique en utilisant les points de contrôle spécifiés :

```
fl.drawingLayer.cubicCurveTo(0, 0, 1, 1, 2, 0);
```

## drawingLayer.curveTo()

### Disponibilité

Flash MX 2004.

### Utilisation

```
drawingLayer.curveTo(xCtl, yCtl, xEnd, yEnd)
```

### Paramètres

*xCtl* Valeur en virgule flottante indiquant la coordonnée *x* du premier point de contrôle.

*yCtl* Valeur en virgule flottante indiquant la coordonnée *y* du premier point de contrôle.

*xEnd* Valeur en virgule flottante indiquant la coordonnée *x* du point de contrôle final.

*yEnd* Valeur en virgule flottante indiquant la coordonnée *y* du point de contrôle final.

### Valeur renvoyée

Aucune.

### Description

Méthode ; dessine un segment de courbe quadratique commençant à l'emplacement actuel du curseur et se terminant au point spécifié. En général, cette méthode n'est utilisée que lors de la création d'outils extensibles.

### Exemple

L'exemple suivant trace une courbe quadratique en utilisant les points de contrôle spécifiés :

```
fl.drawingLayer.curveTo(0, 0, 2, 0);
```

## drawingLayer.drawPath()

### Disponibilité

Flash MX 2004.

### Utilisation

```
drawingLayer.drawPath(path)
```

### Paramètres

*path* L'[Objet Path](#) à dessiner.

### Valeur renvoyée

Aucune.

### Description

Méthode ; dessine le trajet spécifié par le paramètre *path*. En général, cette méthode n'est utilisée que lors de la création d'outils extensibles.

### Exemple

L'exemple suivant dessine le trajet spécifié par l'objet Path nommé *gamePath* :

```
fl.drawingLayer.drawPath(gamePath);
```

## drawingLayer.endDraw()

### Disponibilité

Flash MX 2004.

### Utilisation

```
drawingLayer.endDraw()
```

### Paramètres

Aucun.

### Valeur renvoyée

Aucune.

### Description

Méthode ; quitte le mode dessin. Le mode dessin est utilisé pour les tracés temporaires avec le bouton de la souris. En général, cette méthode n'est utilisée que lors de la création d'outils extensibles.

### Exemple

L'exemple suivant quitte le mode dessin :

```
fl.drawingLayer.endDraw();
```

## drawingLayer.endFrame()

### Disponibilité

Flash MX 2004.

### Utilisation

```
drawingLayer.endFrame()
```

## Paramètres

Aucun.

## Valeur renvoyée

Aucune.

## Description

Méthode ; signale la fin d'un groupe de commandes de dessin. Un groupe de commandes de dessin représente tout ce qui est tracé entre les appels de `drawingLayer.beginFrame()` et `drawingLayer.endFrame()`. L'appel suivant de `drawingLayer.beginFrame()` efface tout ce qui a été tracé par ce groupe de commandes de dessin. En général, cette méthode n'est utilisée que lors de la création d'outils extensibles.

# drawingLayer.lineTo()

## Disponibilité

Flash MX 2004.

## Utilisation

```
drawingLayer.lineTo(x, y)
```

## Paramètres

*x* Valeur en virgule flottante indiquant la coordonnée *x* du point de contrôle final de la ligne à tracer.

*y* Valeur en virgule flottante indiquant la coordonnée *y* du point de contrôle final de la ligne à tracer.

## Valeur renvoyée

Aucune.

## Description

Méthode ; dessine une ligne à partir de la position du curseur et jusqu'au point (*x*,*y*). En général, cette méthode n'est utilisée que lors de la création d'outils extensibles.

## Exemple

L'exemple suivant dessine une ligne à partir de la position du curseur et jusqu'au point (20,30) :

```
fl.drawingLayer.lineTo(20, 30);
```

# drawingLayer.moveTo()

## Disponibilité

Flash MX 2004.

## Utilisation

```
drawingLayer.moveTo(x, y)
```

## Paramètres

*x* Valeur en virgule flottante indiquant la coordonnée *x* de la position de début du dessin.

*y* Valeur en virgule flottante indiquant la coordonnée *y* de la position de début du dessin.

## Valeur renvoyée

Aucune.

## Description

Méthode ; définit la position de dessin actuelle. En général, cette méthode n'est utilisée que lors de la création d'outils extensibles.

## Exemple

L'exemple suivant définit la position de début du dessin au point (10,15) :

```
fl.drawingLayer.moveTo(10, 15);
```

# drawingLayer.newPath()

## Disponibilité

Flash MX 2004.

## Utilisation

```
drawingLayer.newPath()
```

## Paramètres

Aucun.

## Valeur renvoyée

Un objet Path (trajet).

## Description

Méthode ; renvoie un nouvel objet Path. En général, cette méthode n'est utilisée que lors de la création d'outils extensibles. Voir [Objet Path](#).

## Exemple

L'exemple suivant renvoie un nouvel objet Path :

```
fl.drawingLayer.newPath();
```

# drawingLayer.setColor()

## Disponibilité

Flash MX 2004.

## Utilisation

```
drawingLayer.setColor(color)
```

## Paramètres

*color* Définit la couleur des objets dessin à venir, dans l'un des formats suivants :

- Chaîne au format "#RRGGBB" ou "#RRGGBBAA"
- Nombre hexadécimal au format 0xRRGGBB
- Entier représentant l'équivalent décimal d'un nombre hexadécimal

## Valeur renvoyée

Aucune.

## Description

Méthode ; définit la couleur des objets dessin à venir. S'applique uniquement aux données persistantes. Pour utiliser cette méthode, le paramètre transmis à `drawingLayer.beginDraw()` doit être réglé sur `true`. En général, cette méthode n'est utilisée que lors de la création d'outils extensibles. Voir [drawingLayer.beginDraw\(\)](#).

## Exemple

L'exemple suivant trace une ligne rouge sur la scène :

```
fl.drawingLayer.beginDraw( true );
fl.drawingLayer.beginFrame();
fl.drawingLayer.setColor( "#ff0000" );
fl.drawingLayer.moveTo(0,0);
fl.drawingLayer.lineTo(100,100);
fl.drawingLayer.endFrame();
fl.drawingLayer.endDraw();
```



# Objet Edge

## Disponibilité

Flash MX 2004.

## Description

L'objet Edge représente les limites d'une forme sur la scène.

## Résumé des méthodes de l'objet Edge

Les méthodes suivantes sont disponibles pour l'objet Edge :

Méthode	Description
<code>edge.getControl()</code>	Lit un objet point situé à l'emplacement du point de contrôle spécifié du bord.
<code>edge.getHalfEdge()</code>	Renvoie un <a href="#">Objet halfEdge</a> .
<code>edge.setControl()</code>	Définit la position du point de contrôle du bord.
<code>edge.splitEdge()</code>	Fractionne le bord en deux parties.

## Résumé des propriétés de l'objet Edge

Les propriétés suivantes sont disponibles pour l'objet Edge :

Propriété	Description
<code>edge.id</code>	En lecture seule ; entier représentant un identifiant unique pour le bord.
<code>edge.isLine</code>	Lecture seule ; entier d'une valeur de 0 ou 1.

## edge.getControl()

### Disponibilité

Flash MX 2004.

### Utilisation

```
edge.getControl(i)
```

### Paramètres

*i* Entier qui spécifie les points de contrôle du bord à renvoyer. Indiquez 0 pour le premier point de contrôle, 1 pour le point de contrôle central ou 2 pour le point de contrôle final. Si la propriété `edge.isLine` a la valeur `true`, le point de contrôle central est fixé sur le point médian du segment qui relie les points de contrôle de début et de fin.

### Valeur renvoyée

Le point de contrôle spécifié.

### Description

Méthode ; lit un objet point situé à l'emplacement du point de contrôle spécifié du bord.

### Exemple

L'exemple suivant enregistre dans la variable `pt` le premier point de contrôle de la forme spécifiée :

```
var shape = fl.getDocumentDOM().selection[0];  
var pt = shape.edges[0].getControl(0);
```

## edge.getHalfEdge()

### Disponibilité

Flash MX 2004.

### Utilisation

```
edge.getHalfEdge(index)
```

### Paramètres

*index* Entier qui spécifie le demi-bord à renvoyer. La valeur de *index* doit être 0 pour le premier demi-bord ou 1 pour le second demi-bord.

### Valeur renvoyée

Un objet `HalfEdge`.

## Description

Méthode ; renvoie un [Objet halfEdge](#).

## Exemple

L'exemple suivant enregistre dans les variables hEdge0 et hEdge1 le demi-bord du bord spécifié :

```
var shape = fl.getDocumentDOM().selection[0];
var edge = shape.edges[0];
var hEdge0 = edge.getHalfEdge(0);
var hEdge1 = edge.getHalfEdge(1);
```

## edge.id

### Disponibilité

Flash MX 2004.

### Utilisation

edge.id

### Description

Propriété en lecture seule ; entier représentant un identifiant unique pour le bord.

### Exemple

L'exemple suivant enregistre dans la variable my\_shape\_id un identificateur unique pour le bord spécifié :

```
var shape = fl.getDocumentDOM().selection[0];
var my_shape_id = shape.edges[0].id;
```

## edge.isLine

### Disponibilité

Flash MX 2004.

### Utilisation

edge.isLine

### Description

Propriété en lecture seule ; entier ayant la valeur 0 ou 1. La valeur 1 indique que le bord est une ligne droite. Dans ce cas, le point de contrôle central coupe en deux parts égales la ligne qui relie les deux points d'extrémité.

## Exemple

L'exemple suivant détermine si le bord spécifié est une ligne droite, et affiche la valeur 1 (si c'est une ligne droite) ou 0 (si ce n'est pas le cas) dans le panneau de sortie :

```
var shape = fl.getDocumentDOM().selection[0];  
fl.trace(shape.edges[0].isLine);
```

# edge.setControl()

## Disponibilité

Flash MX 2004.

## Utilisation

```
edge.setControl(index, x, y)
```

## Paramètres

*index* Nombre entier qui spécifie le point de contrôle à définir. Les valeurs 0, 1 ou 2 permettent respectivement de spécifier les points de contrôle de début, de milieu et de fin.

*x* Valeur en virgule flottante indiquant la coordonnée horizontale du point de contrôle. Si la scène est en mode de modification ou de modification en place, les coordonnées du point sont calculées par rapport à l'objet en cours de modification. Sinon, les coordonnées du point sont relatives à la scène.

*y* Valeur en virgule flottante indiquant la coordonnée verticale du point de contrôle. Si la scène est en mode de modification ou de modification en place, les coordonnées du point sont calculées par rapport à l'objet en cours de modification. Sinon, les coordonnées du point sont relatives à la scène.

## Valeur renvoyée

Aucune.

## Description

Méthode ; définit la position du point de contrôle du bord. Il est nécessaire d'appeler `shape.beginEdit()` avant d'utiliser cette méthode. Voir [shape.beginEdit\(\)](#).

## Exemple

L'exemple suivant donne au point de contrôle de début du bord spécifié les coordonnées 0, 1 :

```
x = 0; y = 1;  
var shape = fl.getDocumentDOM().selection[0];  
shape.beginEdit();  
shape.edges[0].setControl(0, x, y);  
shape.endEdit();
```

# edge.splitEdge()

## Disponibilité

Flash MX 2004.

## Utilisation

```
edge.splitEdge(t)
```

## Paramètres

*t* Valeur en virgule flottante comprise entre 0 et 1 indiquant le point de fractionnement du bord. La valeur 0 représente un point d'extrémité, et 1 l'autre point. Par exemple, la valeur 0,5 fractionne le bord en son milieu, c'est-à-dire en son centre exact pour une ligne. Si le bord forme une courbe, 0,5 représente le milieu paramétrique de la courbe.

## Valeur renvoyée

Aucune.

## Description

Méthode ; fractionne le bord en deux parties. Il est nécessaire d'appeler [shape.beginEdit\(\)](#) avant d'utiliser cette méthode.

## Exemple

L'exemple suivant fractionne le bord spécifié en deux moitiés :

```
var shape = fl.getDocumentDOM().selection[0];
shape.beginEdit()
shape.edges[0].splitEdge( 0.5 );
shape.endEdit()
```

# Objet Effect

## Disponibilité

Flash MX 2004.

## Description

Cet objet est un descripteur d'effet unique. Les propriétés `fl.activeEffect` et `fl.effects` contiennent ce type d'objet. L'objet Effect représente une occurrence d'un effet dans le scénario. Consultez les sections [fl.activeEffect](#) et [fl.effects](#).

## Résumé des propriétés de l'objet Effect

Outre les propriétés dont la liste apparaît dans le tableau suivant, les objets Effect peuvent également accepter des paramètres définis par l'utilisateur, et qui doivent alors être indiqués dans le fichier XML qui spécifie les propriétés `effect.effectName` et `effect.sourceFile`. Ces paramètres spécifient les éléments d'interface utilisateur à créer (par exemple des champs éditables, des cases à cocher et des zones de liste), qui dépendent du type d'effet voulu. Il est possible de spécifier des étiquettes qui apparaîtront avec le contrôle en sus des valeurs par défaut.

Propriété	Description
<code>effect.effectName</code>	Propriété en lecture seule ; chaîne apparaissant dans le menu contextuel des effets.
<code>effect.groupName</code>	Propriété en lecture seule ; chaîne représentant le nom du groupe d'effets. Ce nom est utilisé pour le menu contextuel hiérarchique des effets.
<code>effect.sourceFile</code>	Propriété en lecture seule ; chaîne spécifiant le nom du fichier source JSFL de l'effet spécifié.
<code>effect.symbolType</code>	Propriété en lecture seule ; chaîne spécifiant le type de symbole à créer durant l'application initiale de l'effet.
<code>effect.useXMLToUI</code>	Valeur booléenne permettant de bloquer le comportement par défaut de XMLUI en construisant une boîte de dialogue comportant un ou plusieurs contrôles.

## effect.effectName

### Disponibilité

Flash MX 2004.

### Utilisation

`effect.effectName`

### Description

Propriété en lecture seule ; chaîne apparaissant dans le menu contextuel des effets. Chaque effet doit posséder un nom unique.

### Exemple

L'exemple suivant enregistre dans la variable `efName` le nom de l'effet actuel :

```
var efName = fl.activeEffect.effectName;
```

## effect.groupName

### Disponibilité

Flash MX 2004.

### Utilisation

`effect.groupName`

### Description

Propriété en lecture seule ; chaîne représentant le nom du groupe d'effets. Ce nom est utilisé pour le menu contextuel hiérarchique des effets. Si cette valeur est une chaîne vide, l'effet apparaît hors groupe en haut du menu contextuel. Le nom du groupe et celui de l'effet sont indiqués dans le fichier XML de l'effet.

### Exemple

L'exemple suivant enregistre dans la variable `efGroupName` le nom du groupe de l'effet actuel :

```
var efGroupName = fl.activeEffect.groupName;
```

## effect.sourceFile

### Disponibilité

Flash MX 2004.

### Utilisation

`effect.sourceFile`

### Description

Propriété en lecture seule ; chaîne spécifiant le nom du fichier source JSFL de l'effet spécifié. Cette chaîne est utilisée pour lier un fichier de paramètres en XML à l'implémentation de l'effet en JSFL. Ce paramètre XML doit être inclus dans le fichier XML de l'effet.

### Exemple

L'exemple suivant enregistre dans la variable `efSourceFile` le nom du fichier source de l'effet en JSFL :

```
var efSourceFile = fl.activeEffect.sourceFile;
```

## effect.symbolType

### Disponibilité

Flash MX 2004.

### Utilisation

`effect.symbolType`

### Description

Propriété en lecture seule ; chaîne spécifiant le type de symbole à créer durant l'application initiale de l'effet. Les types pris en charge sont les suivants : "graphic", "movie clip" et "button". Si aucun type de symbole n'a été spécifié lors de la création de l'effet, la valeur par défaut est "graphic".

### Exemple

L'exemple suivant enregistre dans la variable `efType` le type de symbole de l'effet actuel :

```
var efType = fl.activeEffect.symbolType;
```



# effect.useXMLToUI

## Disponibilité

Flash MX 2004.

## Utilisation

`effect.useXMLToUI`

## Description

Propriété ; valeur booléenne permettant de bloquer le comportement par défaut de XMLUI en construisant une boîte de dialogue comportant un ou plusieurs contrôles. La valeur par défaut est `true`. Si cette valeur est `false`, la boîte de dialogue standard de XMLUI n'est pas affichée et vous devez prendre en charge la création et l'affichage d'une interface utilisateur.

## Exemple

L'exemple suivant spécifie que l'effet comporte sa propre interface utilisateur :

```
function configureEffect() {  
    fl.activeEffect.useXMLToUI = false;  
}
```

# Objet Element

## Disponibilité

Flash MX 2004.

## Description

Tout ce qui apparaît sur la scène est du type `Element`. Par exemple, le code suivant permet de sélectionner un élément :

```
var el = fl.getDocumentDOM().getTimeline().layers[0].frames[0].elements[0];
```

## Résumé des méthodes de l'objet Element

Les méthodes suivantes sont disponibles pour l'objet `Element` :

Méthode	Description
<code>element.getPersistentData()</code>	Lit la valeur des données spécifiées par le paramètre <i>name</i> .
<code>element.getTransformationPoint()</code>	Récupère la valeur du point de transformation de l'élément spécifié.
<code>element.hasPersistentData()</code>	Détermine si les données spécifiées ont été associées à l'élément spécifié.
<code>element.removePersistentData()</code>	Supprime les données persistantes associées à l'objet et correspondant au nom spécifié.
<code>element.setPersistentData()</code>	Enregistre des données avec un élément.
<code>element.setTransformationPoint()</code>	Définit la position du point de transformation de l'élément.

## Résumé des propriétés de l'objet Element

Les propriétés suivantes sont disponibles pour l'objet `Element` :

Propriété	Description
<code>element.depth</code>	Propriété en lecture seule ; entier ayant une valeur supérieure à 0 et représentant la profondeur de l'objet dans l'affichage.
<code>element.elementType</code>	Propriété en lecture seule ; chaîne représentant le type de l'élément spécifié.

Propriété	Description
<code>element.height</code>	Nombre en virgule flottante spécifiant la hauteur de l'élément en pixels.
<code>element.layer</code>	Propriété en lecture seule ; représente l' <a href="#">Objet Layer</a> dans lequel se trouve l'élément.
<code>element.left</code>	Propriété en lecture seule ; nombre en virgule flottante indiquant la coordonnée gauche de l'élément.
<code>element.locked</code>	Une valeur booléenne : <code>true</code> si l'élément est verrouillé, <code>false</code> sinon.
<code>element.matrix</code>	Un <a href="#">Objet Matrix</a> L'objet <code>matrix</code> possède les propriétés <code>a</code> , <code>b</code> , <code>c</code> , <code>d</code> , <code>tx</code> et <code>ty</code> . <code>a</code> , <code>b</code> , <code>c</code> , <code>d</code> sont des valeurs en virgule flottante ; <code>tx</code> et <code>ty</code> sont des coordonnées.
<code>element.name</code>	Chaîne spécifiant le nom de l'élément. Cette chaîne est généralement appelée « nom d'occurrence ».
<code>element.rotation</code>	Valeur codée en virgule flottante ou entier compris entre -180 et 180 qui indique, en degrés, la rotation de l'objet dans le sens des aiguilles d'une montre.
<code>element.scaleX</code>	Valeur codée en virgule flottante qui précise la valeur d'échelle x des rectangles et ovales de type primitif, des symboles et des objets de dessin.
<code>element.scaleY</code>	Valeur codée en virgule flottante qui précise la valeur d'échelle y des rectangles et ovales de type primitif, des symboles et des objets de dessin.
<code>element.selected</code>	Valeur booléenne indiquant si l'objet est sélectionné ou non.
<code>element.skewX</code>	Valeur codée en virgule flottante, comprise entre -180 et 180, et précisant la valeur d'inclinaison x des rectangles et ovales de type primitif, des symboles et des objets de dessin.
<code>element.skewY</code>	Valeur codée en virgule flottante, comprise entre -180 et 180, et précisant la valeur d'inclinaison y des rectangles et ovales de type primitif, des symboles et des objets de dessin.
<code>element.top</code>	Propriété en lecture seule ; coordonnée supérieure de l'élément.
<code>element.transformX</code>	Nombre à virgule flottante spécifiant la valeur x du point de transformation de l'élément sélectionné dans le système de coordonnées du parent de l'élément.

Propriété	Description
<code>element.transformY</code>	Nombre à virgule flottante spécifiant la valeur y du point de transformation de l'élément sélectionné dans le système de coordonnées du parent de l'élément.
<code>element.width</code>	Nombre en virgule flottante spécifiant la largeur de l'élément en pixels.
<code>element.x</code>	Valeur codée en virgule flottante précisant la valeur x du point d'alignement de l'élément sélectionné.
<code>element.y</code>	Valeur codée en virgule flottante précisant la valeur y du point d'alignement de l'élément sélectionné.

## element.depth

### Disponibilité

Flash MX 2004.

### Utilisation

`element.depth`

### Description

Propriété en lecture seule ; entier ayant une valeur supérieure à 0 et représentant la profondeur de l'objet dans l'affichage. L'ordre dans lequel les objets sont tracés sur la scène spécifie l'objet qui figure au-dessus des autres. L'ordre des objets peut également être modifié à l'aide de l'option de menu Modification > Réorganisation.

### Exemple

L'exemple suivant affiche la profondeur de l'élément spécifié dans le panneau de sortie.

```
// Sélectionner un objet et exécuter ce script.
fl.trace("Depth of selected object: " +
    fl.getDocumentDOM().selection[0].depth);
```

Voir l'exemple relatif à `element.elementType`.

# element.elementType

## Disponibilité

Flash MX 2004.

## Utilisation

`element.elementType`

## Description

Propriété en lecture seule ; chaîne représentant le type de l'élément spécifié. Elle peut avoir l'une des valeurs suivantes : "shape" (forme), "text" (texte), "instance" ou "shapeObj" (objet forme). Un "shapeObj" peut être créé à l'aide d'un outil extensible.

## Exemple

L'exemple suivant enregistre dans la variable `eType` le type du premier élément :

```
// Dans un nouveau fichier, placer un clip dans le calque supérieur
// de la première image, puis exécuter ce script
var eType =
    fl.getDocumentDOM().getTimeline().layers[0].frames[0].elements[0].elementType; // eType = instance
```

L'exemple suivant affiche plusieurs propriétés pour tous les éléments de l'image ou du calque actif :

```
var tl = fl.getDocumentDOM().getTimeline()
var elts = tl.layers[tl.currentLayer].frames[tl.currentFrame].elements;
for (var x = 0; x < elts.length; x++) {
    var elt = elts[x];
    fl.trace("Element "+ x + " Name = " + elt.name + " Type = " +
        elt.elementType + " location = " + elt.left + "," + elt.top + " Depth = "
        + elt.depth);
}
```

# element.getPersistentData()

## Disponibilité

Flash MX 2004.

## Utilisation

`element.getPersistentData(name)`

## Paramètres

*name* Chaîne identifiant les données à renvoyer.

## Valeur renvoyée

Les données spécifiées par le paramètre *name* ; ou 0 si ces données n'existent pas.

## Description

Méthode ; lit la valeur des données spécifiées par le paramètre *name*. Le type renvoyé dépend du type de données enregistrées (consultez la section `element.setPersistentData()`). Seuls les symboles et les bitmaps acceptent des données persistantes.

## Exemple

L'exemple suivant définit et lit les données pour l'élément spécifié, affiche ces valeurs dans le panneau de sortie, puis supprime les données :

```
// Au moins un symbole ou un bitmap est sélectionné dans
// la première image du premier calque.
var elt =
    fl.getDocumentDOM().getTimeline().layers[0].frames[0].elements[0];
elt.setPersistentData("myData","integer", 12);
if (elt.hasPersistentData("myData")){
    fl.trace("myData = "+ elt.getPersistentData("myData"));
    elt.removePersistentData( "myData" );
    fl.trace("myData = "+ elt.getPersistentData("myData"));
}
```

# element.getTransformationPoint()

## Disponibilité

Flash CS3 Professional.

## Utilisation

`element.getTransformationPoint()`

## Paramètres

Aucun.

## Valeur renvoyée

Un point (par exemple, {x:10, y:20}, sachant que x et y sont des nombres à virgule flottante) qui précise la position du point de transformation (également *point d'origine* ou *point zéro*) dans le système de coordonnées de l'élément.

## Description

Méthode ; récupère la valeur du point de transformation de l'élément spécifié.

REMARQUE

Les points de transformation sont définis par rapport à différents emplacements, en fonction du type d'élément sélectionné. Pour plus d'informations, voir `element.setTransformationPoint()`.

## Exemple

L'exemple suivant lit le point de transformation pour le troisième élément de la neuvième image du premier calque dans le document. La propriété `transPoint.x` donne la coordonnée *x* du point de transformation. La propriété `transPoint.y` donne la coordonnée *y* du point de transformation.

```
var transPoint = fl.getDocumentDOM().getTimeline().layers[0].frames[8].
    elements[2].getTransformationPoint();
```

## Voir aussi

`document.getTransformationPoint()`, `element.setTransformationPoint()`,  
`element.transformX`, `element.transformY`

# element.hasPersistentData()

## Disponibilité

Flash MX 2004.

## Utilisation

`element.hasPersistentData(name)`

## Paramètres

*name* Chaîne spécifiant le nom de l'élément de données à tester.

## Valeur renvoyée

Une valeur booléenne : `true` si les données spécifiées sont associées à l'objet, `false` sinon.

## Description

Méthode ; détermine si les données spécifiées ont été associées à l'élément spécifié. Seuls les symboles et les bitmaps acceptent des données persistantes.

## Exemple

Voir `element.getPersistentData()`.

# element.height

## Disponibilité

Flash MX 2004.

## Utilisation

`element.height`

## Description

Propriété ; nombre en virgule flottante spécifiant la hauteur de l'élément, en pixels.

### REMARQUE

N'utilisez pas cette propriété pour redimensionner un champ texte. Pour ce faire, sélectionnez le champ texte et appelez `document.setTextRectangle()`. En effet, l'utilisation de cette propriété avec un champ texte met le texte à l'échelle.

## Exemple

L'exemple suivant fixe la hauteur de l'élément spécifié à 100 pixels :

```
fl.getDocumentDOM().getTimeline().layers[0].frames[0].elements[0].height =  
100;
```

# element.layer

## Disponibilité

Flash 8.

## Utilisation

`element.layer`

## Description

Propriété en lecture seule ; représente l'[Objet Layer](#) dans lequel se trouve l'élément.

## Exemple

L'exemple suivant enregistre dans la variable `theLayer` l'objet Layer (calque) qui contient l'élément :

```
var theLayer = element.layer;
```



# element.left

## Disponibilité

Flash MX 2004.

## Utilisation

`element.left`

## Description

Propriété en lecture seule ; nombre en virgule flottante indiquant la coordonnée gauche de l'élément. Pour les éléments qui se trouvent dans une scène, la valeur de `element.left` est calculée par rapport au coin supérieur gauche de la scène. Pour les éléments qui se trouvent dans un symbole, elle est exprimée par rapport au point d'alignement (également *point d'origine* ou *point zéro*) du symbole. Utilisez `document.setSelectionBounds()` ou `document.moveSelectionBy()` pour définir cette propriété.

## Exemple

L'exemple suivant illustre le changement de valeur de cette propriété lorsqu'un élément est déplacé :

```
// Sélectionner un élément sur la scène, puis exécuter ce script.  
var sel = fl.getDocumentDOM().selection[0];  
fl.trace("Left (before) = " + sel.left);  
fl.getDocumentDOM().moveSelectionBy({x:100, y:0});  
fl.trace("Left (after) = " + sel.left);
```

Consultez l'exemple relatif à [element.elementType](#).

# element.locked

## Disponibilité

Flash MX 2004.

## Utilisation

`element.locked`

## Description

Propriété ; valeur booléenne. `true` si l'élément est verrouillé, `false` sinon. Si la valeur de `element.elementType` est "shape", cette propriété est ignorée.

## Exemple

L'exemple suivant verrouille le premier élément de la première image du calque supérieur :

```
// L'effet est similaire à l'option de menu Modification > Disposition >  
// Verrouiller :  
fl.getDocumentDOM().getTimeline().layers[0].frames[0].elements[0].locked =  
    true;
```

# element.matrix

## Disponibilité

Flash MX 2004.

## Utilisation

`element.matrix`

## Description

Propriété ; objet Matrix. Une matrice possède les propriétés `a`, `b`, `c`, `d`, `tx` et `ty`. Les propriétés `a`, `b`, `c` et `d` sont des nombres en virgule flottante. Les propriétés `tx` et `ty` sont des coordonnées. Voir [Objet Matrix](#).

## Exemple

L'exemple suivant déplace l'élément spécifié de 10 pixels sur l'axe *x*, et de 20 pixels sur l'axe *y* :

```
var mat =  
    fl.getDocumentDOM().getTimeline().layers[0].frames[0].elements[0].matrix;  
mat.tx += 10;  
mat.ty += 20;  
fl.getDocumentDOM().getTimeline().layers[0].frames[0].elements[0].matrix =  
    mat;
```

# element.name

## Disponibilité

Flash MX 2004.

## Utilisation

`element.name`

## Description

Propriété ; chaîne spécifiant le nom de l'élément. Cette chaîne est généralement appelée « nom d'occurrence ». Si la valeur de `element.elementType` correspond à "shape", cette propriété n'est pas prise en compte. Voir [element.elementType](#).

### Exemple

L'exemple suivant définit sur "clip\_mc" le nom d'occurrence du premier élément de l'image 1 au niveau du calque supérieur :

```
fl.getDocumentDOM().getTimeline().layers[0].frames[0].elements[0].name =  
    "clip_mc";
```

Consultez l'exemple relatif à [element.elementType](#).

## element.removePersistentData()

### Disponibilité

Flash MX 2004.

### Utilisation

`element.removePersistentData(name)`

### Paramètres

*name*    Chaîne spécifiant le nom des données à supprimer.

### Valeur renvoyée

Aucune.

### Description

Méthode ; supprime les données persistantes associées à l'objet et correspondant au nom spécifié. Seuls les symboles et les bitmaps acceptent des données persistantes.

### Exemple

Voir [element.getPersistentData\(\)](#).

## element.rotation

### Disponibilité

Flash CS3 Professional.

### Utilisation

`element.rotation`

### Description

Propriété ; valeur codée en virgule flottante ou entier compris entre -180 et 180 qui indique, en degrés, la rotation de l'objet dans le sens des aiguilles d'une montre.

## Exemple

L'exemple suivant définit à 45 degrés la rotation de l'élément actuellement sélectionné :

```
var element = fl.getDocumentDOM().selection[0];
fl.trace("Element rotation = " + element.rotation);
element.rotation = 45;
fl.trace("After setting rotation to 45: rotation = " + element.rotation);
```

# element.scaleX

## Disponibilité

Flash CS3 Professional.

## Utilisation

`element.scaleX`

## Description

Propriété ; valeur codée en virgule flottante qui précise la valeur d'échelle *x* des rectangles et ovales de type primitif, des symboles et des objets de dessin. La valeur 1 indique une échelle de 100 %.

## Exemple

L'exemple suivant définit la valeur d'échelle *x* de la sélection en cours sur 2 (double sa valeur) :

```
var element = fl.getDocumentDOM().selection[0];
element.scaleX = 2;
```

## Voir aussi

[element.scaleY](#)

# element.scaleY

## Disponibilité

Flash CS3 Professional.

## Utilisation

`element.scaleY`

## Description

Propriété ; valeur codée en virgule flottante qui précise la valeur d'échelle *y* des rectangles et ovales de type primitif, des symboles et des objets de dessin. La valeur 1 indique une échelle de 100 %.

## Exemple

L'exemple suivant définit la valeur d'échelle *y* de la sélection en cours sur 2 (double sa valeur) :

```
var element = fl.getDocumentDOM().selection[0];
element.scaleY = 2;
```

## Voir aussi

[element.scaleX](#)

# element.selected

## Disponibilité

Flash 8.

## Utilisation

`element.selected`

## Description

Propriété ; valeur booléenne indiquant si l'élément est sélectionné (`true`) ou non (`false`).

## Exemple

L'exemple suivant sélectionne l'élément :

```
element.selected = true;
```

# element.setPersistentData()

## Disponibilité

Flash MX 2004.

## Utilisation

`element.setPersistentData(name, type, value)`

## Paramètres

*name* Chaîne spécifiant le nom à associer aux données. Ce nom est utilisé pour lire les données.

*type* Chaîne spécifiant le type des données. Les valeurs possibles sont les suivantes : "integer", "integerArray", "double", "doubleArray", "string" et "byteArray".

*value* Spécifie la valeur à associer à l'objet. Le type de données de *value* dépend de la valeur du paramètre *type*. La valeur spécifiée doit être appropriée pour le type de données spécifié par le paramètre *type*.

## Valeur renvoyée

Aucune.

## Description

Méthode ; enregistre des données avec un élément. Ces données sont accessibles en JavaScript à la réouverture du fichier FLA contenant l'élément. Seuls les symboles et les bitmaps acceptent des données persistantes.

## Exemple

Voir `element.getPersistentData()`.

# element.setTransformationPoint()

## Disponibilité

Flash CS3 Professional.

## Utilisation

```
element.setTransformationPoint(transformationPoint)
```

## Paramètres

*transformationPoint* Point (par exemple, {x:10, y:20}, sachant que x et y sont des nombres à virgule flottante) spécifiant des valeurs pour un point de transformation d'un élément ou d'un groupe.

- Formes : *transformationPoint* est défini par rapport au document (0,0 étant le coin supérieur gauche de la scène).
- Symboles : *transformationPoint* est défini par rapport au point d'alignement du symbole (0,0 étant situé au point d'alignement).
- Texte : *transformationPoint* est défini par rapport au champ texte (0,0 étant le coin supérieur gauche du champ texte).
- Bitmaps/vidéos : *transformationPoint* est défini par rapport au bitmap/vidéo (0,0 étant le coin supérieur gauche du bitmap ou de la vidéo).
- Objets de dessin, objets de type primitif et groupes : *transformationPoint* est défini par rapport au centre de l'élément ou du groupe (0,0 étant le point central de l'élément ou du groupe).

## Valeur renvoyée

Aucune.

## Description

Méthode ; définit la position du point de transformation de l'élément.

Cette méthode est similaire à `document.setTransformationPoint()`, mais diffère cependant sur certains aspects :

- Le point de transformation des objets de dessin, des objets de type primitif et des groupes est calculé par rapport au centre de l'élément ou du groupe, et non par rapport à la scène.
- Il est possible de définir des points de transformation d'éléments sans les sélectionner préalablement.

Cette méthode déplace le point de transformation, mais ne déplace pas l'élément contrairement aux propriétés `element.transformX` et `element.transformY`.

## Exemple

L'exemple suivant règle sur 100, 200 le point de transformation du troisième élément sur la scène :

```
fl.getDocumentDOM().getTimeline().layers[0].frames[0].  
elements[2].setTransformationPoint({x:100, y:200});
```

## Voir aussi

`document.setTransformationPoint()`, `element.getTransformationPoint()`,  
`element.transformX`, `element.transformY`

# element.skewX

## Disponibilité

Flash CS3 Professional.

## Utilisation

`element.skewX`

## Description

Propriété ; valeur codée en virgule flottante, comprise entre -180 et 180, et précisant la valeur d'inclinaison *x* des rectangles et ovales de type primitif, des symboles et des objets de dessin.

## Exemple

L'exemple suivant définit la valeur d'inclinaison *x* de la sélection en cours sur 10 :

```
var element = fl.getDocumentDOM().selection[0];  
element.skewX = 10;
```

## Voir aussi

`document.setTransformationPoint()`, `element.skewY`

# element.skewY

## Disponibilité

Flash CS3 Professional.

## Utilisation

`element.skewY`

## Description

Propriété ; valeur codée en virgule flottante, comprise entre -180 et 180, et précisant la valeur d'inclinaison *y* des rectangles et ovales de type primitif, des symboles et des objets de dessin.

## Exemple

L'exemple suivant définit la valeur d'inclinaison *y* de la sélection en cours sur 10 :

```
var element = fl.getDocumentDOM().selection[0];  
element.skewY = 10;
```

## Voir aussi

[document.setTransformationPoint\(\)](#), [element.skewX](#)

# element.top

## Disponibilité

Flash MX 2004.

## Utilisation

`element.top`

## Description

Propriété en lecture seule ; coordonnée supérieure de l'élément. Pour les éléments qui se trouvent dans une scène, la valeur de `element.top` est exprimée relativement au coin supérieur gauche de la scène. Pour les éléments qui se trouvent dans un symbole, elle est exprimée relativement au point d'alignement du symbole. Utilisez [document.setSelectionBounds\(\)](#) ou [document.moveSelectionBy\(\)](#) pour définir cette propriété.



## Exemple

L'exemple suivant illustre le changement de valeur de cette propriété lorsqu'un élément est déplacé :

```
// Sélectionner un élément dans la scène, puis exécuter ce script.  
var sel = fl.getDocumentDOM().selection[0];  
fl.trace("Top (before) = " + sel.top);  
fl.getDocumentDOM().moveSelectionBy({x:0, y:100});  
fl.trace("Top (after) = " + sel.top);
```

Consultez l'exemple relatif à [element.elementType](#).

# element.transformX

## Disponibilité

Flash CS3 Professional.

## Utilisation

`element.transformX`

## Description

Propriété ; nombre à virgule flottante spécifiant la valeur  $x$  du point de transformation de l'élément sélectionné dans le système de coordonnées du parent de l'élément. La définition d'une nouvelle valeur pour cette propriété déplace l'élément. Par contre, la méthode [element.setTransformationPoint\(\)](#) déplace le point de transformation, mais pas l'élément.

## Voir aussi

[element.getTransformationPoint\(\)](#), [element.setTransformationPoint\(\)](#),  
[element.transformY](#)

# element.transformY

## Disponibilité

Flash CS3 Professional.

## Utilisation

`element.transformY`

## Description

Propriété ; nombre à virgule flottante spécifiant la valeur *y* du point de transformation de l'élément sélectionné dans le système de coordonnées du parent de l'élément. La définition d'une nouvelle valeur pour cette propriété déplace l'élément. Par contre, la méthode `element.setTransformationPoint()` déplace le point de transformation, mais pas l'élément.

## Voir aussi

`element.getTransformationPoint()`, `element.setTransformationPoint()`,  
`element.transformX`

# element.width

## Disponibilité

Flash MX 2004.

## Utilisation

`element.width`

## Description

Propriété ; nombre en virgule flottante spécifiant la largeur de l'élément, en pixels.

### REMARQUE

N'utilisez pas cette propriété pour redimensionner un champ texte. Pour ce faire, sélectionnez le champ texte et appelez `document.setTextRectangle()`. En effet, l'utilisation de cette propriété avec un champ texte met le texte à l'échelle.

## Exemple

L'exemple suivant fixe la largeur de l'élément spécifié à 100 pixels :

```
fl.getDocumentDOM().getTimeline().layers[0].frames[0].elements[0].width=100;
```

# element.x

## Disponibilité

Flash CS3 Professional.

## Utilisation

`element.x`

## Description

Propriété ; valeur codée en virgule flottante et précisant la valeur  $x$  du point d'alignement de l'élément sélectionné.

## Exemple

L'exemple suivant règle la valeur du point d'alignement de l'élément spécifié sur 100, 200 :

```
fl.getDocumentDOM().getTimeline().layers[0].frames[0].elements[0].widthx=
    100;
fl.getDocumentDOM().getTimeline().layers[0].frames[0].elements[0].y= 200;
```

## Voir aussi

[element.y](#)

# element.y

## Disponibilité

Flash CS3 Professional.

## Utilisation

`element.y`

## Description

Propriété ; valeur codée en virgule flottante précisant la valeur  $y$  du point d'alignement de l'élément sélectionné.

## Exemple

Voir [element.x](#).

# Objet Fill

## Disponibilité

Flash MX 2004.

## Description

L'objet Fill contient toutes les propriétés du paramètre de couleur de remplissage (Fill color) du panneau Outils ou de la forme sélectionnée. Pour récupérer un objet Fill, utilisez `document.getCustomFill()`.

## Résumé des propriétés de l'objet Fill

Les propriétés suivantes sont disponibles pour l'objet Fill :

Propriété	Description
<code>fill.color</code>	Chaîne, valeur hexadécimale ou entier représentant la couleur de remplissage.
<code>fill.colorArray</code>	Tableau de couleurs en dégradé.
<code>fill.focalPoint</code>	Entier spécifiant le décalage horizontal du point focal du dégradé par rapport au point de transformation.
<code>fill.linearRGB</code>	Valeur booléenne qui indique si le remplissage doit être effectué sous forme de dégradé RVB linéaire ou radial.
<code>fill.matrix</code>	Un <a href="#">Objet Matrix</a> qui définit l'emplacement, l'orientation et les échelles des remplissages avec dégradé.
<code>fill.overflow</code>	Chaîne qui indique le comportement d'un débordement de dégradé.
<code>fill.posArray</code>	Tableau d'entiers compris entre 0 et 255, indiquant la position de la couleur correspondante.
<code>fill.style</code>	Chaîne qui spécifie le style de remplissage.

## fill.color

## Disponibilité

Flash MX 2004.

## Utilisation

`fill.color`

## Description

Propriété ; couleur du remplissage, exprimée dans l'un des formats suivants :

- Chaîne au format "#RRGGBB" ou "#RRGGBBAA"
- Nombre hexadécimal au format 0xRRGGBB
- Entier représentant l'équivalent décimal d'un nombre hexadécimal

## Exemple

L'exemple suivant définit la couleur de remplissage de la sélection :

```
var fill = fl.getDocumentDOM().getCustomFill();
fill.color = '#FFFFFF';
fl.getDocumentDOM().setCustomFill( fill );
```

# fill.colorArray

## Disponibilité

Flash MX 2004.

## Utilisation

fill.colorArray

## Description

Propriété ; tableau de couleurs du dégradé, exprimées sous forme d'entiers. Cette propriété n'est disponible que si la valeur de la propriété fill.style correspond à "radialGradient" ou "linearGradient". Consultez la section [fill.style](#).

## Exemple

L'exemple suivant affiche le tableau de couleurs de la sélection dans le panneau de sortie :

```
var fill = fl.getDocumentDOM().getCustomFill();
if(fill.style == "linearGradient" || fill.style == "radialGradient")
    alert(fill.colorArray);
```

L'exemple suivant règle le remplissage sur le dégradé linéaire :

```
var fill = fl.getDocumentDOM().getCustomFill();
fill.style = "linearGradient";
fill.colorArray = ["#00ff00", "#ff00ff"];
fill.posArray = [0, 255];
fl.getDocumentDOM().setCustomFill(fill);
```

# fill.focalPoint

## Disponibilité

Flash 8.

## Utilisation

`fill.focalPoint`

## Description

Propriété ; entier spécifiant le décalage horizontal du point focal du dégradé par rapport au point de transformation. Par exemple, une valeur de 10 place le point focal à 10/255 de la distance entre le point de transformation et le bord du dégradé. Une valeur de -255 place le point focal sur le bord gauche du dégradé. La valeur par défaut est 0.

Cette propriété n'est disponible que si la valeur de la propriété `fill.style` est "radialGradient".

## Exemple

L'exemple suivant fixe, pour la sélection en cours, le point focal d'un dégradé radial à 100 pixels à droite du centre de la forme.

```
var fill = fl.getDocumentDOM().getCustomFill();
fill.style = "radialGradient";
fill.colorArray = ["#00ff00", "#ff00ff"];
fill.posArray = [0, 255];
fill.focalPoint = 10100;
fl.getDocumentDOM().setCustomFill(fill);
```

# fill.linearRGB

## Disponibilité

Flash 8.

## Utilisation

`fill.linearRGB`

## Description

Propriété ; valeur booléenne qui indique si le remplissage doit être effectué sous forme de dégradé RVB linéaire ou radial. Cette propriété doit être réglée sur `true` pour obtenir une interpolation linéaire du dégradé, sur `false` pour obtenir une interpolation radiale. La valeur par défaut est `false`.

## Exemple

L'exemple suivant spécifie un rendu par interpolation RVB linéaire du dégradé de la sélection en cours :

```
var fill = fl.getDocumentDOM().getCustomFill();
fill.linearRGB style = true"radialGradient";
fill.colorArray = ["#00ff00", "#ff00ff"];
fill.posArray = [0, 255];
fill.focalPoint = 100;
fill.linearRGB = true;
fl.getDocumentDOM().setCustomFill(fill);
```

## fill.matrix

### Disponibilité

Flash MX 2004.

### Utilisation

`fill.matrix`

### Description

Propriété ; un [Objet Matrix](#) qui définit l'emplacement, l'orientation et les échelles des remplissages avec dégradé.

## Exemple

L'exemple suivant utilise la propriété `fill.matrix` pour spécifier un remplissage dégradé sur la sélection en cours :

```
var fill = fl.getDocumentDOM().getCustomFill();
fill.style = 'radialGradient';
fill.colorArray = ['#00ff00', '#ff00ff'];
fill.posArray = [0, 255];
fill.focalPoint = 100;
fill.linearRGB = false;
fill.overflow = 'repeat';
var mat = fl.getDocumentDOM().selection[0].matrix;
mat.a = 0.0167083740234375;
mat.b = -0.0096435546875;
mat.c = 0.0312957763671875;
mat.d = 0.05419921875;
mat.tx = 288.65;
mat.ty = 193.05;
for (i in mat) {
    fl.trace(i+' : '+mat[i]);
}
fl.getDocumentDOM().setCustomFill(fill);
```

# fill.overflow

## Disponibilité

Flash 8.

## Utilisation

`fill.overflow`

## Description

Propriété ; chaîne qui indique le comportement d'un débordement de dégradé. Les valeurs gérées sont "extend" (extension), "repeat" (répétition) et "reflect" (réflexion) ; ces trois chaînes ne sont pas sensibles à la casse. La valeur par défaut est "extend".

## Exemple

L'exemple suivant spécifie un débordement de type "extend" pour la sélection en cours :

```
var fill = fl.getDocumentDOM().getCustomFill();
fill.overflow = "extend";
fl.getDocumentDOM().setCustomFill(fill);
```

# fill.posArray

## Disponibilité

Flash MX 2004.

## Utilisation

`fill.posArray`

## Description

Propriété ; tableau d'entiers compris entre 0 et 255, indiquant la position de la couleur correspondante. Cette propriété n'est disponible que si la valeur de la propriété `fill.style` est "radialGradient" ou "linearGradient".

## Exemple

L'exemple suivant spécifie les couleurs à utiliser pour un dégradé linéaire dans la sélection actuelle :

```
var fill = fl.getDocumentDOM().getCustomFill();
fill.style = "linearGradient";
fill.colorArray = [ 0x00ff00, 0xff0000, 0x0000ff ];
fill.posArray = [0, 100, 200];
fl.getDocumentDOM().setCustomFill( fill );
```



# fill.style

## Disponibilité

Flash MX 2004.

## Utilisation

`fill.style`

## Description

Propriété ; chaîne qui spécifie le style de remplissage. Les valeurs gérées sont "solid" (uni), "linearGradient" (dégradé linéaire), "radialGradient" (dégradé radial) et "noFill" (pas de remplissage). Pour un objet qui ne comporte pas de remplissage, cette propriété a la valeur "noFill".

Si cette propriété a la valeur "linearGradient" ou "radialGradient", les propriétés `fill.colorArray` et `fill.posArray` sont également disponibles.

## Exemple

L'exemple suivant spécifie les couleurs à utiliser pour un dégradé linéaire dans la sélection actuelle :

```
var fill = fl.getDocumentDOM().getCustomFill();
fill.style= "linearGradient";
fill.colorArray = [ 0x00ff00, 0xff0000, 0x0000ff ];
fill.posArray = [0, 100, 200];
fl.getDocumentDOM().setCustomFill( fill );
```

# Objet Filter

## Disponibilité

Flash 8.

## Description

L'objet `Filter` contient toutes les propriétés de tous les filtres. La propriété `filter.name` spécifie le type de filtre et détermine les propriétés qui s'appliquent à chaque filtre.

Voir `filter.name`.

Pour renvoyer la liste de filtres associée à un ou plusieurs objets, utilisez `document.getFilters()`. Pour appliquer des filtres à un ou plusieurs objets, utilisez `document.setFilters()`. Consultez les sections `document.getFilters()` et `document.setFilters()`.

## Résumé des propriétés de l'objet Filter

L'objet `Filter` gère les propriétés suivantes :

Propriété	Description
<code>filter.angle</code>	Nombre en virgule flottante spécifiant l'angle de l'ombre ou de la couleur de surbrillance, exprimé en degrés.
<code>filter.blurX</code>	Valeur codée en virgule flottante et spécifiant la quantité de flou dans la direction x, exprimée en pixels.
<code>filter.blurY</code>	Valeur codée en virgule flottante et spécifiant la quantité de flou dans la direction y, exprimée en pixels.
<code>filter.brightness</code>	Nombre en virgule flottante spécifiant la luminosité du filtre.
<code>filter.color</code>	Chaîne, valeur hexadécimale ou entier représentant la couleur du filtre.
<code>filter.contrast</code>	Nombre en virgule flottante spécifiant la valeur de contraste du filtre.
<code>filter.distance</code>	Nombre en virgule flottante spécifiant la distance, en pixels, entre l'effet du filtre et un objet.
<code>filter.enabled</code>	Valeur booléenne indiquant si le filtre spécifié est activé ou non.
<code>filter.hideObject</code>	Valeur booléenne indiquant si l'image source est cachée.
<code>filter.highlightColor</code>	Chaîne, valeur hexadécimale ou entier représentant la couleur de soulignement.

Propriété	Description
<code>filter.hue</code>	Nombre en virgule flottante spécifiant la teinte du filtre.
<code>filter.inner</code>	Valeur booléenne indiquant si l'ombre est une ombre interne.
<code>filter.knockout</code>	Valeur booléenne indiquant si le filtre est un filtre masque.
<code>filter.name</code>	Propriété en lecture seule ; chaîne indiquant le type de filtre.
<code>filter.quality</code>	Chaîne spécifiant la qualité du flou.
<code>filter.saturation</code>	Nombre en virgule flottante spécifiant la valeur de saturation du filtre.
<code>filter.shadowColor</code>	Chaîne, valeur hexadécimale ou entier représentant la couleur de l'ombre.
<code>filter.strength</code>	Entier spécifiant l'intensité du filtre en pourcentage.
<code>filter.type</code>	Chaîne spécifiant le type de biseau ou de néon.

## filter.angle

### Disponibilité

Flash 8.

### Utilisation

`filter.angle`

### Description

Propriété ; nombre en virgule flottante spécifiant l'angle de l'ombre ou de la couleur de surbrillance, exprimé en degrés. Les valeurs gérées sont comprises entre 0 et 360. Cette propriété est définie pour les objets Filter dont la propriété `filter.name` correspond à "bevelFilter", "dropShadowFilter", "gradientBevelFilter" ou "gradientGlowFilter".

### Exemple

L'exemple suivant définit un angle de 120 pour les filtres Biseau sur les objets sélectionnés :

```
var myFilters = fl.getDocumentDOM().getFilters();
for(i=0; i < myFilters.length; i++) {
    if(myFilters[i].name == 'bevelFilter'){
        myFilters[i].angle = 120;
    }
}
fl.getDocumentDOM().setFilters(myFilters);
```

## Voir aussi

[document.setFilterProperty\(\)](#)

# filter.blurX

## Disponibilité

Flash 8.

## Utilisation

`filter.blurX`

## Description

Propriété ; valeur codée en virgule flottante et spécifiant la quantité de flou dans la direction *x*, exprimée en pixels. Les valeurs gérées sont comprises entre 0 et 255. Cette propriété est définie pour les objets `Filter` dont la propriété [filter.name](#) correspond à "bevelFilter", "blurFilter", "dropShadowFilter", "glowFilter", "gradientBevelFilter" ou "gradientGlowFilter".

## Exemple

L'exemple suivant règle la valeur `blurX` sur 30 et la valeur `blurY` sur 20 pour les filtres Flou sur les objets sélectionnés :

```
var myFilters = fl.getDocumentDOM().getFilters();
for(i=0; i < myFilters.length; i++){
    if(myFilters[i].name == 'blurFilter'){
        myFilters[i].blurX = 30;
        myFilters[i].blurY = 20;
    }
}
fl.getDocumentDOM().setFilters(myFilters);
```

## Voir aussi

[document.setFilterProperty\(\)](#), [filter.blurY](#)

# filter.blurY

## Disponibilité

Flash 8.

## Utilisation

`filter.blurY`

## Description

Propriété ; valeur codée en virgule flottante et spécifiant la quantité de flou dans la direction *y*, exprimée en pixels. Les valeurs gérées sont comprises entre 0 et 255. Cette propriété est définie pour les objets Filter dont la propriété `filter.name` correspond à "bevelFilter", "blurFilter", "dropShadowFilter", "glowFilter", "gradientBevelFilter" ou "gradientGlowFilter".

## Exemple

Voir `filter.blurX`.

## Voir aussi

`document.setFilterProperty()`, `filter.blurX`

# filter.brightness

## Disponibilité

Flash 8.

## Utilisation

`filter.brightness`

## Description

Propriété ; nombre en virgule flottante spécifiant la luminosité du filtre. Les valeurs gérées sont comprises entre -100 et 100. Cette propriété est réservée aux objets Filter dont la propriété `filter.name` correspond à "adjustColorFilter".

## Exemple

L'exemple suivant définit une luminosité de 30,5 pour les filtres de réglage de la couleur sur les objets sélectionnés :

```
var myFilters = fl.getDocumentDOM().getFilters();
for(i=0; i < myFilters.length; i++){
    if(myFilters[i].name == 'adjustColorFilter'){
        myFilters[i].brightness = 30.5;
    }
}
fl.getDocumentDOM().setFilters(myFilters);
```

# filter.color

## Disponibilité

Flash 8.

## Utilisation

`filter.color`

## Description

Propriété ; couleur du filtre, exprimée dans l'un des formats suivants :

- Chaîne au format "#RRVBBB" ou "#RRGGBBAA"
- Nombre hexadécimal au format 0xRRVBBB
- Entier représentant l'équivalent décimal d'un nombre hexadécimal

Cette propriété est réservée aux objets Filter dont la propriété `filter.name` correspond à "dropShadowFilter" ou "glowFilter".

## Exemple

L'exemple suivant règle la couleur sur "#ff00003e" pour les filtres d'ombre portée sur les objets sélectionnés :

```
var myFilters = fl.getDocumentDOM().getFilters();
for(i=0; i < myFilters.length; i++){
    if(myFilters[i].name == 'dropShadowFilter'){
        myFilters[i].color = '#ff00003e';
    }
}
fl.getDocumentDOM().setFilters(myFilters);
```

## Voir aussi

`document.setFilterProperty()`

# filter.contrast

## Disponibilité

Flash 8.

## Utilisation

`filter.contrast`

## Description

Propriété ; nombre en virgule flottante spécifiant la valeur de contraste du filtre. Les valeurs gérées sont comprises entre -100 et 100. Cette propriété est réservée aux objets Filter dont la propriété `filter.name` correspond à "adjustColorFilter".

## Exemple

L'exemple suivant définit un contraste de 15,5 pour les filtres de réglage de la couleur sur les objets sélectionnés :

```
var myFilters = fl.getDocumentDOM().getFilters();
for(i=0; i < myFilters.length; i++){
    if(myFilters[i].name == 'adjustColorFilter'){
        myFilters[i].contrast = -15.5;
    }
}
fl.getDocumentDOM().setFilters(myFilters);
```

# filter.distance

## Disponibilité

Flash 8.

## Utilisation

`filter.distance`

## Description

Propriété ; nombre en virgule flottante spécifiant la distance, en pixels, entre l'effet du filtre et un objet. Les valeurs gérées sont comprises entre 255 et 255. Cette propriété est définie pour les objets Filter dont la propriété `filter.name` correspond à "bevelFilter", "dropShadowFilter", "gradientBevelFilter" ou "gradientGlowFilter".

## Exemple

L'exemple suivant règle la distance sur 10 pixels pour les filtres d'ombre portée sur les objets sélectionnés :

```
var myFilters = fl.getDocumentDOM().getFilters();
for(i=0; i < myFilters.length; i++){
    if(myFilters[i].name == 'dropShadowFilter'){
        myFilters[i].distance = 10;
    }
}
fl.getDocumentDOM().setFilters(myFilters);
```

## Voir aussi

[document.setFilterProperty\(\)](#)

# filter.enabled

## Disponibilité

Flash CS3 Professional.

## Utilisation

`filter.enabled`

## Description

Propriété ; valeur booléenne indiquant si le filtre spécifié est activé (`true`) ou non (`false`).

## Exemple

L'exemple suivant désactive les filtres de couleurs appliqués aux objets sélectionnés :

```
var myFilters = fl.getDocumentDOM().getFilters();
for(i=0; i < myFilters.length; i++){
    if(myFilters[i].name == 'adjustColorFilter'){
        myFilters[i].enabled = false;
    }
}
fl.getDocumentDOM().setFilters(myFilters);
```

# filter.hideObject

## Disponibilité

Flash 8.

## Utilisation

`filter.hideObject`

## Description

Propriété ; valeur booléenne indiquant si l'image source est masquée (`true`) ou affichée (`false`). Cette propriété est réservée aux objets Filter dont la propriété `filter.name` correspond à "dropShadowFilter".

## Exemple

L'exemple suivant règle la valeur `hideObject` sur `true` pour les filtres d'ombre portée sur les objets sélectionnés :

```
var myFilters = fl.getDocumentDOM().getFilters();
for(i=0; i < myFilters.length; i++){
    if(myFilters[i].name == 'dropShadowFilter'){
        myFilters[i].hideObject = true;
    }
}
fl.getDocumentDOM().setFilters(myFilters);
```



# filter.highlightColor

## Disponibilité

Flash 8.

## Utilisation

`filter.highlightColor`

## Description

Propriété ; couleur de soulignement, exprimée dans l'un des formats suivants :

- Chaîne au format "#RRVVBB" ou "#RRGGBBAA"
- Nombre hexadécimal au format 0xRRVVBB
- Entier représentant l'équivalent décimal d'un nombre hexadécimal

Cette propriété est réservée aux objets Filter dont la propriété `filter.name` correspond à "bevelFilter".

## Exemple

L'exemple suivant règle la couleur de soulignement sur "#ff00003e" pour les filtres de biseau sur les objets sélectionnés :

```
var myFilters = fl.getDocumentDOM().getFilters();
for(i=0; i < myFilters.length; i++){
    if(myFilters[i].name == 'bevelFilter'){
        myFilters[i].highlightColor = '#ff00003e';
    }
}
fl.getDocumentDOM().setFilters(myFilters);
```

# filter.hue

## Disponibilité

Flash 8.

## Utilisation

`filter.hue`

## Description

Propriété ; nombre en virgule flottante spécifiant la teinte du filtre. Les valeurs gérées sont comprises entre -180 et 180. Cette propriété est réservée aux objets Filter dont la propriété `filter.name` correspond à "adjustColorFilter".

## Exemple

L'exemple suivant définit une teinte de 120 pour les filtres de réglage de la couleur sur les objets sélectionnés :

```
var myFilters = fl.getDocumentDOM().getFilters();
for(i=0; i < myFilters.length; i++){
    if(myFilters[i].name == 'adjustColorFilter'){
        myFilters[i].hue = 120;
    }
}
fl.getDocumentDOM().setFilters(myFilters);
```

## filter.inner

### Disponibilité

Flash 8.

### Utilisation

filter.inner

### Description

Propriété ; valeur booléenne indiquant si l'ombre est intérieure (true) ou non (false).

Cette propriété est réservée aux objets Filter dont la propriété [filter.name](#) correspond à "dropShadowFilter" ou "glowFilter".

## Exemple

L'exemple suivant règle la valeur de la propriété inner sur true pour les filtres de néon sur les objets sélectionnés :

```
var myFilters = fl.getDocumentDOM().getFilters();
for(i=0; i < myFilters.length; i++){
    if(myFilters[i].name == 'glowFilter'){
        myFilters[i].inner = true;
    }
}
fl.getDocumentDOM().setFilters(myFilters);
```

### Voir aussi

[document.setFilterProperty\(\)](#)

# filter.knockout

## Disponibilité

Flash 8.

## Utilisation

filter.knockout

## Description

Propriété ; valeur booléenne indiquant si le filtre masque (`true`) ou non (`false`) l'image source. Cette propriété est définie pour les objets Filter dont la propriété `filter.name` correspond à "bevelFilter", "dropShadowFilter", "glowFilter", "gradientBevelFilter" ou "gradientGlowFilter".

## Exemple

L'exemple suivant règle la propriété knockout sur `true` pour les filtres de néon sur les objets sélectionnés :

```
var myFilters = fl.getDocumentDOM().getFilters();
for(i=0; i < myFilters.length; i++){
    if(myFilters[i].name == 'glowFilter'){
        myFilters[i].knockout = true;
    }
}
fl.getDocumentDOM().setFilters(myFilters);
```

## Voir aussi

[document.setFilterProperty\(\)](#)

# filter.name

## Disponibilité

Flash 8.

## Utilisation

filter.name

## Description

Propriété en lecture seule ; chaîne indiquant le type de filtre. La valeur de cette propriété identifie les autres propriétés disponibles de l'objet Filter. Elle peut avoir l'une des valeurs suivantes : "adjustColorFilter", "bevelFilter", "blurFilter", "dropShadowFilter", "glowFilter", "gradientBevelFilter" ou "gradientGlowFilter".

## Exemple

L'exemple suivant affiche le nom des filtres et les positions d'index dans le panneau Sortie :

```
var myFilters = fl.getDocumentDOM().getFilters();
var traceStr = "";
for(i=0; i < myFilters.length; i++){
    traceStr = traceStr + " At index " + i + ": " + myFilters[i].name;
}
fl.trace(traceStr);
```

## Voir aussi

[document.getFilters\(\)](#), [document.setFilterProperty\(\)](#)

# filter.quality

## Disponibilité

Flash 8.

## Utilisation

`filter.quality`

## Description

Propriété ; chaîne spécifiant la qualité du flou. Les valeurs gérées sont "low", "medium" et "high" ("high" est similaire à un flou gaussien). Cette propriété est définie pour les objets `Filter` dont la propriété `filter.name` correspond à "bevelFilter", "blurFilter", "dropShadowFilter", "glowFilter", "gradientGlowFilter" ou "gradientBevelFilter".

## Exemple

L'exemple suivant règle la qualité de flou sur "medium" pour les filtres de flou sur les objets sélectionnés :

```
var myFilters = fl.getDocumentDOM().getFilters();
for(i=0; i < myFilters.length; i++){
    if(myFilters[i].name == 'glowFilter'){
        myFilters[i].quality = 'medium';
    }
}
fl.getDocumentDOM().setFilters(myFilters);
```

## Voir aussi

[document.setFilterProperty\(\)](#)

# filter.saturation

## Disponibilité

Flash 8.

## Utilisation

`filter.saturation`

## Description

Propriété ; nombre en virgule flottante spécifiant la valeur de saturation du filtre. Les valeurs gérées sont comprises entre -100 et 100. Cette propriété est réservée aux objets Filter dont la propriété `filter.name` correspond à "adjustColorFilter".

## Exemple

L'exemple suivant règle la saturation sur -100 (niveaux de gris) pour les filtres de réglage de la couleur sur les objets sélectionnés :

```
var myFilters = fl.getDocumentDOM().getFilters();
for(i=0; i < myFilters.length; i++){
    if(myFilters[i].name == 'adjustColorFilter'){
        myFilters[i].saturation = 0-100;
    }
}
fl.getDocumentDOM().setFilters(myFilters);
```

## Voir aussi

`document.setFilterProperty()`

# filter.shadowColor

## Disponibilité

Flash 8.

## Utilisation

`filter.shadowColor`

## Description

Propriété ; couleur de l'ombre, exprimée dans l'un des formats suivants :

- Chaîne au format "#RRVVBB" ou "#RRGGBBAA"
- Nombre hexadécimal au format 0xRRVVBB
- Entier représentant l'équivalent décimal d'un nombre hexadécimal

Cette propriété est réservée aux objets Filter dont la propriété `filter.name` correspond à "bevelFilter".

## Exemple

L'exemple suivant règle la couleur de l'ombre sur "#ff00003e" pour les filtres de biseau sur les objets sélectionnés :

```
var myFilters = fl.getDocumentDOM().getFilters();
for(i=0; i < myFilters.length; i++){
    if(myFilters[i].name == 'bevelFilter'){
        myFilters[i].shadowColor = '#ff00003e';
    }
}
fl.getDocumentDOM().setFilters(myFilters);
```

## Voir aussi

[document.setFilterProperty\(\)](#)

# filter.strength

## Disponibilité

Flash 8.

## Utilisation

filter.strength

## Description

Propriété ; entier spécifiant l'intensité du filtre en pourcentage. Les valeurs gérées sont comprises entre 0 et 25 500. Cette propriété est définie pour les objets Filter dont la propriété [filter.name](#) correspond à "bevelFilter", "dropShadowFilter", "glowFilter", "gradientGlowFilter" ou "gradientBevelFilter".

## Exemple

L'exemple suivant règle l'intensité sur 50 pour les filtres de néon sur les objets sélectionnés :

```
var myFilters = fl.getDocumentDOM().getFilters();
for(i=0; i < myFilters.length; i++){
    if(myFilters[i].name == 'glowFilter'){
        myFilters[i].strength = 50;
    }
}
fl.getDocumentDOM().setFilters(myFilters);
```

## Voir aussi

[document.setFilterProperty\(\)](#)

# filter.type

## Disponibilité

Flash 8.

## Utilisation

`filter.type`

## Description

Propriété ; chaîne spécifiant le type de filtre de biseau ou de néon. Les valeurs gérées sont "inner", "outer" et "full". Cette propriété est réservée aux objets Filter dont la propriété `filter.name` correspond à "bevelFilter", "gradientGlowFilter" ou "gradientBevelFilter".

## Exemple

L'exemple suivant règle sur "full" le type de tous les filtres de biseau sur les objets sélectionnés :

```
var myFilters = fl.getDocumentDOM().getFilters();
for(i=0; i < myFilters.length; i++){
    if(myFilters[i].name == 'bevelFilter'){
        myFilters[i].type = 'full';
    }
}
fl.getDocumentDOM().setFilters(myFilters);
```

## Voir aussi

`document.setFilterProperty()`

# Objet flash (fl)

## Disponibilité

Flash MX 2004.

## Description

L'objet flash représente l'application Flash. Il porte le nom `flash` ou `fl`. Cette documentation utilise systématiquement le terme `fl` dans les exemples de code.

## Résumé des méthodes de l'objet flash

Vous pouvez utiliser les méthodes suivantes avec l'objet flash :

Méthode	Description
<code>fl.addEventListener()</code>	Enregistre une fonction à appeler à la réception d'un événement particulier.
<code>fl.browseForFileURL()</code>	Ouvre une boîte de dialogue Fichier > Ouvrir ou Fichier > Enregistrer pour permettre à l'utilisateur d'indiquer un nom de fichier à ouvrir ou à enregistrer.
<code>fl.browseForFolderURL()</code>	Affiche la boîte de dialogue Rechercher un dossier et permet à l'utilisateur de sélectionner un dossier.
<code>fl.clipCopyString()</code>	Copie la chaîne spécifiée dans le Presse-papiers.
<code>fl.closeAll()</code>	Ferme tous les documents ouverts et affiche la boîte de dialogue Enregistrer sous pour les éventuels documents qui n'ont pas été enregistrés.
<code>fl.closeAllPlayerDocuments()</code>	Ferme tous les fichiers SWF qui ont été ouverts par le biais de Contrôle > Tester l'animation.
<code>fl.closeDocument()</code>	Ferme le document spécifié.
<code>fl.closeProject()</code>	Ferme le fichier du projet Flash (FLP, Flash Project) actuellement ouvert.
<code>fl.createDocument()</code>	Ouvre un nouveau document et le sélectionne.
<code>fl.createProject()</code>	Crée un fichier de projet Flash (FLP, Flash Project) avec le nom indiqué.
<code>fl.downloadLatestVersion()</code>	A partir du serveur Version Cue, télécharge la version la plus récente d'un fichier qui n'est pas actuellement ouvert.



Méthode	Description
<code>fl.enableImmediateUpdates()</code>	Permet au développeur du script d'activer l'actualisation immédiate du scénario lors de l'exécution d'effets.
<code>fl.fileExists()</code>	Vérifie si un fichier existe déjà sur le disque dur.
<code>fl.findDocumentDOM()</code>	Permet de cibler un fichier spécifique au moyen de son identificateur unique.
<code>fl.findDocumentIndex()</code>	Renvoie un tableau d'entiers représentant la position d'un document dans le tableau <code>fl.documents</code> .
<code>fl.findObjectInDocByName()</code>	Expose des éléments avec des noms d'occurrences contenant un texte particulier dans un document.
<code>fl.findObjectInDocByType()</code>	Expose des éléments d'un type particulier dans un document.
<code>fl.getAppMemoryInfo()</code>	Renvoie un entier représentant le nombre d'octets utilisés dans la zone mémoire de Flash.exe spécifiée.
<code>fl.getDocumentDOM()</code>	Lit le DOM ( <a href="#">Objet Document</a> ) du document actuellement actif.
<code>fl.getProject()</code>	Renvoie un objet Project représentant le projet actuellement ouvert.
<code>fl.mapPlayerURL()</code>	Convertit une adresse URL avec des séquences d'échappement en Unicode en une URL en UTF-8 ou MBCS.
<code>fl.openDocument()</code>	Ouvre dans une nouvelle fenêtre de document un document Flash (FLA) à modifier, et active cette fenêtre.
<code>fl.openProject()</code>	Ouvre un fichier de projet Flash (FLP, Flash Project) à modifier dans l'outil de programmation Flash.
<code>fl.openScript()</code>	Ouvre un script (JSFL, AS, ASC) ou tout autre fichier (XML, TXT) dans l'éditeur de texte de Flash.
<code>fl.quit()</code>	Ferme Flash en proposant à l'utilisateur d'enregistrer les documents modifiés.
<code>fl.reloadEffects()</code>	Recharge tous les descripteurs d'effets définis dans le dossier Configuration Effects de l'utilisateur.
<code>fl.reloadTools()</code>	Reconstruit le panneau Outils à partir du contenu du fichier toolconfig.xml. Cette méthode n'est utilisée que lors de la création d'outils extensibles.

Méthode	Description
<code>fl.removeEventListener()</code>	Annule l'inscription d'une fonction enregistrée à l'aide de <code>fl.addEventListener()</code> .
<code>fl.resetAS3PackagePaths()</code>	Réinitialise le paramètre de chemin de classe global sur la valeur par défaut dans la boîte de dialogue des paramètres d'ActionScript 3.0.
<code>fl.resetPackagePaths()</code>	Réinitialise le paramètre de chemin de classe global sur la valeur par défaut dans la boîte de dialogue des paramètres d'ActionScript 2.0.
<code>fl.revertDocumentToLastVersion()</code>	Remplace la version du document spécifié par celle du serveur Version Cue.
<code>fl.runScript()</code>	Exécute un fichier JavaScript.
<code>fl.saveAll()</code>	Enregistre tous les documents ouverts et affiche la boîte de dialogue Enregistrer sous pour les éventuels documents qui n'ont jamais été enregistrés.
<code>fl.saveAVersionOfDocument()</code>	Enregistre une version du document spécifié sur le serveur Version Cue.
<code>fl.saveDocument()</code>	Enregistre le document spécifié sous forme de fichier FLA.
<code>fl.saveDocumentAs()</code>	Affiche la boîte de dialogue Enregistrer sous pour le document spécifié.
<code>fl.selectElement()</code>	Active l'option de sélection ou d'édition d'un élément.
<code>fl.selectTool()</code>	Sélectionne l'outil spécifié dans le panneau Outils.
<code>fl.setActiveWindow()</code>	Définit le document spécifié comme fenêtre active.
<code>fl.showIdleMessage()</code>	Vous permet de désactiver l'avertissement relatif à un script trop long.
<code>fl.synchronizeDocumentWithHeadVersion()</code>	Synchronise le document spécifié avec la version la plus récente du serveur Version Cue.
<code>fl.trace()</code>	Envoie une chaîne de texte dans le panneau Sortie.

# Résumé des propriétés de l'objet flash

Vous pouvez utiliser les propriétés suivantes avec l'objet flash :

Propriété	Description
<code>fl.actionsPanel</code>	Propriété en lecture seule ; <a href="#">Objet actionsPanel</a> .
<code>fl.activeEffect</code>	Propriété en lecture seule ; l' <a href="#">Objet Effect</a> de l'effet actuellement appliqué.
<code>fl.as3PackagePaths</code>	Chaîne correspondant au paramètre de chemin de classe global dans la boîte de dialogue des paramètres d'ActionScript 3.0.
<code>fl.compilerErrors</code>	Propriété en lecture seule ; <a href="#">Objet compilerErrors</a> .
<code>fl.componentsPanel</code>	Propriété en lecture seule ; <a href="#">Objet componentsPanel</a> représentant le panneau Composants.
<code>fl.configDirectory</code>	Propriété en lecture seule ; chaîne indiquant le chemin d'accès complet du dossier Configuration de l'utilisateur local. Ce chemin est spécifique à chaque plate-forme.
<code>fl.configURI</code>	Propriété en lecture seule ; chaîne indiquant le chemin d'accès complet du dossier Configuration de l'utilisateur local sous la forme « fichier:/// URI ».
<code>fl.contactSensitiveSelection</code>	Valeur booléenne spécifiant si le mode de sélection Activer au contact est activé.
<code>fl.createNewDocList</code>	Propriété en lecture seule ; tableau de chaînes représentant les divers types de documents qu'il est possible de créer.
<code>fl.createNewDocListType</code>	Propriété en lecture seule ; tableau de chaînes représentant les extensions de fichiers des divers types de documents qu'il est possible de créer.
<code>fl.createNewTemplateList</code>	Propriété en lecture seule ; tableau de chaînes représentant les divers types de modèles qu'il est possible de créer.
<code>fl.documents</code>	Propriété en lecture seule ; tableau des objets Document (voir <a href="#">Objet Document</a> ) représentant les documents (fichiers FLA) actuellement ouverts pour modification.
<code>fl.drawingLayer</code>	Propriété en lecture seule ; <a href="#">Objet drawingLayer</a> que doit utiliser un outil extensible lorsque l'utilisateur dessine un tracé temporaire à l'aide de la souris.
<code>fl.effects</code>	Propriété en lecture seule ; tableau des objets Effect (voir <a href="#">Objet Effect</a> ), basé sur le fichier de paramètres XML.

Propriété	Description
<code>fl.Math</code>	Propriété en lecture seule ; l' <a href="#">Objet Math</a> fournissant les méthodes utilisées pour les opérations de matrice et de point.
<code>fl.mruRecentFileList</code>	Propriété en lecture seule ; tableau des noms de fichier complets de la liste des fichiers récemment utilisés (MRU, Most Recently Used) gérée par l'outil de programmation Flash.
<code>fl.mruRecentFileListType</code>	Propriété en lecture seule ; tableau des types de fichiers de la liste des fichiers récemment utilisés (MRU, Most Recently Used) gérée par l'outil de programmation Flash.
<code>fl.packagePaths</code>	Chaîne correspondant au paramètre de chemin de classe global dans la boîte de dialogue des paramètres d'ActionScript 2.0.
<code>fl.objectDrawingMode</code>	Entier représentant le mode de dessin d'objet qui est activé.
<code>fl.outputPanel</code>	Propriété en lecture seule ; référence à l' <a href="#">Objet outputPanel</a> .
<code>fl.scriptURI</code>	Propriété en lecture seule ; chaîne représentant le chemin d'accès du script JSFL actuellement en cours d'exécution et exprimée sous la forme « fichier:/// URI ».
<code>fl.tools</code>	Propriété en lecture seule ; tableau des objets Tools.
<code>fl.version</code>	Propriété en lecture seule ; chaîne longue de la version de l'outil de programmation Flash, plate-forme comprise.
<code>fl.xmlui</code>	Propriété en lecture seule ; <a href="#">Objet XMLUI</a> .

## fl.actionsPanel

### Disponibilité

Flash CS3 Professional.

### Utilisation

`fl.actionsPanel`

### Description

Propriété en lecture seule ; objet `actionsPanel` représentant le panneau Action actuellement affiché. Pour plus d'informations sur l'utilisation de cette propriété, consultez la section [Objet actionsPanel](#).

# fl.activeEffect

## Disponibilité

Flash MX 2004.

## Utilisation

`fl.activeEffect`

## Description

Propriété en lecture seule ; l'[Objet Effect](#) de l'effet actuellement appliqué. La liste des propriétés disponibles pour `fl.activeEffect` figure dans la section « [Résumé des propriétés de l'objet Effect](#) », à la page 222.

## Exemple

L'exemple suivant enregistre dans la variable `ef` un objet représentant l'effet actuel :

```
var ef = fl.activeEffect;
```

# fl.addEventListener()

## Disponibilité

Flash CS3 Professional.

## Utilisation

`fl.addEventListener(eventType, callbackFunction)`

## Paramètres

*eventType* Chaîne spécifiant le type d'événement à transmettre à cette fonction de rappel.

Les valeurs gérées sont les suivantes : "documentNew", "documentOpened",

"documentClosed", "mouseMove", "documentChanged", "layerChanged" et

"frameChanged".

### REMARQUE

La valeur `documentChanged` ne signifie pas que le contenu d'un document a été modifié, mais qu'un autre document se trouve à présent au premier plan. Ainsi, `fl.getDocumentDOM()` renverra une valeur différente de celle qui a été envoyée avant que l'événement se produise.

*callbackFunction* Chaîne spécifiant la fonction que vous voulez exécuter chaque fois que l'événement se produit.

## Valeur renvoyée

Aucune.

## Description

Méthode ; enregistre une fonction à appeler quand un événement particulier se produit.

Si vous décidez d'utiliser cette méthode, sachez que lorsque l'événement se produit fréquemment (comme cela peut être le cas avec `mouseMove`) et que la fonction demande un certain temps pour s'exécuter, votre application peut se bloquer ou éventuellement devenir instable et générer des erreurs.

## Exemple

L'exemple suivant affiche un message dans le panneau Sortie à la fermeture d'un document :

```
myFunction = function () {  
    fl.trace('document was closed'); }  
fl.addEventListener("documentClosed", myFunction);
```

## Voir aussi

[fl.removeEventListener\(\)](#)

# fl.as3PackagePaths

## Disponibilité

Flash CS3 Professional.

## Utilisation

`fl.as3PackagePaths`

## Description

Propriété ; chaîne correspondant au paramètre de chemin de classe global dans la boîte de dialogue des paramètres d'ActionScript 3.0. Les chemins d'accès de classe sont délimités par des points-virgules (;) dans une chaîne. Pour afficher ou modifier des paramètres de chemin de classe d'ActionScript 2.0 utilisez [fl.packagePaths](#).

## Exemple

L'exemple suivant montre comment modifier les paramètres de chemin de classe d'ActionScript 3.0 :

```
fl.trace(fl.as3PackagePaths);  
// Sortie (en supposant un démarrage avec les valeurs par défaut)  
// .;$(AppConfig)/ActionScript 3.0/Classes  
fl.as3PackagePaths="buying;selling";  
fl.trace(fl.as3PackagePaths);  
// Sortie  
// buying (achat); selling (vente)
```

## Voir aussi

[fl.resetAS3PackagePaths\(\)](#)

# fl.browseForFileURL()

## Disponibilité

Flash MX 2004.

## Utilisation

```
fl.browseForFileURL(browseType [, title [, previewArea]])
```

## Paramètres

*browseType* Chaîne spécifiant le type d'opération à effectuer sur les fichiers. Les valeurs gérées sont "open" (ouvrir), "select" (sélectionner) ou "save" (enregistrer). Les valeurs "open" et "select" ouvrent la boîte de dialogue d'ouverture de fichier. Chaque valeur est prévue pour offrir la compatibilité avec Dreamweaver. La valeur "save" ouvre la boîte de dialogue d'enregistrement de fichier.

*title* Chaîne indiquant le titre de la boîte de dialogue d'ouverture ou d'enregistrement de fichier. Si ce paramètre est omis, une chaîne par défaut est utilisée. Ce paramètre est facultatif.

*previewArea* Paramètre facultatif qui est ignoré par Flash et Fireworks ; il sert uniquement à assurer la compatibilité avec Dreamweaver.

## Valeur renvoyée

URL du fichier, exprimée sous la forme « fichier :///URI ». Renvoie la valeur `null` si l'utilisateur a fermé la boîte de dialogue en cliquant sur le bouton Annuler.

## Description

Méthode ; ouvre une boîte de dialogue Fichier > Ouvrir ou Fichier > Enregistrer pour permettre à l'utilisateur d'indiquer un nom de fichier à ouvrir ou à enregistrer.

## Exemple

L'exemple suivant permet à l'utilisateur de choisir un fichier FLA, et ouvre ce fichier. (La méthode `fl.browseForFileURL()` permet de rechercher tous les types de fichiers, mais `fl.openDocument()` ne permet d'ouvrir que des fichiers FLA.)

```
var fileURL = fl.browseForFileURL("open", "Select file");  
var doc = fl.openDocument(fileURL);
```

## Voir aussi

[fl.browseForFolderURL\(\)](#)

# fl.browseForFolderURL()

## Disponibilité

Flash 8.

## Utilisation

```
fl.browseForFolderURL([description])
```

## Paramètres

*description* Chaîne facultative spécifiant la description de la boîte de dialogue Rechercher un dossier. Si ce paramètre est omis, rien n'apparaît dans la zone de description.

## Valeur renvoyée

URL du fichier, exprimée sous la forme « fichier :///URI ». Renvoie la valeur `null` si l'utilisateur a fermé la boîte de dialogue en cliquant sur le bouton Annuler.

## Description

Méthode ; affiche la boîte de dialogue Rechercher un dossier et permet à l'utilisateur de sélectionner un dossier.

### REMARQUE

Le titre de la boîte de dialogue est toujours « Rechercher un dossier ». Utilisez le paramètre *description* pour ajouter sous le titre des informations dans la zone de description, par exemple « Sélectionnez un dossier » ou « Sélectionnez le chemin contenant le profil à importer ».

## Exemple

L'exemple suivant permet à l'utilisateur de sélectionner un dossier, puis affiche la liste des fichiers contenus dans ce dossier :

```
var folderURI = fl.browseForFolderURL("Select a folder.");  
var folderContents = FLfile.listFolder(folderURI);
```

## Voir aussi

[fl.browseForFileURL\(\)](#), [Objet FLfile](#)



# fl.clipCopyString()

## Disponibilité

Flash CS3 Professional.

## Utilisation

```
fl.clipCopyString(string)
```

## Paramètres

*string* Chaîne à copier dans le Presse-papiers.

## Valeur renvoyée

Aucune.

## Description

Méthode ; copie la chaîne spécifiée dans le Presse-papiers.

Pour copier la sélection actuelle dans le Presse-papiers, utilisez `document.clipCopy()`.

## Exemple

L'exemple suivant copie le chemin d'accès du document actif dans le Presse-papiers :

```
var documentPath = fl.getDocumentDOM().path;  
fl.clipCopyString(documentPath);
```

# fl.closeAll()

## Disponibilité

Flash MX 2004.

## Utilisation

```
fl.closeAll([bPromptToSave])
```

## Paramètres

*bPromptToSave* Valeur booléenne facultative indiquant quelle boîte de dialogue doit s'afficher : Enregistrer, pour les fichiers modifiés depuis leur enregistrement ou Enregistrer sous, pour les fichiers qui n'ont jamais été enregistrés. La valeur par défaut est `true`.

## Valeur renvoyée

Aucune.

## Description

Méthode ; ferme tous les fichiers ouverts (fichiers FLA, SWF, JSFL, etc.). Si vous voulez fermer tous ces fichiers sans enregistrer une seule modification dans l'un d'entre eux, transmettez la valeur `false` pour *bPromptToSave*. Cette méthode ne ferme pas l'application.

## Exemple

Le code suivant ferme tous les fichiers ouverts en invitant l'utilisateur à enregistrer les fichiers nouveaux ou modifiés.

```
fl.closeAll();
```

## Voir aussi

[fl.closeAllPlayerDocuments\(\)](#), [fl.closeDocument\(\)](#)

# fl.closeAllPlayerDocuments()

## Disponibilité

Flash CS3 Professional.

## Utilisation

```
fl.closeAllPlayerDocuments()
```

## Paramètres

Aucun.

## Valeur renvoyée

Valeur booléenne : `true` si une ou plusieurs fenêtres d'animation étaient ouvertes, `false` dans le cas contraire.

## Description

Méthode ; ferme tous les fichiers SWF qui ont été ouverts par le biais de Contrôle > Tester l'animation.

## Exemple

L'exemple suivant ferme tous les fichiers SWF qui ont été ouverts par le biais de Contrôle > Tester l'animation.

```
fl.closeAllcloseAllPlayerDocuments();
```

## Voir aussi

[fl.closeAll\(\)](#), [fl.closeDocument\(\)](#)

# fl.closeDocument()

## Disponibilité

Flash MX 2004.

## Utilisation

```
fl.closeDocument(documentObject [, bPromptToSaveChanges])
```

## Paramètres

*documentObject* [Objet Document](#). Si *documentObject* fait référence au document actif, il est possible que la fenêtre Document ne se ferme pas avant la fin de l'exécution du script qui appelle cette méthode.

*bPromptToSaveChanges* Valeur booléenne. Lorsque la valeur de *bPromptToSaveChanges* est *false*, l'utilisateur n'est pas invité à enregistrer le document si celui-ci contient des modifications non enregistrées ; le fichier est simplement fermé et les changements ignorés. Si *bPromptToSaveChanges* prend la valeur *true* et si le document contient des modifications non enregistrées, l'utilisateur en est prévenu par la boîte de dialogue d'enregistrement standard avec les boutons Oui et Non. La valeur par défaut est *true*. Ce paramètre est facultatif.

## Valeur renvoyée

Valeur booléenne : *true* si l'opération réussie, sinon *false*.

## Description

Méthode ; ferme le document spécifié.

## Exemple

L'exemple suivant illustre deux façons de fermer un document.

```
// Fermer le document spécifié en invitant l'utilisateur à enregistrer
// les changements.
fl.closeDocument(fl.documents[0]);
fl.closeDocument(fl.documents[0] , true); // L'utilisation de true
// est facultative.
// Fermer le document spécifié sans inviter l'utilisateur à enregistrer
// les changements.
fl.closeDocument(fl.documents[0], false);
```

## Voir aussi

[fl.closeAll\(\)](#)

# fl.closeProject()

## Disponibilité

Flash 8.

## Utilisation

```
fl.closeProject()
```

## Paramètres

Aucun.

## Valeur renvoyée

Valeur booléenne `true` si le projet est correctement fermé, `false` si aucun fichier de projet n'est ouvert.

## Description

Méthode ; ferme le fichier du projet Flash (FLP, Flash Project) actuellement ouvert.

## Exemple

L'exemple suivant tente de fermer un fichier de projet, puis affiche un message informant de la réussite ou de l'échec de l'opération :

```
fl.trace("The project was" + (fl.closeProject() ? "closed" : "not  
closed"));
```

## Voir aussi

[fl.getProject\(\)](#), [fl.openProject\(\)](#), [Objet Project](#)

# fl.compilerErrors

## Disponibilité

Flash CS3 Professional.

## Utilisation

```
fl.compilerErrors
```

## Description

Propriété en lecture seule ; objet `compilerErrors` représentant le panneau des erreurs. Pour plus d'informations sur l'utilisation de cette propriété, consultez la section [Objet compilerErrors](#).

# fl.componentsPanel

## Disponibilité

Flash MX 2004.

## Utilisation

`fl.componentsPanel`

## Description

Propriété en lecture seule ; [Objet componentsPanel](#) représentant le panneau Composants.

## Exemple

L'exemple suivant stocke un objet `componentsPanel` dans la variable `comPanel` :

```
var comPanel = fl.componentsPanel;
```

# fl.configDirectory

## Disponibilité

Flash MX 2004.

## Utilisation

`fl.configDirectory`

## Description

Propriété en lecture seule ; chaîne indiquant le chemin d'accès complet du dossier Configuration de l'utilisateur local. Le format de ce chemin est spécifique à chaque plate-forme. Pour spécifier ce chemin sous la forme « fichier:/// URI », ce qui n'est pas spécifique à une plate-forme, utilisez [fl.configURI](#).

## Exemple

L'exemple suivant affiche le dossier Configuration dans le panneau Sortie.

```
fl.trace("My local configuration directory is " + fl.configDirectory);
```

# fl.configURI

## Disponibilité

Flash MX 2004.

## Utilisation

`fl.configURI`

## Description

Propriété en lecture seule ; chaîne indiquant le chemin d'accès complet du dossier Configuration de l'utilisateur local sous la forme « fichier:/// URI ». Voir aussi [fl.configDirectory](#).

## Exemple

L'exemple suivant exécute le script spécifié. L'utilisation de `fl.configURI` permet de spécifier l'emplacement du script quelle que soit la plate-forme sur laquelle le script est exécuté.

```
// Pour exécuter une commande du menu Commandes, remplacez "Test.jsfl"  
// par la commande à exécuter dans la ligne suivante.  
fl.runScript( fl.configURI + "Commands/Test.jsfl" );
```

# fl.contactSensitiveSelection

## Disponibilité

Flash 8.

## Utilisation

`fl.contactSensitiveSelection`

## Description

Valeur booléenne précisant si le mode de sélection Activer au contact est activé (`true`) ou désactivé (`false`).

## Exemple

L'exemple suivant montre comment désactiver le mode de sélection Activer au contact avant d'effectuer une sélection, puis comment lui rendre sa valeur initiale après avoir effectué la sélection :

```
var contact = fl.contactSensitiveSelection;  
fl.contactSensitiveSelection = false;  
// Insérer le code de sélection ici.  
fl.contactSensitiveSelection = contact;
```

# fl.createDocument()

## Disponibilité

Flash MX 2004.

## Utilisation

```
fl.createDocument([docType])
```

## Paramètres

*docType* Chaîne spécifiant le type de document à créer. Les valeurs gérées sont "timeline" (scénario), "presentation" et "application". La valeur par défaut est "timeline" et revient à sélectionner Fichier > Nouveau > Fichier Flash (ActionScript 3.0). Ce paramètre est facultatif.

### REMARQUE

Cette méthode ne prend pas en charge la création d'un document Flash Mobile.

## Valeur renvoyée

En cas de succès de la méthode, l'objet Document associé au nouveau document. En cas d'erreur, elle renvoie la valeur `undefined`.

## Description

Méthode ; ouvre un nouveau document et le sélectionne. Les valeurs de taille, résolution et couleur sont les valeurs par défaut actuelles.

## Exemple

L'exemple suivant crée différents types de documents :

```
// Créer deux documents Flash de type scénario.  
fl.createDocument();  
fl.createDocument("timeline");  
// Créer un document Flash de type diaporama.  
fl.createDocument("presentation");  
// Créer un document de type application de formulaires.  
fl.createDocument("application");
```

# fl.createNewDocList

## Disponibilité

Flash MX 2004.

## Utilisation

`fl.createNewDocList`

## Description

Propriété en lecture seule ; tableau de chaînes représentant les divers types de documents qu'il est possible de créer.

## Exemple

L'exemple suivant affiche dans le panneau Sortie les types de documents qu'il est possible de créer :

```
fl.trace("Number of choices " + fl.createNewDocList.length);
for (i = 0; i < fl.createNewDocList.length; i++)
    fl.trace("choice: " + fl.createNewDocList[i]);
```

# fl.createNewDocListType

## Disponibilité

Flash MX 2004.

## Utilisation

`fl.createNewDocListType`

## Description

Propriété en lecture seule ; tableau de chaînes représentant les extensions de fichiers des divers types de documents qu'il est possible de créer. Les entrées du tableau correspondent directement (à indice identique) aux entrées du tableau `fl.createNewDocList`.

## Exemple

L'exemple suivant affiche dans le panneau Sortie les extensions des types de documents qu'il est possible de créer :

```
fl.trace("Number of types " + fl.createNewDocListType.length);
for (i = 0; i < fl.createNewDocListType.length; i++) fl.trace("type: " +
    fl.createNewDocListType[i]);
```



# fl.createNewTemplateList

## Disponibilité

Flash MX 2004.

## Utilisation

`fl.createNewTemplateList`

## Description

Propriété en lecture seule ; tableau de chaînes représentant les divers types de modèles qu'il est possible de créer.

## Exemple

L'exemple suivant affiche dans le panneau Sortie les types de modèles qu'il est possible de créer :

```
fl.trace("Number of template types: " + fl.createNewTemplateList.length);  
for (i = 0; i < fl.createNewTemplateList.length; i++) fl.trace("type: " +  
    fl.createNewTemplateList[i]);
```

# fl.createProject()

## Disponibilité

Flash 8.

## Utilisation

`fl.createProject(fileURI [ , name ])`

## Paramètres

*fileURI* Chaîne de type « fichier:/// URI » spécifiant le nom du fichier de projet Flash à créer.

*name* Chaîne facultative qui s'affiche comme nom du projet dans le panneau Projet. Si le paramètre *name* est omis, c'est le nom du fichier FLP (excluant le chemin d'accès ou l'extension) qui est affiché dans le panneau Projet.

## Valeur renvoyée

Renvoie un [Objet Project](#) en cas de réussite de la méthode, `undefined` s'il est impossible de créer le fichier (par exemple, si *fileURI* contient un dossier qui n'existe pas).

## Description

Méthode ; crée un fichier de projet Flash (FLP, Flash Project) avec le nom indiqué. S'il est impossible de créer le fichier, une boîte de dialogue d'information s'affiche. Si le fichier existe déjà, une boîte de dialogue s'affiche et demande si le fichier doit être écrasé.

## Exemple

L'exemple suivant crée un fichier de projet dans le répertoire indiqué (s'il existe) et spécifie le nom à afficher dans le panneau Projet :

```
var myProject = fl.createProject("file:///C:/Projects/  
    MasterProject_2005.flp", "Master Project");
```

## Voir aussi

[fl.getProject\(\)](#), [fl.openProject\(\)](#), [Objet Project](#)

# fl.documents

## Disponibilité

Flash MX 2004.

## Utilisation

`fl.documents`

## Description

Propriété en lecture seule ; tableau des objets Document (consultez la section [Objet Document](#)) représentant les documents (fichiers FLA) ouverts pour modification.

## Exemple

L'exemple suivant enregistre dans la variable `docs` un tableau des documents ouverts :

```
var docs = fl.documents;
```

L'exemple suivant affiche dans le panneau Sortie les noms de tous les documents ouverts :

```
for (doc in fl.documents) {  
    fl.trace(fl.documents[doc].name);  
}
```

# fl.downloadLatestVersion()

## Disponibilité

Flash CS3 Professional.

## Utilisation

```
fl.downloadLatestVersion(fileURI)
```

## Paramètres

*fileURI* Chaîne exprimée sous la forme « fichier:/// URI » et précisant le chemin d'accès local du fichier à télécharger depuis le serveur Version Cue. Seuls les fichiers qui ne sont pas encore ouverts peuvent être téléchargés. Si le fichier spécifié par *fileURI* est déjà ouvert, cette méthode n'a aucun effet.

## Valeur renvoyée

Valeur booléenne `true` si le téléchargement du fichier a été correctement effectué, `false` dans le cas contraire.

## Description

Méthode ; télécharge, à partir du serveur Version Cue, la version la plus récente d'un fichier qui n'est pas actuellement ouvert. Pour télécharger la dernière version d'un fichier ouvert, utilisez `document.synchronizeWithHeadVersion()`.

## Exemple

L'exemple suivant télécharge le fichier nommé `myFile.fla` à partir du serveur Version Cue :

```
fl.downloadLatestVersion("file:///C:/MyFiles/Version Cue/docs/myFile.fla");
```

## Voir aussi

```
document.synchronizeWithHeadVersion(), fl.revertDocumentToLastVersion(),  
fl.saveAVersionOfDocument(), fl.synchronizeDocumentWithHeadVersion()
```

# fl.drawingLayer

## Disponibilité

Flash MX 2004.

## Utilisation

`fl.drawingLayer`

## Description

Propriété en lecture seule ; [Objet drawingLayer](#) que doit utiliser un outil extensible lorsque l'utilisateur dessine un tracé temporaire à l'aide de la souris (par exemple, pour créer un rectangle de sélection).

## Exemple

Voir la section [drawingLayer.setColor\(\)](#).

# fl.effects

## Disponibilité

Flash MX 2004.

## Utilisation

`fl.effects`

## Description

Propriété en lecture seule ; tableau des objets Effect (consultez la section [Objet Effect](#)), basé sur le fichier de paramètres XML. Il ne s'agit pas des effets mais de la description des effets. La longueur du tableau correspond au nombre d'effets (basé sur le fichier XML de définition des paramètres, pas sur le nombre de fichiers d'implémentation JSFL) enregistrés au lancement du programme.

## Exemple

L'exemple suivant renvoie le premier effet enregistré :

```
ef = fl.effects[0]
```

# fl.enableImmediateUpdates()

## Disponibilité

Flash MX 2004.

## Utilisation

```
fl.enableImmediateUpdates(bEnableUpdates)
```

## Paramètres

*bEnableUpdates* Valeur booléenne spécifiant si l'actualisation immédiate du scénario doit être activée (*true*) ou désactivée (*false*) lors de l'exécution d'effets.

## Valeur renvoyée

Aucune.

## Description

Méthode ; permet au développeur du script d'activer l'actualisation immédiate du scénario lors de l'exécution d'effets. L'actualisation immédiate est normalement désactivée, afin que l'utilisateur ne voie pas les étapes intermédiaires, ce qui peut être déroutant et nécessiter plus de temps que nécessaire. Cette méthode est uniquement destinée au débogage, et il est déconseillé de l'utiliser dans des fichiers déployés. Lorsque l'effet est terminé, l'état interne est réinitialisé (l'actualisation immédiate du scénario est inhibée).

## Exemple

L'exemple suivant active l'actualisation immédiate.

```
fl.enableImmediateUpdates(true) ;  
fl.trace("Immediate updates are enabled");
```

# fl.fileExists()

## Disponibilité

Flash MX 2004.

## Utilisation

```
fl.fileExists(fileURI)
```

## Paramètres

*fileURI* Chaîne exprimée sous la forme « fichier:/// URI » et contenant le chemin d'accès du fichier.

## Valeur renvoyée

Valeur booléenne : *true* si le fichier existe, sinon *false*.

## Description

Méthode ; vérifie si un fichier existe déjà sur le disque dur.

## Exemple

Pour chaque fichier spécifié, l'exemple suivant affiche dans le panneau Sortie la valeur `true` si le fichier existe, `false` dans le cas contraire.

```
alert(fl.fileExists("file:///C:/example fla"));
alert(fl.fileExists("file:///C:/example.jsfl"));
alert(fl.fileExists(""));
```

# fl.findDocumentDOM()

## Disponibilité

Flash CS3 Professional.

## Utilisation

```
fl.findDocumentDOM(id)
```

## Paramètres

*id* Entier représentant un identificateur unique pour un document.

## Valeur renvoyée

Un objet Document, ou la valeur `null` s'il n'existe aucun document doté de l'*id* spécifié.

## Description

Méthode ; permet de cibler un fichier spécifique au moyen de son identificateur unique (au lieu d'utiliser sa valeur d'index, par exemple). Utilisez cette méthode conjointement avec [document.id](#).

## Exemple

L'exemple suivant illustre la lecture de l'ID d'un document, puis son utilisation pour cibler le document en question.

```
var originalDocID = fl.getDocumentDOM().id;
// autre code ici, fonctionne peut-être dans des fichiers différents
var targetDoc = fl.findDocumentDOM(originalDocID);
// Définir la hauteur de la scène sur 400 pixels dans le document original.
targetDoc.height = 400;
```

## Voir aussi

[fl.findDocumentIndex\(\)](#)

# fl.findDocumentIndex()

## Disponibilité

Flash MX 2004.

## Utilisation

```
fl.findDocumentIndex(name)
```

## Paramètres

*name* Nom du document dont vous voulez connaître l'index. Ce document doit être ouvert.

## Valeur renvoyée

Un tableau d'entiers représentant la position du document *name* dans le tableau

`fl.documents`.

## Description

Méthode ; renvoie un tableau d'entiers représentant la position du document *name* dans le tableau `fl.documents`. Il se peut que plusieurs documents du même nom soient ouverts (s'ils se trouvent dans des dossiers différents).

## Exemple

L'exemple suivant affiche, dans le panneau Sortie, des informations sur la position d'index de tout fichier appelé `test fla` :

```
var filename = "test fla"
var docIndex = fl.findDocumentIndex(filename);
for (var index in docIndex)
    fl.trace(filename + " is open at index " + docIndex[index]);
```

## Voir aussi

[fl.documents](#), [fl.findDocumentDOM\(\)](#)

# fl.findObjectInDocByName()

## Disponibilité

Flash CS3 Professional.

## Utilisation

```
fl.findObjectInDocByName(instanceName, document)
```

## Paramètres

*instanceName* Chaîne indiquant intégralement ou partiellement le nom d'occurrence d'un élément dans le document spécifié.

*document* L'[Objet Document](#) dans lequel rechercher l'élément spécifié.

## Valeur renvoyée

Un tableau d'objets génériques. Utilisez la propriété `.obj` de chaque élément dans le tableau pour obtenir l'objet. Cet objet possède les propriétés suivantes : `keyframe` (image-clé), `layer` (calque), `timeline` (scénario) et `parent`. Vous pouvez utiliser ces propriétés pour accéder à la hiérarchie de l'objet. Pour plus d'informations sur ces propriétés et la façon d'y accéder, voir [fl.findObjectInDocByName\(\)](#).

Vous pouvez également accéder aux méthodes et aux propriétés des valeurs `layer` et `timeline` ; il s'agit respectivement des équivalents de l'[Objet Layer](#) et de l'[Objet Timeline](#).

## Description

Méthode ; expose des éléments avec des noms d'occurrences contenant un texte particulier dans un document.

## Exemples

L'exemple suivant effectue une recherche dans le document actif pour trouver des champs texte contenant "text" dans leur nom d'occurrence, puis il modifie le contenu de ces champs texte :

```
var nameToSearchFor = "text";
var doc = fl.getDocumentDOM();
var results = fl.findObjectInDocByName(nameToSearchFor, doc);
if (results.length > 0) {
    for (var i = 0; i < results.length; i++) {
        results[i].obj.setTextString("new text");
    }
    alert("success, found " + results.length + " objects");
}
else {
    alert("failed, no objects of type " + nameToSearchFor + " found");
}
```

## Voir aussi

[fl.findObjectInDocByType\(\)](#)



# fl.findObjectInDocByType()

## Disponibilité

Flash CS3 Professional.

## Utilisation

`fl.findObjectInDocByType(elementType, document)`

## Paramètres

*elementType* Chaîne représentant le type d'élément à rechercher. Pour connaître les valeurs gérées, consultez la section [element.elementType](#).

*document* L'[Objet Document](#) dans lequel rechercher l'élément spécifié.

## Valeur renvoyée

Un tableau d'objets génériques. Utilisez la propriété `.obj` de chaque élément dans le tableau pour obtenir l'objet Element. Chaque objet possède les propriétés suivantes : `keyframe` (image-clé), `layer` (calque), `timeline` (scénario) et `parent`. Vous pouvez utiliser ces propriétés pour accéder à la hiérarchie de l'objet.

Vous pouvez également accéder aux méthodes et aux propriétés des valeurs `layer` et `timeline` ; il s'agit respectivement des équivalents de l'[Objet Layer](#) et de l'[Objet Timeline](#).

Le deuxième et le troisième exemple de la section Exemples montrent comment accéder à ces propriétés.

## Description

Méthode ; expose des éléments d'un type particulier dans un document.

## Exemple

L'exemple suivant recherche des champs texte dans le document actif, puis modifie leur contenu :

```
var doc = fl.getDocumentDOM();
var typeToSearchFor = "text";
var results = fl.findObjectInDocByType(typeToSearchFor, doc);
if (results.length > 0) {
    for (var i = 0; i < results.length; i++) {
        results[i].obj.setTextString("new text");
    }
    alert("success, found " + results.length + " objects");
}
else {
    alert("failed, no objects of type " + typeToSearchFor + " found");
}
```

L'exemple suivant montre comment accéder aux propriétés spéciales de l'objet renvoyé par cette méthode :

```
var doc = fl.getDocumentDOM();
var resultsArray = findObjectInDocByType("text", doc);
if (resultsArray.length > 0)
{
    var firstItem = resultsArray[0];

    // firstItem.obj- Il s'agit de l'objet Element trouvé.

    // Vous pouvez accéder aux propriétés suivantes de cet objet :
    // firstItem.keyframe- L'image-clé dans laquelle l'élément se trouve.
    // firstItem.layer- Le calque dans lequel l'image-clé se trouve.
    // firstItem.timeline- Le scénario dans lequel le calque se trouve.
    // firstItem.parent- Le parent du scénario. Par exemple,
    // le scénario peut se trouver dans une occurrence de symbole.
}
```

L'exemple suivant montre comment sauvegarder le DOM pour trouver le nom d'un calque dans lequel un champ texte a été trouvé par le biais de l'objet `resultArray.obj` :

```
var doc = fl.getDocumentDOM();
var typeToSearchFor = "text";
var resultsArray = fl.findObjectInDocByType(typeToSearchFor, doc);
if (resultsArray.length > 0) {
    for (var i = 0; i < resultsArray.length; i++) {
        resultsArray[i].obj.setTextString("new text");
        var firstItem = resultsArray[0];
        firstItemObj = firstItem.obj;
        fl.trace(firstItemObj.layer.name+" layerName");
    }
} else {
    alert("failed, no objects of type " + typeToSearchFor + " found");
}
```

### Voir aussi

[fl.findObjectInDocByName\(\)](#)

## fl.getAppMemoryInfo()

### Disponibilité

Flash 8 (Windows uniquement).

### Utilisation

`fl.getAppMemoryInfo(memType)`

## Paramètres

*memType* Nombre entier précisant la zone d'utilisation mémoire à demander. La liste des valeurs gérées est indiquée dans la description suivante.

## Valeur renvoyée

Un entier représentant le nombre d'octets utilisés dans la zone mémoire de Flash.exe spécifiée.

## Description

Méthode (Windows uniquement). Renvoie un entier représentant le nombre d'octets utilisés dans la zone mémoire de Flash.exe spécifiée. Utilisez le tableau suivant pour déterminer la valeur à transmettre comme *memType* :

<b>memType</b>	<b>Données de ressource</b>
0	PAGEFAULTCOUNT
1	PEAKWORKINGSETSIZE
2	WORKINGSETSIZE
3	QUOTAPEAKPAGEDPOOLUSAGE
4	QUOTAPAGEDPOOLUSAGE
5	QUOTAPEAKNONPAGEDPOOLUSAGE
6	QUOTANONPAGEDPOOLUSAGE
7	PAGEFILEUSAGE
8	PEAKPAGEFILEUSAGE

## Exemple

L'exemple suivant affiche l'utilisation de la mémoire de travail actuelle :

```
var memsize = fl.getAppMemoryInfo(2);
fl.trace("Flash current memory consumption is " + memsize + " bytes or " +
    memsize/1024 + " KB");
```

# fl.getDocumentDOM()

## Disponibilité

Flash MX 2004.

## Utilisation

```
fl.getDocumentDOM()
```

## Paramètres

Aucun.

## Valeur renvoyée

Un objet `Document`, ou la valeur `null` si aucun document n'est ouvert.

## Description

Méthode ; lit le DOM ([Objet Document](#)) du document actuellement actif (fichier FLA).

Si un ou plusieurs documents sont ouverts, mais qu'aucun d'eux n'est actif (si, par exemple, un fichier JSFL a le focus), la méthode lit le DOM du dernier document actif.

## Exemple

L'exemple suivant affiche dans le panneau de sortie le nom du document Flash actif, ou du document Flash qui a été actif le plus récemment.

```
var currentDoc = fl.getDocumentDOM();  
fl.trace(currentDoc.name);
```

# fl.getProject()

## Disponibilité

Flash 8.

## Utilisation

```
fl.getProject()
```

## Paramètres

Aucun.

## Valeur renvoyée

Un [Objet Project](#) représentant le projet actuellement ouvert. Si aucun projet n'est ouvert, la méthode renvoie `undefined`.

## Description

Méthode ; renvoie un [Objet Project](#) représentant le projet actuellement ouvert.

## Exemple

L'exemple suivant affiche dans le panneau de sortie le nom du projet actuellement ouvert :

```
fl.trace("Current project: " + fl.getProject().name);
```

## Voir aussi

[fl.createProject\(\)](#), [fl.openProject\(\)](#), [Objet Project](#)

# fl.mapPlayerURL()

## Disponibilité

Flash MX 2004.

## Utilisation

```
fl.mapPlayerURL(URI [, returnMBCS])
```

## Paramètres

*URI* Chaîne contenant l'adresse URL avec séquences d'échappement en Unicode à convertir.

*returnMBCS* Valeur booléenne devant être définie comme `true` pour que le chemin soit renvoyé converti en MBCS avec séquences d'échappement. Sinon, la méthode renvoie une chaîne en UTF-8. La valeur par défaut est `false`. Ce paramètre est facultatif.

## Valeur renvoyée

Une chaîne qui est l'adresse URL convertie.

## Description

Méthode ; convertit une adresse URL avec séquences d'échappement en Unicode en une URL en UTF-8 ou MBCS. Il est conseillé d'appeler cette méthode si la chaîne doit être utilisée en ActionScript pour accéder à une ressource externe. De plus, il est nécessaire de l'utiliser pour gérer des caractères sur deux octets.

## Exemple

L'exemple suivant convertit une URL en UTF-8 pour permettre à Flash Player de la charger :

```
var url = MMExecute( "fl.mapPlayerURL(" + myURL + ", false);" );  
mc.loadMovie( url );
```

# fl.Math

## Disponibilité

Flash MX 2004.

## Utilisation

```
fl.Math
```

## Description

Propriété en lecture seule ; l'[Objet Math](#) fournit les méthodes utilisées pour les opérations de matrice et de point.

## Exemple

L'exemple suivant montre la matrice de transformation de l'objet sélectionné et son inverse :

```
// Sélectionner un élément sur la scène, puis exécuter ce script.
var mat = fl.getDocumentDOM().selection[0].matrix;
for(var prop in mat){
    fl.trace("mat."+prop+" = " + mat[prop]);
}
var invMat = fl.Math.invertMatrix( mat );
for(var prop in invMat) {
    fl.trace("invMat."+prop+" = " + invMat[prop]);
}
```

## fl.mruRecentFileList

### Disponibilité

Flash MX 2004.

### Utilisation

fl.mruRecentFileList

### Description

Propriété en lecture seule ; tableau des noms de fichier complets de la liste des fichiers récemment utilisés (MRU, Most Recently Used) gérée par l'outil de programmation Flash.

## Exemple

L'exemple suivant affiche dans le panneau de sortie le nombre de fichiers récemment ouverts et le nom de chacun de ces fichiers :

```
fl.trace("Number of recently opened files: " +
    fl.mruRecentFileList.length);
for (i = 0; i < fl.mruRecentFileList.length; i++) fl.trace("file: " +
    fl.mruRecentFileList[i]);
```

# fl.mruRecentFileListType

## Disponibilité

Flash MX 2004.

## Utilisation

`fl.mruRecentFileListType`

## Description

Propriété en lecture seule ; tableau des types de fichiers de la liste des fichiers récemment utilisés (MRU, Most Recently Used) gérée par l'outil de programmation Flash. Ce tableau correspond à celui de la propriété `fl.mruRecentFileList`.

## Exemple

L'exemple suivant affiche dans le panneau de sortie le nombre de fichiers récemment ouverts et le type de chacun de ces fichiers :

```
fl.trace("Number of recently opened files: " +  
    fl.mruRecentFileListType.length);  
for (i = 0; i < fl.mruRecentFileListType.length; i++) fl.trace("type: " +  
    fl.mruRecentFileListType[i]);
```

# fl.objectDrawingMode

## Disponibilité

Flash 8.

## Utilisation

`fl.objectDrawingMode`

## Description

Propriété ; valeur booléenne déterminant si le mode de dessin d'objet est activé (`true`) ou si le mode de dessin de fusion est activé (`false`).

## Exemple

L'exemple suivant active/désactive le mode Dessin d'objet :

```
var toggleMode = fl.objectDrawingMode;  
if (toggleMode) {  
    fl.objectDrawingMode = false;  
} else {  
    fl.objectDrawingMode = true;  
}
```

# fl.openDocument()

## Disponibilité

Flash MX 2004.

## Utilisation

```
fl.openDocument(fileURI)
```

## Paramètres

*fileURI* Chaîne exprimée sous la forme « fichier:/// URI » et précisant le nom du fichier à ouvrir.

## Valeur renvoyée

En cas de succès de la méthode, l'[Objet Document](#) associé au document récemment ouvert. Si le fichier est introuvable, ou si ce n'est pas un fichier FLA valide, une erreur est affichée et l'exécution du script est interrompue.

## Description

Méthode ; ouvre dans une nouvelle fenêtre de document un document Flash (FLA) à modifier, et active cette fenêtre. Pour l'utilisateur, l'effet est identique au choix de l'option de menu Fichier > Ouvrir et à la sélection d'un fichier. Si le fichier spécifié est déjà ouvert, la fenêtre qui contient ce document passe au premier plan. La fenêtre qui contient le fichier spécifié devient le document sélectionné.

## Exemple

L'exemple suivant ouvre un fichier nommé Document.fla qui est enregistré dans le répertoire racine du lecteur C. Le code stocke un objet Document représentant ce document dans la variable `doc` et définit le document pour qu'il soit le document actuellement sélectionné. En d'autres termes, tant que la fenêtre active reste la même (tant que le focus n'est pas modifié), `fl.getDocumentDOM()` renvoie à ce document.

```
var doc = fl.openDocument("file:///c:/Document.fla");
```



# fl.openProject()

## Disponibilité

Flash MX 2004 ; valeur de retour modifiée dans Flash 8.

## Utilisation

```
fl.openProject(fileURI)
```

## Paramètres

*fileURI* Chaîne exprimée sous la forme « fichier:/// URI » et indiquant le chemin d'accès du fichier de projet Flash (FLP) à ouvrir.

## Valeur renvoyée

Aucune dans Flash MX 2004, un [Objet Project](#) dans Flash 8.

## Description

Méthode ; ouvre un fichier de projet Flash (FLP, Flash Project) à modifier dans l'outil de programmation Flash.

## Exemple

L'exemple suivant ouvre un fichier de projet nommé myProjectFile.flp et enregistré dans le répertoire racine du lecteur C.

```
fl.openProject("file:///c:/myProjectFile.flp");
```

## Voir aussi

[fl.closeProject\(\)](#), [fl.createProject\(\)](#), [fl.getProject\(\)](#), [Objet Project](#)

# fl.openScript()

## Disponibilité

Flash MX 2004.

## Utilisation

```
fl.openScript(fileURI)
```

## Paramètres

*fileURI* Chaîne exprimée sous la forme « fichier:/// URI » et précisant le chemin d'accès du fichier JSFL, AS, ASC, XML, TXT ou de tout autre fichier à charger dans l'éditeur de texte de Flash.

### Valeur renvoyée

Aucune.

### Description

Méthode ; ouvre un script (JSFL, AS, ASC) ou tout autre fichier (XML, TXT) dans l'éditeur de texte de Flash.

### Exemple

L'exemple suivant ouvre un fichier nommé my\_test.jsfl et enregistré dans le répertoire /temp du lecteur C :

```
fl.openScript("file:///c:/temp/my_test.jsfl");
```

## fl.outputPanel

### Disponibilité

Flash MX 2004.

### Utilisation

```
fl.outputPanel
```

### Description

Propriété en lecture seule ; référence à l'[Objet outputPanel](#).

### Exemple

Voir [Objet outputPanel](#).

## fl.packagePaths

### Disponibilité

Flash CS3 Professional.

### Utilisation

```
fl.packagePaths
```

### Description

Propriété ; chaîne correspondant au paramètre de chemin de classe global dans la boîte de dialogue des paramètres d'ActionScript 2.0. Les chemins d'accès de classe sont délimités par des points-virgules (;) dans une chaîne. Pour afficher ou modifier des paramètres de chemin de classe d'ActionScript 3.0 utilisez [fl.as3PackagePaths](#).

## Exemple

L'exemple suivant montre comment modifier les paramètres de chemin de classe d'ActionScript 2.0 :

```
fl.trace(fl.packagePaths);  
// Sortie (en supposant un démarrage avec les valeurs par défaut)  
// .;$(LocalData)/Classes  
fl.packagePaths="buying;selling";  
fl.trace(fl.packagePaths);  
// Sortie  
// buying (achat); selling (vente)
```

## Voir aussi

[fl.resetPackagePaths\(\)](#)

# fl.quit()

## Disponibilité

Flash MX 2004.

## Utilisation

```
fl.quit([bPromptIfNeeded])
```

## Paramètres

*bPromptIfNeeded* Valeur booléenne devant prendre la valeur `true` (par défaut) pour que l'utilisateur soit invité à enregistrer les documents modifiés. Si la valeur de ce paramètre est `false`, l'utilisateur ne sera pas invité à enregistrer les documents modifiés. Dans ce cas, toute modification des documents ouverts sera annulée, et l'application se fermera immédiatement. Bien que cette méthode soit utile pour le traitement de fichiers par lot, il est conseillé de l'utiliser avec précaution. Ce paramètre est facultatif.

## Valeur renvoyée

Aucune.

## Description

Méthode ; ferme Flash en proposant à l'utilisateur d'enregistrer les documents modifiés.

## Exemple

L'exemple suivant illustre la fermeture du programme en proposant ou non d'enregistrer les documents modifiés :

```
// Quitter en proposant à l'utilisateur d'enregistrer les documents
// modifiés.
fl.quit();
fl.quit(true); // L'utilisation de true est facultative.
// Quitter sans enregistrer les documents modifiés.
fl.quit(false);
```

## fl.reloadEffects()

### Disponibilité

Flash MX 2004.

### Utilisation

```
fl.reloadEffects()
```

### Paramètres

Aucun.

### Valeur renvoyée

Aucune.

### Description

Méthode ; recharge tous les descripteurs d'effets définis dans le dossier Configuration Effects de l'utilisateur. Elle permet de modifier rapidement les scripts durant le développement, donc d'améliorer les effets sans redémarrer l'application. Il est conseillé d'appeler cette méthode à partir d'une commande placée dans le dossier Commands.

## Exemple

L'exemple suivant est un script d'une ligne qui peut être placé dans le dossier Commands. S'il est nécessaire de recharger des effets, il est ensuite facile d'ouvrir le menu Commandes et d'exécuter le script.

```
fl.reloadEffects();
```

# fl.reloadTools()

## Disponibilité

Flash MX 2004.

## Utilisation

```
fl.reloadTools()
```

## Paramètres

Aucun.

## Valeur renvoyée

Aucune.

## Description

Méthode ; reconstruit le panneau Outils à partir du contenu du fichier toolconfig.xml.

Cette méthode n'est utilisée que lors de la création d'outils extensibles. Employez cette méthode quand vous devez recharger le panneau Outils, par exemple après avoir modifié le fichier JSFL définissant un outil qui se trouve déjà dans le panneau.

## Exemple

L'exemple suivant est un script d'une ligne qui peut être placé dans le dossier Commands. Lorsque vous voulez recharger le panneau Outils, exécutez le script depuis le menu Commandes.

```
fl.reloadTools();
```

# fl.removeEventListener()

## Disponibilité

Flash CS3 Professional.

## Utilisation

```
fl.removeEventListener(eventType)
```

## Paramètres

*eventType* Chaîne spécifiant le type d'événement à supprimer de cette fonction de rappel.

Les valeurs gérées sont les suivantes : "documentNew", "documentOpened", "documentClosed", "mouseMove", "documentChanged", "layerChanged" et "frameChanged".

### Valeur renvoyée

La valeur booléenne `true` si l'écouteur d'événement a été correctement supprimé, `false` si la fonction n'a jamais été ajoutée à la liste par le biais de la méthode `fl.addEventListener()`.

### Description

Annule l'inscription d'une fonction enregistrée à l'aide de `fl.removeEventListener()`.

### Exemple

L'exemple suivant retire l'écouteur d'événement associé à l'événement `documentClosed` :

```
fl.removeEventListener("documentClosed");
```

### Voir aussi

[fl.addEventListener\(\)](#)

## fl.resetAS3PackagePaths()

### Disponibilité

Flash CS3 Professional.

### Utilisation

```
fl.resetAS3PackagePaths()
```

### Paramètres

Aucun.

### Description

Méthode ; réinitialise le paramètre de chemin de classe global sur la valeur par défaut dans la boîte de dialogue des paramètres d'ActionScript 3.0. Pour le réinitialiser dans ActionScript 2.0, utilisez [fl.resetPackagePaths\(\)](#).

### Exemple

L'exemple suivant montre comment réinitialiser le paramètre de chemin de classe d'ActionScript 3.0 sur sa valeur par défaut :

```
fl.resetAS3PackagePaths();
```

### Voir aussi

[fl.as3PackagePaths](#)

# fl.resetPackagePaths()

## Disponibilité

Flash CS3 Professional.

## Utilisation

```
fl.resetPackagePaths()
```

## Paramètres

Aucun.

## Description

Méthode ; réinitialise le paramètre de chemin de classe global sur la valeur par défaut dans la boîte de dialogue des paramètres d'ActionScript 2.0. Pour le réinitialiser dans ActionScript 3.0, utilisez [fl.resetAS3PackagePaths\(\)](#).

## Exemple

L'exemple suivant montre comment réinitialiser le paramètre de chemin de classe d'ActionScript 2.0 sur sa valeur par défaut :

```
fl.resetPackagePaths();
```

## Voir aussi

[fl.packagePaths](#)

# fl.revertDocument()

## Disponibilité

Flash MX 2004.

## Utilisation

```
fl.revertDocument(documentObject)
```

## Paramètres

*documentObject* [Objet Document](#). Si *documentObject* fait référence au document actif, il est possible que la fenêtre Document ne soit pas rétablie avant la fin de l'exécution du script qui appelle cette méthode.

## Valeur renvoyée

Valeur booléenne : `true` en cas de rétablissement réussi, `false` dans le cas contraire.

## Description

Méthode ; restaure la dernière version enregistrée du document FLA spécifié. Contrairement à l'option de menu Fichier > Rétablir, cette méthode n'affiche pas de boîte de dialogue demandant à l'utilisateur de confirmer l'opération. Voir aussi `document.revert()` et `document.canRevert()`.

Pour remplacer la version du document spécifié par celle du serveur Version Cue, utilisez `fl.revertDocumentToLastVersion()`.

## Exemple

L'exemple suivant rétablit la dernière version enregistrée du document FLA actuel ; les éventuelles modifications effectuées depuis le dernier enregistrement sont perdues.

```
fl.revertDocument(fl.getDocumentDOM());
```

# fl.revertDocumentToLastVersion()

## Disponibilité

Flash CS3 Professional.

## Utilisation

```
fl.revertDocumentToLastVersion(documentObject)
```

## Paramètres

*documentObject*    [Objet Document](#).

## Valeur renvoyée

Valeur booléenne `true` si le document est correctement rétabli, `false` dans le cas contraire.

## Description

Méthode ; si le rétablissement du fichier est possible, remplace la version du document spécifié par celle du serveur Version Cue et consigne les éventuelles erreurs dans le panneau Sortie.

Pour rétablir la dernière version enregistrée localement d'un document, utilisez `fl.revertDocument()`.

## Exemple

L'exemple suivant remplace la version du document actif par celle qui est stockée sur le serveur Version Cue :

```
fl.revertDocumentToLastVersion(fl.getDocumentDOM());
```



## Voir aussi

`document.revertToLastVersion()`, `fl.downloadLatestVersion()`,  
`fl.saveAVersionOfDocument()`, `fl.synchronizeDocumentWithHeadVersion()`

# fl.runScript()

## Disponibilité

Flash MX 2004.

## Utilisation

```
fl.runScript(fileURI [, funcName [, arg1, arg2, ...]])
```

## Paramètres

*fileURI* Chaîne exprimée sous la forme « fichier:/// URI » et précisant le nom du fichier de script à exécuter.

*funcName* Chaîne identifiant une fonction à exécuter dans le fichier JSFL spécifié dans le paramètre *fileURI*. Ce paramètre est facultatif.

*arg* Paramètre facultatif spécifiant un ou plusieurs arguments à passer à *funcname*.

## Valeur renvoyée

Une chaîne si *funcName* est spécifié, aucune valeur dans le cas contraire.

## Description

Méthode ; exécute un fichier JavaScript. Si une fonction est spécifiée dans l'un des arguments, la méthode exécute cette fonction et tout le code du script qui ne fait pas partie de cette fonction. Le reste du code du script est exécuté avant la fonction.

## Exemple

Supposons qu'un fichier de script appelé testScript.jsfl réside dans le répertoire racine du lecteur C et qu'il contienne les éléments suivants :

```
function testFunc(num, minNum) {  
    fl.trace("in testFunc: 1st arg: " + num + " 2nd arg: " + minNum);  
}  
for (i=0; i<2; i++) {  
    fl.trace("in for loop i=" + i);  
}  
fl.trace("end of for loop");  
// Fin de testScript.jsfl
```

Si vous émettez la commande suivante

```
fl.runScript("file:///C:/testScript.jsfl", "testFunc", 10, 1);
```

les informations suivantes apparaissent dans le panneau Sortie :

```
in for loop i=0  
in for loop i=1  
end of for loop  
in testFunct: 1st arg: 10 2nd arg: 1
```

Il est également possible d'appeler simplement `testScript.jsfl` sans exécuter de fonction, comme suit :

```
fl.runScript("file:///C:/testScript.jsfl");
```

Le résultat suivant s'affiche dans le panneau Sortie :

```
in for loop i=0  
in for loop i=1  
end of for loop
```

## fl.saveAll()

### Disponibilité

Flash MX 2004.

### Utilisation

```
fl.saveAll()
```

### Paramètres

Aucun.

### Valeur renvoyée

Aucune.

### Description

Méthode ; enregistre tous les documents ouverts.

#### REMARQUE

Si le fichier n'a jamais été enregistré ou n'a pas été modifié depuis son dernier enregistrement, le fichier n'est pas enregistré. Pour permettre la sauvegarde d'un fichier qui n'a pas été enregistré, ni modifié, utilisez `fl.saveDocumentAs()`.

### Exemple

L'exemple suivant enregistre tous les documents ouverts qui ont été précédemment enregistrés et modifiés depuis leur dernier enregistrement.

```
fl.saveAll();
```

### Voir aussi

`document.save()`, `document.saveAndCompact()`, `fl.saveDocument()`,  
`fl.saveDocumentAs()`

## fl.saveAVersionOfDocument()

### Disponibilité

Flash CS3 Professional.

### Utilisation

`fl.saveAVersionOfDocument(document)`

### Paramètres

*document*    [Objet Document](#).

### Valeur renvoyée

Valeur booléenne `true` si une version du document est correctement enregistrée sur le serveur Version Cue ; `false` sinon.

### Description

Méthode ; si le fichier peut être enregistré sur le serveur Version Cue, affiche une boîte de dialogue permettant à l'utilisateur d'entrer des commentaires sur la version, enregistre une version du document spécifié sur le serveur et consigne les éventuelles erreurs dans le panneau Sortie.

### Exemple

L'exemple suivant enregistre le document actif sur le serveur Version Cue :

```
fl.saveAVersionOfDocument(fl.getDocumentDOM());
```

### Voir aussi

[document.saveAVersion\(\)](#)

# fl.saveDocument()

## Disponibilité

Flash MX 2004.

## Utilisation

```
fl.saveDocument(document [, fileURI])
```

## Paramètres

*document* **Objet Document** spécifiant le document à enregistrer. Si la valeur de *document* est `null`, le document actif est enregistré.

*fileURI* Chaîne exprimée sous la forme « fichier:/// URI » et précisant le nom du document enregistré. Si le paramètre *fileURI* est `null` ou omis, le document est enregistré sous son nom actuel. Ce paramètre est facultatif.

## Valeur renvoyée

Valeur booléenne : `true` en cas de réussite de l'enregistrement, `false` sinon.

### REMARQUE

Si le fichier n'a jamais été enregistré, ou s'il n'a pas été modifié depuis son dernier enregistrement, il n'est pas enregistré et la valeur `false` est renvoyée. Pour permettre la sauvegarde d'un fichier qui n'a pas été enregistré, ni modifié, utilisez `fl.saveDocumentAs()`.

## Description

méthode ; enregistre le document spécifié sous forme de fichier FLA.

## Exemple

L'exemple suivant enregistre le document actif et deux autres documents spécifiés :

```
// Enregistrer le document actif.  
alert(fl.saveDocument(fl.getDocumentDOM()));  
// Enregistrer les documents spécifiés.  
alert(fl.saveDocument(fl.documents[0], "file:///C:/example1 fla"));  
alert(fl.saveDocument(fl.documents[1], "file:///C:/example2 fla"));
```

## Voir aussi

`document.save()`, `document.saveAndCompact()`, `fl.saveAll()`, `fl.saveDocumentAs()`

# fl.saveDocumentAs()

## Disponibilité

Flash MX 2004.

## Utilisation

```
fl.saveDocumentAs(document)
```

## Paramètres

*document* [Objet Document](#) spécifiant le document à enregistrer. Si *document* est `null`, le document actif est enregistré.

## Valeur renvoyée

Valeur booléenne : `true` en cas de succès de l'opération Enregistrer sous, sinon `false`.

## Description

Méthode ; affiche la boîte de dialogue Enregistrer sous pour le document spécifié.

## Exemple

L'exemple suivant invite l'utilisateur à enregistrer le document spécifié, puis affiche un message d'alerte qui indique si le document a été enregistré :

```
alert(fl.saveDocumentAs(fl.documents[1]));
```

## Voir aussi

[document.save\(\)](#), [document.saveAndCompact\(\)](#), [fl.saveAll\(\)](#), [fl.saveDocument\(\)](#)

# fl.scriptURI

## Disponibilité

Flash CS3 Professional.

## Utilisation

```
fl.scriptURI
```

## Description

Propriété en lecture seule ; chaîne représentant le chemin d'accès du script JSFL actuellement en cours d'exécution et exprimée sous la forme « fichier:/// URI ». Si le script a été appelé à partir de [fl.runScript\(\)](#), cette propriété représente le chemin d'accès du script parent immédiat. Ainsi, de multiples appels à [fl.runScript\(\)](#) sont évités pour trouver le chemin d'accès du script de l'appel original.

## Exemple

L'exemple suivant affiche dans le panneau Sortie le chemin d'accès du script JSFL actuellement en cours d'exécution :

```
fl.trace(fl.scriptURI);
```

## Voir aussi

[fl.runScript\(\)](#)

# fl.selectElement()

## Disponibilité

Flash CS3 Professional.

## Utilisation

```
fl.selectElement(elementObject, editMode)
```

## Paramètres

*elementObject* L'[Objet Element](#) que vous voulez sélectionner.

*editMode* Valeur booléenne indiquant si vous voulez modifier l'élément (`true`) ou uniquement le sélectionner (`false`).

## Valeur renvoyée

La valeur booléenne `true` si l'élément a été correctement sélectionné, `false` dans le cas contraire.

## Description

Méthode ; active l'option de sélection ou d'édition d'un élément. En règle générale, vous utilisez cette méthode sur des objets renvoyés par [fl.findObjectInDocByName\(\)](#) ou par [fl.findObjectInDocByType\(\)](#).

## Exemple

L'exemple suivant sélectionne un élément nommé "second text field" si un élément est trouvé dans le document :

```
var nameToSearchFor = "second text field";  
var doc = fl.getDocumentDOM();
```

```
// Commencer par afficher la séquence 1 (valeur d'index 0).  
document.editScene(0);
```

```
// Rechercher l'élément par le nom.
var results = fl.findObjectInDocByName(nameToSearchFor, doc);
if (results.length > 0) {
    // Sélectionner le premier élément trouvé.
    // Transmettre la valeur false pour que l'objet symbolInstance
    // que vous recherchez soit sélectionné.
    // Si vous transmettez la valeur true, l'occurrence du symbole
    // passe en mode d'édition.
    fl.selectElement(results[0], false);
    alert("success, found " + results.length + " objects")
}
else {
    alert("failed, no objects with name " + nameToSearchFor + " found");
}
```

## Voir aussi

[fl.findObjectInDocByName\(\)](#), [fl.findObjectInDocByType\(\)](#)

# fl.selectTool()

## Disponibilité

Flash CS3 Professional.

## Utilisation

```
fl.selectTool(toolName)
```

## Paramètres

*toolName* Chaîne spécifiant le nom de l'outil à sélectionner. Consultez la section « Description » ci-dessous pour connaître les valeurs acceptées par ce paramètre.

## Description

Méthode ; sélectionne l'outil spécifié dans le panneau Outils. Les valeurs possibles par défaut pour *toolName* sont les suivantes : "arrow", "bezierSelect", "freeXform", "fillXform", "lasso", "pen", "penplus", "penminus", "penmodify", "text", "line", "rect", "oval", "rectPrimitive", "ovalPrimitive", "polystar", "pencil", "brush", "inkBottle", "bucket", "eyeDropper", "eraser", "hand" et "magnifier".

Si un utilisateur ou vous-même créez des outils personnalisés, les noms de ces outils peuvent également être transmis comme paramètre *toolName*. La liste des noms d'outils se trouve dans le fichier suivant :

- Windows 2000 ou Windows XP :

*lecteur d'amorçage* \Documents and Settings\*utilisateur* \Local Settings \Application Data \Adobe \Flash CS3 \*langue* \Configuration \Tools \toolConfig.xml

■ Mac OS X :

Disque dur Macintosh/Utilisateurs/*utilisateur*/Library/Application Support/Adobe/  
Flash CS3/*langue*/Configuration/Tools//toolConfig.xml

### Exemple

L'exemple suivant sélectionne l'outil Plume.

```
fl.selectTool("pen");
```

### Voir aussi

[Objet Tools](#), [Objet ToolObj](#)

## fl.setActiveWindow()

### Disponibilité

Flash MX 2004.

### Utilisation

```
fl.setActiveWindow(document [, bActivateFrame])
```

### Paramètres

*document* [Objet Document](#) spécifiant le document à sélectionner comme fenêtre active.

*bActivateFrame* Paramètre facultatif qui est ignoré par Flash et Fireworks, est uniquement destiné à la compatibilité avec Dreamweaver.

### Valeur renvoyée

Aucune.

### Description

Méthode ; définit le document spécifié comme fenêtre active. Cette méthode est également prise en charge par Dreamweaver et Fireworks. S'il existe plusieurs fenêtres ouvertes sur le même document (créées à l'aide de l'option Modifier dans une nouvelle fenêtre), la première fenêtre est sélectionnée.

### Exemple

L'exemple suivant illustre deux façons d'enregistrer un document spécifié :

```
fl.setActiveWindow(fl.documents[0]);

var theIndex = fl.findDocumentIndex("myFile.fla");
fl.setActiveWindow(fl.documents[theIndex]);
```



# fl.showIdleMessage()

## Disponibilité

Flash 8.

## Utilisation

```
fl.showIdleMessage(show)
```

## Paramètres

*show* Valeur booléenne indiquant si le message d'avertissement sur une exécution du script trop longue doit être activé ou désactivé.

## Valeur renvoyée

Aucune.

## Description

Méthode ; permet de désactiver le message avertissant que l'exécution du script est trop longue ; transmettez la valeur `false` pour *show*. Il est conseillé d'utiliser cette méthode pour les opérations de traitement de fichiers par lot qui nécessitent beaucoup de temps. Pour réactiver l'alerte, appelez une nouvelle fois cette méthode en transmettant cette fois la valeur `true` pour *show*.

## Exemple

L'exemple suivant montre comment désactiver puis réactiver le message avertissant que l'exécution du script est trop longue :

```
fl.showIdleMessage(false);  
var result = timeConsumingFunction();  
fl.showIdleMessage(true); ;  
var result = timeConsumingFunction;
```

# fl.synchronizeDocumentWithHeadVersion()

## Disponibilité

Flash CS3 Professional.

## Utilisation

```
fl.showIdleMessage(true)synchronizeDocumentWithHeadVersion(documentObject);
```

## Paramètres

*documentObject* A [Objet Document](#).

### Valeur renvoyée

Valeur booléenne `true` si la synchronisation du fichier spécifié s'est correctement déroulée sur le serveur Version Cue, sinon `false`.

### Description

Méthode ; synchronise le document spécifié avec la version la plus récente trouvée sur le serveur Version Cue et consigne les éventuelles erreurs dans le panneau Sortie. Cette méthode est identique à `document.synchronizeWithHeadVersion()`.

### Exemple

L'exemple suivant enregistre le document actif sur le serveur Version Cue :

```
fl.synchronizeWithHeadVersion(fl.getDocumentDOM());
```

### Voir aussi

`fl.downloadLatestVersion()`, `fl.revertDocumentToLastVersion()`,  
`fl.saveAVersionOfDocument()`

## fl.tools

### Disponibilité

Flash MX 2004.

### Utilisation

```
fl.tools
```

### Description

Propriété en lecture seule ; tableau des objets Tools (consultez la section [Objet Tools](#)). Cette méthode n'est utilisée que lors de la création d'outils extensibles.

## fl.trace()

### Disponibilité

Flash MX 2004.

### Utilisation

```
fl.trace(message)
```

### Paramètres

*message* Chaîne apparaissant dans le panneau Sortie.

## Valeur renvoyée

Aucune.

## Description

Méthode. Envoie une chaîne de texte au panneau Sortie, suivie d'un code de nouvelle ligne, et affiche le panneau Sortie s'il n'est pas déjà visible. Cette méthode est identique à `outputPanel.trace()` et fonctionne comme l'instruction `trace()` en code ActionScript.

Pour envoyer une ligne vide, utilisez `fl.trace("")` ou `fl.trace("\n")`. Vous pouvez utiliser cette dernière commande en ligne si vous intégrez `\n` à la chaîne *message*.

## Exemple

L'exemple suivant affiche plusieurs lignes de texte dans le panneau Sortie :

```
fl.outputPanel.clear();
fl.trace("Hello World!!!");
var myPet = "cat";
fl.trace("\nI have a " + myPet);
fl.trace("");
fl.trace("I love my " + myPet);
fl.trace("Do you have a " + myPet + "?");
```

## fl.version

### Disponibilité

Flash MX 2004.

### Utilisation

```
fl.version
```

### Description

Propriété en lecture seule ; chaîne longue de la version de l'outil de programmation Flash, plate-forme comprise.

### Exemple

L'exemple suivant affiche la version de l'outil de programmation Flash dans le panneau Sortie :

```
alert(fl.version); // Par exemple, WIN 9,0,0,375
```

# fl.xmlui

## Disponibilité

Flash MX 2004.

## Utilisation

`fl.xmlui`

## Description

Propriété en lecture seule ; [Objet XMLUI](#). Cette propriété permet de lire et définir les propriétés de XMLUI dans une boîte de dialogue XMLUI, et d'accepter ou annuler par programmation la boîte de dialogue standard.

## Exemple

Voir [Objet XMLUI](#).

# Objet FLfile

## Disponibilité

Flash MX 2004 7.2.

## Description

L'objet FLfile permet de créer des extensions Flash capables d'accéder aux dossiers et fichiers locaux, de les modifier et de les supprimer. L'API FLfile est proposée sous forme d'extension de l'API JavaScript. Cette extension est appelée *bibliothèque partagée* et se trouve dans le dossier suivant :

- Windows 2000 ou Windows XP :

*lecteur d'amorçage*\Documents and Settings\*utilisateur*\Local Settings\Application Data\Adobe\Flash CS3\*langue*\Configuration\External Libraries\FLfile.dll

- Mac OS X :

Disque dur Macintosh/Utilisateurs/*utilisateur*/Library/Application Support/Adobe/Flash CS3/*langue*/Configuration/External Libraries/FLfile.dll

### REMARQUE

Il ne faut pas confondre les bibliothèques partagées qui contiennent les symboles de vos documents Flash avec les bibliothèques partagées de l'API JavaScript. Ce sont deux types de composants différents.

Les méthodes de FLfile s'appliquent aux fichiers et aux dossiers (répertoires) du disque. Chaque méthode prend donc un ou plusieurs paramètres qui spécifient l'emplacement d'un fichier ou d'un dossier. Cet emplacement de fichier ou de dossier est exprimé sous forme d'une chaîne très semblable à l'adresse URL d'un site Web. Ce type d'adresse, appelé file URI (Uniform Resource Identifier) respecte le format représenté ici (guillemets compris) :

```
"file:///drive|/folder 1/folder 2/.../filename"
```

Par exemple, si vous souhaitez créer sur le lecteur C un dossier nommé « config » et le placer dans le répertoire Program Files/MyApp, utilisez la commande suivante :

```
FLfile.createFolder("file:///C:/Program Files/MyApp/config");
```

Si vous souhaitez ensuite placer dans ce dossier un fichier nommé « config.ini », la commande sera la suivante :

```
FLfile.write("file:///C:/Program Files/MyApp/config/config.ini", "");
```

Pour créer un dossier sous Macintosh, vous pouvez utiliser une commande similaire à celle-ci :

```
FLfile.createFolder("file:///Macintosh/MyApp/config");
```

# Résumé des méthodes de l'objet FLfile

Vous pouvez utiliser les méthodes suivantes avec l'objet FLfile :

Méthode	Description
<code>FLfile.copy()</code>	Copie un fichier.
<code>FLfile.createFolder()</code>	Crée un ou plusieurs dossiers.
<code>FLfile.exists()</code>	Vérifie l'existence d'un fichier ou d'un dossier.
<code>FLfile.getAttributes()</code>	Vérifie les attributs d'un fichier : accessible en écriture, en lecture seule, caché, visible ou dossier système.
<code>FLfile.getCreationDate()</code>	Détermine combien de secondes se sont écoulées entre le 1 janvier 1970 et l'heure à laquelle le fichier ou le dossier a été créé.
<code>FLfile.getCreationDateObj()</code>	Lit la date de création d'un fichier ou d'un dossier.
<code>FLfile.getModificationDate()</code>	Détermine combien de secondes se sont écoulées entre le 1 janvier 1970 et l'heure à laquelle le fichier ou le dossier a été modifié pour la dernière fois.
<code>FLfile.getModificationDateObj()</code>	Lit la date de dernière modification d'un fichier ou d'un dossier.
<code>FLfile.getSize()</code>	Lit la taille d'un fichier.
<code>FLfile.listFolder()</code>	Liste le contenu d'un dossier.
<code>FLfile.read()</code>	Lit le contenu d'un fichier.
<code>FLfile.remove()</code>	Supprime un fichier ou un dossier.
<code>FLfile.setAttributes()</code>	Donne à un fichier ou un dossier l'attribut lecture seule, lecture et écriture, caché ou visible.
<code>FLfile.write()</code>	Crée un fichier, écrit dans un fichier ou ajoute à la fin d'un fichier.

# FLfile.copy()

## Disponibilité

Flash MX 2004 7.2.

## Utilisation

`FLfile.copy(fileURI, copyURI)`

## Paramètres

*fileURI* Chaîne de type « fichier:/// URI » qui indique le fichier à copier.

*copyURI* Chaîne de type « fichier:/// URI » qui spécifie l'emplacement et le nom du fichier copié.

## Valeur renvoyée

Valeur booléenne `true` si l'opération a réussi, ou `false` sinon.

## Description

Méthode ; copie un fichier d'un emplacement à un autre. Cette méthode renvoie `false` si *copyURI* existe déjà.

## Exemple

L'exemple suivant crée une copie de sauvegarde d'un fichier de configuration nommé `config.ini`, et place cette copie dans le même dossier que l'original, avec un nouveau nom :

```
var originalFileURI="file:///C:/Program Files/MyApp/config.ini";  
var newFileURI="file:///C:/Program Files/MyApp/config_backup.ini";  
FLfile.copy(originalFileURI, newFileURI);
```

Il est possible d'exécuter la même tâche avec une seule commande :

```
FLfile.copy("file:///C|:/Program Files/MyApp/config.ini", file:///C|/  
Program Files/MyApp/config_backup.ini");
```

# FLfile.createFolder()

## Disponibilité

Flash MX 2004 7.2.

## Utilisation

`FLfile.createFolder(folderURI)`

## Paramètres

*folderURI* Adresse URI indiquant le chemin d'accès de la structure de dossiers à créer.

## Valeur renvoyée

Valeur booléenne `true` si l'opération a réussi, `false` si `folderURI` existe déjà.

## Description

Méthode ; crée un ou plusieurs dossiers à l'emplacement spécifiée.

Vous pouvez créer plusieurs dossiers en même temps. Par exemple, la commande suivante crée les dossiers `MyData` et `TempData` s'ils n'existent pas :

```
FLfile.createFolder("file:///c:/MyData/TempData")
```

## Exemple

L'exemple suivant crée deux sous-dossiers dans le dossier de configuration ([fl.configURI](#)) :

```
fl.trace(FLfile.createFolder(fl.configURI+"folder01/subfolder01"));
```

L'exemple suivant tente de créer un dossier appelé `tempFolder` à la racine du lecteur C, puis affiche une alerte informant de la réussite ou de l'échec de l'opération.

```
var folderURI = "file:///c:/tempFolder";
if (FLfile.createFolder(folderURI)) {
    alert("Created " + folderURI);
}
else {
    alert(folderURI + " already exists");
}
```

## Voir aussi

[FLfile.remove\(\)](#), [FLfile.write\(\)](#)

# FLfile.exists()

## Disponibilité

Flash MX 2004 7.2.

## Utilisation

```
FLfile.exists(fileURI)
```

## Paramètres

*fileURI* Chaîne de type « fichier:/// URI » qui indique le fichier à vérifier.

## Valeur renvoyée

Valeur booléenne `true` si l'opération a réussi, sinon `false`.



## Description

Méthode ; vérifie si le fichier spécifié existe. Si vous voulez spécifier un dossier dans lequel créer le fichier, ce dossier doit déjà exister. Pour créer des dossiers, utilisez `FLfile.createFolder()`.

## Exemples

L'exemple suivant vérifie l'existence d'un fichier nommé mydata.txt dans le dossier temp, puis affiche une alerte signalant l'existence ou l'absence d'un tel fichier :

```
var fileURI = "file:///c:/temp/mydata.txt";
if (FLfile.exists(fileURI)) {
    alert( fileURI + " exists.");
}
else {
    alert( fileURI + " does not exist.");
}
```

L'exemple suivant vérifie l'existence d'un fichier de configuration requis dans le dossier MyApplication. Si le fichier n'existe pas, il est créé automatiquement.

```
var configFile = "file:///C:/MyApplication/config.ini";
if (!FLfile.exists(configFile)) {
    FLfile.write(configFile, "");
}
```

## Voir aussi

`FLfile.write()`

# FLfile.getAttributes()

## Disponibilité

Flash MX 2004 7.2.

## Utilisation

`FLfile.getAttributes(fileOrFolderURI)`

## Paramètres

*fileOrFolderURI* Chaîne exprimée sous la forme « fichier:/// URI » et indiquant le fichier ou le dossier dont les attributs doivent être lus.

## Valeur renvoyée

Chaîne représentant les attributs du fichier ou dossier spécifié.

### REMARQUE

Les résultats sont impossibles à prédire si le fichier ou le dossier n'existe pas. Il est nécessaire d'appeler `FLfile.exists()` avant d'utiliser cette méthode.

## Description

Méthode ; renvoie une chaîne qui représente les attributs du fichier ou dossier indiqué, ou une chaîne vide si le fichier ne possède pas d'attributs spécifiques (c'est-à-dire s'il n'est pas en lecture seule, caché, etc.). Employez toujours `FLfile.exists()` pour vérifier l'existence du fichier ou du dossier avant d'utiliser cette méthode.

Les caractères de cette chaîne représentent les attributs sous la forme suivante :

- R — *fileOrFolderURI* est en lecture seule
- D — *fileOrFolderURI* est un dossier (répertoire)
- H — *fileOrFolderURI* est caché (Windows uniquement)
- S — *fileOrFolderURI* est un fichier ou dossier système (Windows uniquement)
- A — *fileOrFolderURI* est prêt pour l'archivage (Windows uniquement)

Par exemple, si *fileOrFolderURI* est un dossier caché, la chaîne renvoyée est "DH".

## Exemple

L'exemple suivant lit les attributs du fichier `mydata.txt` et affiche un message d'alerte si ce fichier est en lecture seule.

```
var URI = "file:///c:/temp/mydata.txt";
if (FLfile.exists(URI)){
    var attr = FLfile.getAttributes(URI);
    if (attr && (attr.indexOf("R") != -1)) { // La chaîne renvoyée contient R.
        alert(URI + " is read only!");
    }
}
```

## Voir aussi

[FLfile.setAttributes\(\)](#)

# FLfile.getCreationDate()

## Disponibilité

Flash MX 2004 7.2.

## Utilisation

`FLfile.getCreationDate(fileOrFolderURI)`

## Paramètres

*fileOrFolderURI* Chaîne de type « fichier:/// URI » qui indique le fichier ou le dossier dont la date et l'heure de création doivent être lues, sous forme hexadécimale.

## Valeur renvoyée

Chaîne contenant un nombre hexadécimal qui représente le nombre de secondes écoulées entre le 1 janvier 1970 et l'heure à laquelle le fichier ou le dossier a été créé, ou "00000000" si le fichier ou le dossier n'existe pas.

## Description

Méthode ; détermine combien de secondes se sont écoulées entre le 1 janvier 1970 et l'heure à laquelle le fichier ou le dossier a été créé. Cette méthode est principalement utilisée pour comparer les dates de création ou de modification de fichiers ou de dossiers.

## Exemple

L'exemple suivant détermine si un fichier a été modifié depuis sa création :

```
// Assurez-vous que le fichier indiqué existe.
var fileURI = "file:///C:/MyApplication/MyApp fla";
var creationTime = FLfile.getCreationDate(fileURI);
var modificationTime = FLfile.getModificationDate(fileURI);
if ( modificationTime > creationTime ) {
    alert("The file has been modified since it was created.");
}
else {
    alert("The file has not been modified since it was created.");
}
```

## Voir aussi

[FLfile.getCreationDateObj\(\)](#), [FLfile.getModificationDate\(\)](#)

# FLfile.getCreationDateObj()

## Disponibilité

Flash MX 2004 7.2.

## Utilisation

`FLfile.getCreationDateObj(fileOrFolderURI)`

## Paramètres

*fileOrFolderURI* Chaîne de type « fichier:/// URI » qui indique le fichier ou le dossier dont la date et l'heure de création doivent être lues, sous forme d'un objet Date JavaScript.

## Valeur renvoyée

Objet Date JavaScript qui représente la date et l'heure auxquelles le fichier ou le dossier spécifié a été créé. Si le fichier ou le dossier n'existe pas, l'objet contient des informations indiquant qu'il a été créé à minuit (GMT) le 31 décembre 1969.

## Description

Méthode ; renvoie un objet Date JavaScript qui représente la date et l'heure auxquelles le fichier ou le dossier spécifié a été créé.

## Exemple

L'exemple suivant affiche dans le panneau Sortie, dans un format lisible par l'opérateur, la date à laquelle un fichier a été créé :

```
// Assurez-vous que le fichier indiqué existe.  
var file1Date = FLfile.getCreationDateObj("file:///c:/temp/file1.txt");  
fl.trace(file1Date);
```

## Voir aussi

[FLfile.getCreationDate\(\)](#), [FLfile.getModificationDateObj\(\)](#)

# FLfile.getModificationDate()

## Disponibilité

Flash MX 2004 7.2.

## Utilisation

`FLfile.getModificationDate(fileOrFolderURI)`

## Paramètres

*fileOrFolderURI* Chaîne de type « fichier:/// URI » qui indique le fichier dont la date de modification doit être lue, sous forme hexadécimale.

## Valeur renvoyée

Chaîne contenant un nombre hexadécimal qui représente le nombre de secondes écoulées entre le 1 janvier 1970 et l'heure à laquelle le fichier ou le dossier a été modifié pour la dernière fois, ou "00000000" si le fichier ou le dossier n'existe pas.

## Description

Méthode ; détermine combien de secondes se sont écoulées entre le 1 janvier 1970 et l'heure à laquelle le fichier ou le dossier a été modifié pour la dernière fois. Cette méthode est principalement utilisée pour comparer les dates de création ou de modification de fichiers ou de dossiers.

## Exemple

L'exemple suivant compare les dates de modification de deux fichiers et détermine lequel des deux a été modifié le plus récemment :

```
// Assurez-vous que le fichier indiqué existe.
file1 = "file:///C:/MyApplication/MyApp.flc"
file2 = "file:///C:/MyApplication/MyApp.as"
modificationTime1 = FLfile.getModificationDate(file1)
modificationTime2 = FLfile.getModificationDate(file2)

if(modificationTime1 > modificationTime2) {
    alert("File 2 is older than File 1")
}
else if(modificationTime1 < modificationTime2) {
    alert("File 1 is older than File 2")
}
else {
    alert("File 1 and File 2 were saved at the same time")
}
```

## Voir aussi

[FLfile.getCreationDate\(\)](#), [FLfile.getModificationDateObj\(\)](#)

# FLfile.getModificationDateObj()

## Disponibilité

Flash MX 2004 7.2.

## Utilisation

`FLfile.getModificationDateObj(fileOrFolderURI)`

## Paramètres

*fileOrFolderURI* Chaîne de type « fichier:/// URI » qui indique le fichier ou le dossier dont la date et l'heure de modification doivent être lues, sous forme d'un objet Date JavaScript.

## Valeur renvoyée

Objet Date JavaScript qui représente la date et l'heure auxquelles le fichier ou le dossier spécifié a été modifié pour la dernière fois. Si le fichier ou le dossier n'existe pas, l'objet contient des informations indiquant qu'il a été créé à minuit (GMT) le 31 décembre 1969.

## Description

Méthode ; renvoie un objet Date JavaScript qui représente la date et l'heure auxquelles le fichier ou le dossier spécifié a été modifié pour la dernière fois.

## Exemple

L'exemple suivant affiche dans le panneau Sortie, dans un format lisible par l'opérateur, la date à laquelle un fichier a été modifié pour la dernière fois :

```
// Assurez-vous que le fichier indiqué existe.  
var file1Date = FLfile.getModificationDateObj("file:///c:/temp/file1.txt");  
trace(file1Date);
```

## Voir aussi

[FLfile.getCreationDateObj\(\)](#), [FLfile.getModificationDate\(\)](#)

# FLfile.getSize()

## Disponibilité

Flash MX 2004 7.2.

## Utilisation

```
FLfile.getSize(fileURI)
```

## Paramètres

*fileURI* Chaîne de type « fichier:/// URI » qui indique le fichier dont la taille doit être lue.

## Valeur renvoyée

Un entier qui représente la taille en octets du fichier spécifié, ou 0 si ce fichier n'existe pas.

## Description

Méthode ; renvoie un entier qui représente la taille en octets du fichier spécifié, ou 0 si ce fichier n'existe pas. Si la valeur renvoyée est 0, vous pouvez utiliser [FLfile.exists\(\)](#) pour déterminer si le fichier est un fichier de zéro octets ou s'il n'existe pas.

Cette méthode renvoie des valeurs de taille de fichier correctes uniquement pour les fichiers n'excédant pas 2 Go.

## Exemple

L'exemple suivant stocke la taille du fichier mydata.txt dans la variable `fileSize` :

```
var URL = "file:///c:/temp/mydata.txt";  
var fileSize = FLfile.getSize(URL);
```

# FLfile.listFolder()

## Disponibilité

Flash MX 2004 7.2.

## Utilisation

```
FLfile.listFolder(folderURI [, filesOrDirectories])
```

## Paramètres

*folderURI* Chaîne de type « fichier:/// URI » qui indique le dossier dont le contenu doit être lu. Il est possible d'insérer des caractères génériques dans *folderURI*. Les caractères génériques utilisables sont \* (un ou plusieurs caractères) et ? (un caractère).

*filesOrDirectories* Chaîne facultative qui spécifie si seuls doivent être renvoyés les noms des fichiers ou les noms des dossiers (répertoires). Si ce paramètre est omis, les noms des dossiers et des fichiers sont renvoyés. Les valeurs gérées sont "files" et "directories".

## Valeur renvoyée

Un tableau de chaînes qui représente le contenu du dossier, ou *false* si ce dossier n'existe pas.

## Description

Méthode ; renvoie un tableau de chaînes qui représente le contenu du dossier, ou un tableau vide si ce dossier n'existe pas.

## Exemples

L'exemple suivant renvoie un tableau représentant les fichiers, les dossiers, ou les deux types d'éléments à la fois dans le répertoire Program Files :

```
var folderURI = "file:///C:/WINDOWS/Program Files" ;
var fileList = FLfile.listFolder(folderURI, "files") // fichiers
var fileList = FLfile.listFolder(folderURI, "directories") // dossiers
var fileList = FLfile.listFolder(folderURI) // fichiers et dossiers
```

L'exemple suivant renvoie un tableau contenant tous les fichiers texte (.txt) du répertoire «temp» et affiche cette liste dans un message d'alerte :

```
var folderURI = "file:///c:/temp";
var fileMask = "*.txt";
var list = FLfile.listFolder(folderURI + "/" + fileMask, "files");
if (list) {
    alert(folderURI + " contains: " + list.join(" "));
}
```

L'exemple suivant utilise un masque de nom de fichiers dans le dossier *folderURI* spécifié afin de renvoyer les noms de tous les fichiers exécutables du dossier Windows :

```
var executables = FLfile.listFolder("file:///C:/WINDOWS/*.exe", "files")
alert(executables.join("\n"))
```



# FLfile.read()

## Disponibilité

Flash MX 2004 7.2.

## Utilisation

FLfile.read()

## Paramètres

*fileOrFolderURI* Chaîne de type « fichier:/// URI » qui indique le fichier ou le dossier dont les attributs doivent être lus.

## Valeur renvoyée

Le contenu du fichier spécifié sous forme de chaîne, ou `null` si la lecture échoue.

## Description

Méthode ; renvoie le contenu du fichier spécifié sous forme de chaîne, ou `null` si la lecture échoue.

## Exemples

L'exemple suivant lit le fichier `mydata.txt` et, si la lecture réussit, affiche le contenu de ce fichier dans un message d'alerte.

```
var fileURI = "file:///c:/temp/mydata.txt";
var str = FLfile.read( fileURI);
if (str) {
    alert( fileURL + " contains: " + str);
}
```

L'exemple suivant lit le code ActionScript d'un fichier de classe et le stocke dans la variable `code` :

```
var classFileURI = "file:///C:/MyApplication/TextCarousel.as";
var code = FLfile.read(classFileURI);
```

# FLfile.remove()

## Disponibilité

Flash MX 2004 7.2.

## Utilisation

FLfile.remove(*fileOrFolderURI*)

## Paramètres

*fileOrFolderURI* Chaîne de type « fichier:/// URI » qui indique le fichier ou le dossier à supprimer.

## Valeur renvoyée

Valeur booléenne `true` si l'opération a réussi, sinon `false`.

## Description

Méthode ; supprime le fichier ou dossier spécifié. Si le dossier contient des fichiers, ceux-ci sont également supprimés. Il est impossible de supprimer les fichiers ayant l'attribut R (lecture seule).

## Exemples

L'exemple suivant prévient l'utilisateur de l'existence d'un fichier, puis assure la suppression de celui-ci si l'utilisateur choisit cette option :

```
var fileURI = prompt ("Enter file/folder to be deleted: ", "file:///c:/temp/delete.txt");
if (FLfile.exists(fileURI)) {
    var confirm = prompt("File exists. Delete it? (y/n)", "y");
    if (confirm == "y" || confirm == "Y") {
        if(FLfile.remove(fileURI)) {
            alert(fileURI + " is deleted.");
        }
        else {
            alert("fail to delete " + fileURI);
        }
    }
}
else {
    alert(fileURI + " does not exist");
}
```

L'exemple suivant supprime un fichier de configuration créé par une application :

```
if(FLfile.remove("file:///C:/MyApplication/config.ini")) {
    alert("Configuration file deleted");
}
```

L'exemple suivant supprime le dossier Configuration et tout son contenu :

```
FLfile.remove("file:///C:/MyApplication/Configuration/");
```

## Voir aussi

[FLfile.createFolder\(\)](#), [FLfile.getAttributes\(\)](#)

# FLfile.setAttributes()

## Disponibilité

Flash MX 2004 7.2.

## Utilisation

```
FLfile.setAttributes(fileURI, strAttrs)
```

## Paramètres

*fileURI* Chaîne exprimée sous la forme « fichier:/// URI » et indiquant le fichier dont les attributs doivent être définis.

*strAttrs* Chaîne indiquant les attributs à définir. La liste des valeurs gérées par *strAttrs* figure dans la description.

## Valeur renvoyée

Valeur booléenne `true` si l'opération a réussi.

REMARQUE

Les résultats sont imprévisibles si le fichier ou le dossier n'existe pas. Il est nécessaire d'appeler `FLfile.exists()` avant d'utiliser cette méthode.

## Description

Méthode ; spécifie les attributs système du fichier spécifié.

Les valeurs suivantes sont possibles pour *strAttrs* :

- N – Aucun attribut spécifique (ni lecture seule, ni caché, etc.)
- A – Prêt pour l'archivage (Windows uniquement)
- R – Lecture seule (sur Macintosh, « Lecture seule » signifie verrouillé)
- W – Accessible en écriture (annule R)
- H – Caché (Windows uniquement)
- V – Visible (annule H, Windows uniquement)

Si vous insérez à la fois R et W dans *strAttrs*, le R est ignoré et le fichier est paramétré comme étant accessible en écriture. De même, si vous insérez H et V, le H est ignoré et le fichier est paramétré comme étant visible.

Si vous voulez vous assurer que l'attribut d'archivage n'est pas défini, utilisez cette commande avec le paramètre N avant de définir les attributs. En d'autres termes, le paramètre A ne possède pas de pendant qui annule l'attribut Archive.

## Exemples

L'exemple suivant met le fichier mydata.txt en lecture seule et caché. Il n'a aucun effet sur l'attribut d'archivage.

```
var URI = "file:///c:/temp/mydata.txt";
if (FLfile.exists(URI)) {
    FLfile.setAttributes(URI, "RH");
}
```

L'exemple suivant met le fichier mydata.txt en lecture seule et caché. Il fait aussi en sorte que l'attribut d'archivage ne soit pas fixé.

```
var URI = "file:///c:/temp/mydata.txt";

if (FLfile.exists(URI)) {
    FLfile.setAttributes(URI, "N");
    FLfile.setAttributes(URI, "RH");
}
```

## Voir aussi

[FLfile.getAttributes\(\)](#)

# FLfile.write()

## Disponibilité

Flash MX 2004 7.2.

## Utilisation

```
FLfile.write(fileURI, textToWrite, [ strAppendMode ])
```

## Paramètres

*fileURI* Chaîne de type « fichier:/// URI » qui indique le fichier dans lequel une chaîne doit être écrite.

*textToWrite* Chaîne représentant le texte à placer dans le fichier.

*strAppendMode* Chaîne facultative ayant la valeur « append ». Elle indique que la chaîne *textToWrite* doit être écrite à la fin du fichier existant. Si ce paramètre est omis, le contenu actuel de *fileURI* est remplacé par le contenu de *textToWrite*.

## Valeur renvoyée

Valeur booléenne *true* si l'opération a réussi, ou *false* sinon.

## Description

Méthode ; écrit la chaîne spécifiée dans le fichier spécifié (en UTF-8). Si le fichier n'existe pas, il est créé automatiquement. Toutefois, le dossier dans lequel vous placez le fichier doit exister avant que vous utilisiez cette méthode. Pour créer des dossiers, utilisez

`FLfile.createFolder()`.

## Exemple

L'exemple suivant tente d'écrire la chaîne « xxx » dans le fichier mydata.txt, et affiche un message d'alerte si l'écriture a réussi. Il tente ensuite d'écrire la chaîne « aaa » dans le même fichier et affiche un second message d'alerte si l'écriture a réussi. Après l'exécution de ce script, le fichier mydata.txt ne contiendra que le texte « xxxaaa ».

```
var URI = "file:///c:/temp/mydata.txt";
if (FLfile.write(URI, "xxx")) {
    alert("Wrote xxx to " + URI);
}
if (FLfile.write(URI, "aaa", "append")) {
    alert("Appended aaa to " + fileURI);
}
```

## Voir aussi

`FLfile.createFolder()`, `FLfile.exists()`

# Objet folderItem

**Héritage** [Objet Item](#) > objet folderItem

## Disponibilité

Flash MX 2004.

## Description

L'objet folderItem est une sous-classe de l'objet Item. Il n'existe pas de méthodes ou propriétés particulières pour l'objet folderItem. Voir [Objet Item](#).

# Objet fontItem

**Héritage** [Objet Item](#) > objet fontItem

## Disponibilité

Flash MX 2004.

## Description

L'objet fontItem est une sous-classe de l'objet Item. Il n'existe pas de méthodes ou propriétés particulières pour l'objet fontItem. Voir [Objet Item](#).

# Objet Frame

## Disponibilité

Flash MX 2004.

## Description

L'objet Frame représente les images dans le calque.

## Résumé des méthodes de l'objet Frame

Vous pouvez utiliser les méthodes suivantes avec l'objet Frame :

Méthode	Description
<code>frame.getCustomEase()</code>	Renvoie un tableau d'objets JavaScript possédant tous une propriété x et y.
<code>frame.setCustomEase()</code>	Spécifie une courbe de Bézier cubique à utiliser comme courbe d'accélération personnalisée.

## Résumé des propriétés de l'objet Frame

Vous pouvez utiliser les propriétés suivantes avec l'objet Frame :

Propriété	Description
<code>frame.actionScript</code>	Chaîne représentant du code ActionScript.
<code>frame.duration</code>	En lecture seule ; entier indiquant le nombre d'images dans une séquence d'images.
<code>frame.elements</code>	Propriété en lecture seule ; tableau d'objets Element (voir <a href="#">Objet Element</a> ).
<code>frame.hasCustomEase</code>	Valeur booléenne qui spécifie si l'image extrait ses informations d'accélération de la courbe d'accélération personnalisée.
<code>frame.labelType</code>	Chaîne qui spécifie le type de nom d'image.
<code>frame.motionTweenOrientToPath</code>	Valeur booléenne qui spécifie si l'élément interpolé doit subir une rotation pendant son déplacement le long d'un trajet afin de préserver son angle relativement à chaque point du trajet.
<code>frame.motionTweenRotate</code>	Chaîne qui spécifie le type de rotation de l'élément interpolé.



Propriété	Description
<code>frame.motionTweenRotateTimes</code>	Entier qui spécifie le nombre de rotations de l'élément interpolé entre l'image-clé de départ et la suivante.
<code>frame.motionTweenScale</code>	Valeur booléenne ; spécifie si l'élément interpolé doit subir une mise à l'échelle à la taille de l'objet de l'image-clé suivante, en augmentant ainsi sa taille à chaque image de l'interpolation ( <code>true</code> ), ou s'il ne doit pas subir de mise à l'échelle ( <code>false</code> ).
<code>frame.motionTweenSnap</code>	Valeur booléenne qui spécifie si l'élément interpolé doit être automatiquement « accroché » au point le plus proche du calque de guide de mouvement associé au calque de cette image ( <code>true</code> ) ou non ( <code>false</code> ).
<code>frame.motionTweenSync</code>	Valeur booléenne qui indique, lorsqu'elle est <code>true</code> , que l'animation de l'objet interpolé doit être synchronisée avec le scénario principal.
<code>frame.name</code>	Chaîne qui spécifie le nom de l'image.
<code>frame.shapeTweenBlend</code>	Chaîne qui spécifie comment l'interpolation d'une forme passe en fondu entre cette forme dans l'image-clé de départ de l'interpolation et la même forme dans l'image-clé suivante.
<code>frame.soundEffect</code>	Chaîne qui indique les effets d'un son associé directement à une image ( <code>frame.soundLibraryItem</code> ).
<code>frame.soundLibraryItem</code>	Élément de bibliothèque (voir <a href="#">Objet SoundItem</a> ) utilisé pour créer un son.
<code>frame.soundLoop</code>	Entier qui indique le nombre de fois qu'un son associé directement à une image ( <code>frame.soundLibraryItem</code> ) doit être exécuté.
<code>frame.soundLoopMode</code>	Chaîne qui indique si un son associé directement à une image ( <code>frame.soundLibraryItem</code> ) doit être exécuté un certain nombre de fois ou être lu en boucle.
<code>frame.soundName</code>	Chaîne qui indique le nom d'un son associé directement à une image ( <code>frame.soundLibraryItem</code> ), tel que ce son est enregistré dans la bibliothèque.
<code>frame.soundSync</code>	Chaîne indiquant le mode de synchronisation d'un son associé directement à une image ( <code>frame.soundLibraryItem</code> ).
<code>frame.startFrame</code>	Propriété en lecture seule ; index de la première image d'une séquence.
<code>frame.tweenEasing</code>	Entier qui spécifie la quantité d'accélération à appliquer à l'objet interpolé.

Propriété	Description
<code>frame.tweenType</code>	Chaîne qui spécifie le type d'interpolation.
<code>frame.useSingleEaseCurve</code>	Valeur booléenne qui spécifie si une même courbe d'accélération personnalisée est utilisée pour les informations d'accélération de toutes les propriétés.

## frame.actionScript

### Disponibilité

Flash MX 2004.

### Utilisation

`frame.actionScript`

### Description

Propriété ; chaîne représentant du code ActionScript. Pour insérer un caractère de nouvelle ligne, utilisez `"\n"`.

### Exemple

L'exemple suivant affecte le code `stop()` à l'action du calque supérieur de la première image :

```
fl.getDocumentDOM().getTimeline().layers[0].frames[0].actionScript =
    'stop();';
```

## frame.duration

### Disponibilité

Flash MX 2004.

### Utilisation

`frame.duration`

### Description

Propriété en lecture seule ; entier indiquant le nombre d'images dans une séquence d'images.

### Exemple

L'exemple suivant enregistre dans la variable `frameSpan` le nombre d'images d'une séquence qui débute à la première image du calque supérieur :

```
var frameSpan =
    fl.getDocumentDOM().getTimeline().layers[0].frames[0].duration;
```

# frame.elements

## Disponibilité

Flash MX 2004.

## Utilisation

`frame.elements`

## Description

Propriété en lecture seule ; tableau d'objets `Element` (voir [Objet Element](#)). L'ordre des éléments est celui dans lequel ils apparaissent dans le fichier FLA. Si la scène comporte plusieurs formes, et si toutes ces formes sont dissociées, Flash les traite comme un seul élément. Si chaque forme fait partie d'un groupe (si bien qu'il existe plusieurs groupes dans la scène), Flash les traite comme des éléments séparés. Autrement dit, Flash traite les formes non groupées comme un même élément, quel que soit le nombre de formes individuelles figurant sur la scène. Par exemple, si une image contient trois formes de base (non groupées), la propriété `elements.length` de cette image renvoie la valeur 1. Si nécessaire, vous pouvez sélectionner individuellement toutes les formes et les grouper.

## Exemple

L'exemple suivant enregistre dans la variable `myElements` un tableau des éléments figurant dans le calque supérieur :

```
var myElements =  
    fl.getDocumentDOM().getTimeline().layers[0].frames[0].elements;
```

# frame.getCustomEase()

## Disponibilité

Flash 8.

## Utilisation

`Frame.getCustomEase([property])`

## Paramètres

*property* Chaîne facultative qui spécifie la propriété dont vous voulez lire la valeur d'accélération. Les valeurs possibles sont les suivantes : "all" (toutes), "position", "rotation", "scale" (échelle), "color" et "filters". La valeur par défaut est "all".

## Valeur renvoyée

Renvoie un tableau d'objets JavaScript possédant tous une propriété *x* et *y*.

## Description

Méthode ; renvoie un tableau d'objets qui représentent les points de contrôle de la courbe de Bézier cubique qui définit la courbe d'accélération.

## Exemple

L'exemple suivant renvoie la valeur d'accélération personnalisée de la propriété `position` de la première image du calque supérieur :

```
var theFrame = fl.getDocumentDOM().getTimeline().layers[0].frames[0]
var easeArray = theFrame.getCustomEase("position");
```

## Voir aussi

[frame.hasCustomEase](#), [frame.setCustomEase\(\)](#), [frame.useSingleEaseCurve](#)

# frame.hasCustomEase

## Disponibilité

Flash 8.

## Utilisation

`frame.hasCustomEase`

## Description

Propriété ; valeur booléenne. Si elle est `true`, l'image reçoit ses informations d'accélération de la courbe d'accélération personnalisée. Si elle est `false`, l'image reçoit ses informations d'accélération de la valeur d'accélération.

## Exemple

L'exemple suivant spécifie que la première image du calque supérieur doit recevoir ses informations d'accélération de la valeur d'accélération, plutôt que de la courbe d'accélération personnalisée :

```
var theFrame = fl.getDocumentDOM().getTimeline().layers[0].frames[0]
theFrame.hasCustomEase = false;
```

## Voir aussi

[frame.getCustomEase\(\)](#), [frame.setCustomEase\(\)](#), [frame.useSingleEaseCurve](#)

# frame.labelType

## Disponibilité

Flash MX 2004.

## Utilisation

`frame.labelType`

## Description

Propriété ; chaîne qui spécifie le type de nom d'image. Les valeurs gérées sont "none" (aucun), "name" (nom), "comment" (commentaire) ou "anchor" (ancre). Si cette propriété reçoit la valeur "none", la propriété `frame.name` contient une chaîne vide.

## Exemple

L'exemple suivant définit comme "First Frame" le nom de la première image du calque supérieur, puis comme "comment" son étiquette :

```
fl.getDocumentDOM().getTimeline().layers[0].frames[0].name = 'First Frame';  
fl.getDocumentDOM().getTimeline().layers[0].frames[0].labelType =  
    'comment';
```

# frame.motionTweenOrientToPath

## Disponibilité

Flash MX 2004.

## Utilisation

`frame.motionTweenOrientToPath`

## Description

Propriété ; valeur booléenne qui spécifie si l'élément interpolé doit subir une rotation pendant son déplacement le long d'un trajet afin de préserver son angle relativement à chaque point du trajet (`true`), ou s'il ne doit pas subir de rotation (`false`).

Si vous avez l'intention de spécifier une valeur pour cette propriété, donnez la valeur "none" à `frame.motionTweenRotate`.

# frame.motionTweenRotate

## Disponibilité

Flash MX 2004.

## Utilisation

`frame.motionTweenRotate`

## Description

Propriété ; chaîne qui spécifie le type de rotation de l'élément interpolé. Les valeurs gérées sont les suivantes "none" (aucune), "auto", "clockwise" (sens horaire) et "counter-clockwise" (sens anti-horaire). La valeur "auto" signifie que l'objet subira une rotation dans la direction qui nécessite le moins de mouvement pour correspondre à la rotation de cet objet dans l'image-clé suivante.

Si vous avez l'intention de spécifier une valeur pour la propriété [frame.motionTweenOrientToPath](#), donnez la valeur "none" à cette propriété.

## Exemple

Voir [frame.motionTweenRotateTimes](#).

# frame.motionTweenRotateTimes

## Disponibilité

Flash MX 2004.

## Utilisation

`frame.motionTweenRotateTimes`

## Description

Propriété ; entier qui spécifie le nombre de rotations de l'élément interpolé entre l'image-clé de départ et la suivante.

## Exemple

L'exemple suivant fait effectuer à l'élément qui se trouve dans l'image active trois rotations dans le sens anti-horaire dans le délai nécessaire pour arriver à l'image-clé suivante :

```
fl.getDocumentDOM().getTimeline().layers[0].frames[0].motionTweenRotate =  
    "counter-clockwise";  
fl.getDocumentDOM().getTimeline().layers[0].frames[0].motionTweenRotateTimes  
    = 3;
```

# frame.motionTweenScale

## Disponibilité

Flash MX 2004.

## Utilisation

`frame.motionTweenScale`

## Description

Propriété ; valeur booléenne qui spécifie si l'élément interpolé doit subir une mise à l'échelle à la taille de l'objet de l'image-clé suivante, en augmentant ainsi sa taille à chaque image de l'interpolation (`true`), ou s'il ne doit pas subir de mise à l'échelle (`false`).

## Exemple

L'exemple suivant spécifie que l'élément interpolé doit subir une mise à l'échelle à la taille de l'objet de l'image-clé suivante, en augmentant ainsi sa taille à chaque image de l'interpolation.

```
fl.getDocumentDOM().getTimeline().layers[0].frames[0].motionTweenScale =  
    true;
```

# frame.motionTweenSnap

## Disponibilité

Flash MX 2004.

## Utilisation

`frame.motionTweenSnap`

## Description

Propriété ; valeur booléenne qui spécifie si l'élément interpolé doit être automatiquement « accroché » au point le plus proche du calque de guide de mouvement associé au calque de cette image (`true`) ou non (`false`).

# frame.motionTweenSync

## Disponibilité

Flash MX 2004.

## Utilisation

`frame.motionTweenSync`

## Description

Propriété ; valeur booléenne qui indique, lorsqu'elle est `true`, que l'animation de l'objet interpolé doit être synchronisée avec le scénario principal.

## Exemple

L'exemple suivant indique que l'objet interpolé doit être synchronisé avec le scénario :

```
fl.getDocumentDOM().getTimeline().layers[0].frames[0].motionTweenSync =  
    true;
```

# frame.name

## Disponibilité

Flash MX 2004.

## Utilisation

`frame.name`

## Description

Propriété ; chaîne qui spécifie le nom de l'image.

## Exemple

L'exemple suivant définit comme "First Frame" le nom de la première image du calque supérieur, puis enregistre dans la variable `frameLabel` la valeur de sa propriété `name` :

```
fl.getDocumentDOM().getTimeline().layers[0].frames[0].name = 'First Frame';  
var frameLabel =  
    fl.getDocumentDOM().getTimeline().layers[0].frames[0].name;
```

# frame.setCustomEase()

## Disponibilité

Flash 8.

## Utilisation

`frame.setCustomEase(property, easeCurve)`

## Paramètres

*property* Chaîne qui spécifie la propriété pour laquelle la courbe d'accélération doit être utilisée. Les valeurs gérées sont les suivantes : "all" (toutes), "position", "rotation", "scale" (échelle), "color" et "filters".

*easeCurve* Tableau d'objets qui définissent la courbe d'accélération. Chaque élément de ce tableau doit être un objet JavaScript avec des propriétés *x* et *y*.

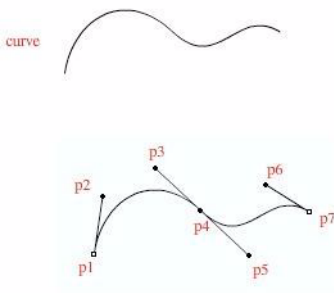


## Valeur renvoyée

Aucune.

## Description

Méthode ; spécifie un tableau de coordonnées de points de contrôle et de points de fin de tangente qui décrivent une courbe de Bézier cubique à utiliser comme courbe d'accélération personnalisée. Ce tableau est basé sur la position horizontale (ordinal : de gauche à droite) des points de contrôle et des points de fin de tangente. Par exemple, la figure ci-dessous illustre une courbe accélération générée si le tableau *easeCurve* contient des valeurs pour les sept points (*p1* à *p7*) :



## Exemple

L'exemple suivant règle la courbe d'accélération de toutes les propriétés de la première image sur le premier calque sur la courbe de Bézier spécifiée par les points de contrôle et les points de fin de tangente stockés dans le tableau *myCurve* :

```
var theFrame = fl.getDocumentDOM().getTimeline().layers[0].frames[0]
var myCurve = [{x:100, y:200},{x:200, y:100}, {x:10, y:0}]
theFrame.setCustomEase("all", myCurve);
```

## Voir aussi

[frame.getCustomEase\(\)](#), [frame.hasCustomEase](#), [frame.useSingleEaseCurve](#)

# frame.shapeTweenBlend

## Disponibilité

Flash MX 2004.

## Utilisation

`frame.shapeTweenBlend`

## Description

Propriété ; chaîne qui spécifie comment l'interpolation d'une forme passe en fondu entre cette forme dans l'image-clé de départ de l'interpolation et la même forme dans l'image-clé suivante. Les valeurs gérées sont "distributive" (distributif) ou "angular" (angulaire).

# frame.soundEffect

## Disponibilité

Flash MX 2004.

## Utilisation

`frame.soundEffect`

## Description

Propriété ; chaîne indiquant les effets d'un son associé directement à une image ([frame.soundLibraryItem](#)). Les valeurs gérées sont les suivantes : "none" (aucune), "left channel" (canal gauche), "right channel" (canal droit), "fade left to right" (fondu de gauche à droite), "fade right to left" (fondu de droite à gauche), "fade in" (fondu à l'ouverture), "fade out" (fondu à la fermeture) et "custom" (personnalisé).

## Exemple

L'exemple suivant spécifie que le son associé à la première image doit apparaître en fondu à l'ouverture :

```
fl.getDocumentDOM().getTimeline().layers[0].frames[0].soundEffect = "fade in";
```

# frame.soundLibraryItem

## Disponibilité

Flash MX 2004.

## Utilisation

frame.soundLibraryItem

## Description

Propriété ; élément de bibliothèque (consultez la section [Objet SoundItem](#)) utilisé pour créer un son. Le son est associé directement à l'image.

## Exemple

L'exemple suivant affecte le premier élément de la bibliothèque à la propriété soundLibraryItem de la première image :

```
// Le premier élément de la bibliothèque doit être un objet Sound.  
fl.getDocumentDOM().getTimeline().layers[0].frames[0].soundLibraryItem  
=fl.getDocumentDOM().library.items[0];
```

# frame.soundLoop

## Disponibilité

Flash MX 2004.

## Utilisation

frame.soundLoop

## Description

Propriété ; entier indiquant le nombre de fois qu'un son associé directement à une image ([frame.soundLibraryItem](#)) doit être exécuté. Si vous avez l'intention de spécifier une valeur pour cette propriété, donnez la valeur "repeat" à la propriété [frame.soundLoopMode](#).

## Exemple

Voir [frame.soundLoopMode](#).

# frame.soundLoopMode

## Disponibilité

Flash MX 2004.

## Utilisation

`frame.soundLoopMode`

## Description

Propriété ; chaîne indiquant si un son associé directement à une image ([frame.soundLibraryItem](#)) doit être exécuté un certain nombre de fois ou être lu en boucle. Les valeurs gérées sont "repeat" (répétition) ou "loop" (en boucle). Pour indiquer le nombre d'exécutions du son, donnez une valeur à la propriété [frame.soundLoop](#).

## Exemple

L'exemple suivant indique qu'un son doit être exécuté deux fois :

```
fl.getDocumentDOM().getTimeline().layers[0].frames[0].soundLoopMode =  
    "repeat";  
fl.getDocumentDOM().getTimeline().layers[0].frames[0].soundLoop = 2;
```

# frame.soundName

## Disponibilité

Flash MX 2004.

## Utilisation

`frame.soundName`

## Description

Propriété ; chaîne indiquant le nom d'un son associé directement à une image ([frame.soundLibraryItem](#)), tel qu'il est enregistré dans la bibliothèque.

## Exemple

L'exemple suivant donne à la propriété `soundName` de la première image la valeur "song1.mp3" ; le fichier `song1.mp3` doit exister dans la bibliothèque :

```
fl.getDocumentDOM().getTimeline().layers[0].frames[0].soundName =  
    "song1.mp3";
```

# frame.soundSync

## Disponibilité

Flash MX 2004.

## Utilisation

`frame.soundSync`

## Description

Propriété ; chaîne indiquant le mode de synchronisation d'un son associé directement à une image (`frame.soundLibraryItem`). Les valeurs gérées sont les suivantes "event" (événement), "stop", "start" et "stream" (flux).

## Exemple

L'exemple suivant indique qu'un son doit être exécuté en flux :

```
fl.getDocumentDOM().getTimeline().layers[0].frames[0].soundSync = 'stream';
```

# frame.startFrame

## Disponibilité

Flash MX 2004.

## Utilisation

`frame.startFrame`

## Description

Propriété en lecture seule ; index de la première image d'une séquence.

## Exemple

Dans l'exemple suivant, `stFrame` est l'index de la première image de la séquence. Dans cet exemple, une séquence d'images s'étend sur six images, de l'image 5 à l'image 10. La valeur de `stFrame` pour n'importe quelle image comprise entre l'image 5 et l'image 10 est donc 4 (ne pas oublier que les valeurs d'index sont différentes des valeurs de numéro d'image).

```
var stFrame =  
    fl.getDocumentDOM().getTimeline().layers[0].frames[4].startFrame;  
fl.trace(stFrame); // 4  
var stFrame =  
    fl.getDocumentDOM().getTimeline().layers[0].frames[9].startFrame;  
fl.trace(stFrame); // 4
```

# frame.tweenEasing

## Disponibilité

Flash MX 2004.

## Utilisation

`frame.tweenEasing`

## Description

Propriété ; entier qui spécifie la quantité d'accélération à appliquer à l'objet interpolé. Les valeurs gérées vont de -100 à 100. Pour débiter en douceur l'interpolation et l'accélérer en approchant de la fin de l'animation, utilisez une valeur comprise entre -1 et -100. Pour débiter rapidement l'interpolation et la ralentir en approchant de la fin de l'animation, utilisez une valeur comprise entre 1 et 100.

## Exemple

L'exemple suivant indique que le mouvement de l'objet en cours d'interpolation doit débiter rapidement, puis décélérer vers la fin de l'animation :

```
fl.getDocumentDOM().getTimeline().layers[0].frames[0].tweenEasing = 50;
```

# frame.tweenType

## Disponibilité

Flash MX 2004.

## Utilisation

`frame.tweenType`

## Description

Propriété ; chaîne qui spécifie le type d'interpolation. Les valeurs gérées sont les suivantes : "motion" (mouvement), "shape" (forme) ou "none" (aucune). La valeur "none" supprime l'interpolation de mouvement. Utilisez la méthode `timeline.createMotionTween()` pour créer une interpolation de mouvement.

Si vous spécifiez la valeur "motion", l'objet à interpoler doit être un symbole, un champ texte ou un groupe d'objets. L'interpolation sera effectuée entre son emplacement dans l'image-clé actuelle et son emplacement dans l'image-clé suivante.

Si vous affectez la valeur "shape", l'objet à interpoler doit être une forme. L'interpolation sera effectuée entre la forme dans l'image-clé actuelle et la forme dans l'image-clé suivante.

## Exemple

L'exemple suivant spécifie que l'objet est une interpolation de mouvement. L'interpolation doit donc être effectuée entre son emplacement dans l'image-clé actuelle et son emplacement dans l'image-clé suivante :

```
fl.getDocumentDOM().getTimeline().layers[0].frames[0].tweenType = "motion";
```

# frame.useSingleEaseCurve

## Disponibilité

Flash 8.

## Utilisation

```
frame.useSingleEaseCurve
```

## Description

Propriété ; valeur booléenne. Si sa valeur est `true`, la même courbe d'accélération personnalisée est utilisée pour les informations d'accélération de toutes les propriétés.

Si elle est `false`, chaque propriété possède sa propre courbe d'accélération.

Cette propriété n'est pas prise en compte si l'accélération personnalisée n'est pas appliquée à l'image.

## Exemple

L'exemple suivant spécifie qu'une même courbe d'accélération personnalisée doit être utilisée pour toutes les propriétés de la première image du premier calque :

```
var theFrame = fl.getDocumentDOM().getTimeline().layers[0].frames[0]
theFrame.useSingleEaseCurve = true;
```

## Voir aussi

```
frame.getCustomEase(), frame.hasCustomEase, frame.setCustomEase()
```

# Objet halfEdge

## Disponibilité

Flash MX 2004.

## Description

L'objet HalfEdge est le côté dirigé du contour d'un [Objet Shape](#). Un contour se compose de deux demi-contours. Il est possible de suivre les contours d'une forme en « parcourant » ces demi-contours. Par exemple, en partant d'un demi-contour, vous pouvez suivre tous les demi-contours du contour d'une forme, et revenir ainsi au demi-contour d'origine.

Les demi-contours sont ordonnés. Un demi-contour représente un côté du contour, et l'autre demi-contour représente l'autre côté.

## Résumé des méthodes de l'objet halfEdge

L'objet HalfEdge gère les méthodes suivantes :

Méthode	Description
<code>halfEdge.getEdge()</code>	Lit l' <a href="#">Objet Edge</a> de l'objet HalfEdge.
<code>halfEdge.getNext()</code>	Lit le demi-contour suivant du contour actif.
<code>halfEdge.getOppositeHalfEdge()</code>	Lit l'objet HalfEdge situé de l'autre côté du contour.
<code>halfEdge.getPrev()</code>	Lit l'objet HalfEdge précédent du contour actif.
<code>halfEdge.getVertex()</code>	Lit l' <a href="#">Objet Vertex</a> au début de l'objet HalfEdge.

## Résumé des propriétés de l'objet halfEdge

Les propriétés suivantes sont disponibles pour l'objet HalfEdge :

Propriété	Description
<code>halfEdge.id</code>	En lecture seule ; entier représentant un identifiant unique pour l'objet HalfEdge.
<code>halfEdge.index</code>	Entier prenant la valeur 0 ou 1 pour déterminer l'index de cet objet HalfEdge dans le contour parent.



# halfEdge.getEdge()

## Disponibilité

Flash MX 2004.

## Utilisation

```
halfEdge.getEdge()
```

## Paramètres

Aucun.

## Valeur renvoyée

Un [Objet Edge](#).

## Description

Méthode ; lit l'objet Edge de l'objet HalfEdge. Voir [Objet Edge](#).

## Exemple

L'exemple suivant montre comment obtenir un contour et un demi-contour pour la forme spécifiée :

```
var shape = fl.getDocumentDOM().selection[0];  
var hEdge = shape.edges[0].getHalfEdge(0);  
var edge = hEdge.getEdge();
```

# halfEdge.getNext()

## Disponibilité

Flash MX 2004.

## Utilisation

```
halfEdge.getNext()
```

## Paramètres

Aucun.

## Valeur renvoyée

Un objet HalfEdge.

## Description

Méthode ; lit le demi-contour suivant du contour actif.

REMARQUE

Bien que les demi-contours possèdent des propriétés de direction et d'ordre séquentiel, ce n'est pas le cas des contours.

## Exemple

L'exemple suivant enregistre dans la variable `nextHalfEdge` le demi-contour suivant du contour spécifié :

```
var shape = fl.getDocumentDOM().selection[0];
var hEdge = shape.edges[0].getHalfEdge( 0 );
var nextHalfEdge = hEdge.getNext();
```

## halfEdge.getOppositeHalfEdge()

### Disponibilité

Flash MX 2004.

### Utilisation

```
halfEdge.getOppositeHalfEdge()
```

### Paramètres

Aucun.

### Valeur renvoyée

Un objet `HalfEdge`.

## Description

Méthode ; lit l'objet `HalfEdge` situé de l'autre côté du contour.

## Exemple

L'exemple suivant enregistre dans la variable `otherHalfEdge` l'objet `hEdge` du demi-contour opposé :

```
var shape = fl.getDocumentDOM().selection[0];
var hEdge = shape.edges[0].getHalfEdge(0);
var otherHalfEdge = hEdge.getOppositeHalfEdge();
```

# halfEdge.getPrev()

## Disponibilité

Flash MX 2004.

## Utilisation

```
halfEdge.getPrev()
```

## Paramètres

Aucun.

## Valeur renvoyée

Un objet HalfEdge.

## Description

Méthode ; lit l'objet HalfEdge précédent du contour actif.

### REMARQUE

Bien que les demi-contours possèdent des propriétés de direction et d'ordre séquentiel, ce n'est pas le cas des contours.

## Exemple

L'exemple suivant enregistre dans la variable `prevHalfEdge` le demi-contour précédent du contour spécifié :

```
var shape = fl.getDocumentDOM().selection[0];  
var hEdge = shape.edges[0].getHalfEdge( 0 );  
var prevHalfEdge = hEdge.getPrev();
```

# halfEdge.getVertex()

## Disponibilité

Flash MX 2004.

## Utilisation

`halfEdge.getVertex()`

## Paramètres

Aucun.

## Valeur renvoyée

Un [Objet Vertex](#)

## Description

Méthode ; lit l'objet Vertex au début de l'objet HalfEdge. Voir [Objet Vertex](#).

## Exemple

L'exemple suivant enregistre dans la variable `vertex` l'objet Vertex situé en tête de l'objet `hEdge` :

```
var shape = fl.getDocumentDOM().selection[0];  
var edge = shape.edges[0];  
var hEdge = edge.getHalfEdge(0);  
var vertex = hEdge.getVertex();
```

# halfEdge.id

## Disponibilité

Flash MX 2004.

## Utilisation

`halfEdge.id`

## Description

Propriété en lecture seule ; entier représentant un identifiant unique pour l'objet HalfEdge.

## Exemple

L'exemple suivant affiche dans le panneau de sortie un identificateur unique pour le demi-contour spécifié :

```
var shape = fl.getDocumentDOM().selection[0];  
alert(shape.contours[0].getHalfEdge().id);
```

# halfEdge.index

## Disponibilité

Flash MX 2004.

## Utilisation

`halfEdge.index`

## Description

Propriété en lecture seule ; entier possédant une valeur de 0 ou 1 qui détermine l'index de cet objet HalfEdge dans le contour parent.

## Exemple

L'exemple suivant affiche dans le panneau de sortie la valeur d'index pour le demi-contour spécifié :

```
var shape = fl.getDocumentDOM().selection[0];  
var hEdge = shape.edges[0].getHalfEdge(0);  
var heIndex = hEdge.index;
```

# Objet Instance

Héritage [Objet Element](#) > Objet Instance

## Disponibilité

Flash MX 2004.

## Description

L'objet Instance est une sous-classe de l'[Objet Element](#).

## Résumé des propriétés de l'objet Instance

Outre les propriétés de l'[Objet Element](#), l'objet Instance possède les propriétés suivantes :

Propriété	Description
<a href="#">instance.instanceType</a>	Propriété en lecture seule ; chaîne représentant le type d'instance.
<a href="#">instance.libraryItem</a>	Élément de bibliothèque utilisé pour instancier cette instance.

## instance.instanceType

### Disponibilité

Flash MX 2004 ; valeur gérée de "video" ajoutée à Flash 8.

### Utilisation

`instance.instanceType`

### Description

Propriété en lecture seule ; chaîne représentant le type d'instance. Les valeurs gérées sont les suivantes : "symbol", "bitmap", "embedded video" (vidéo incorporée), "linked video" (vidéo liée) et "compiled clip" (clip compilé).

REMARQUE

Dans Flash MX 2004, la valeur de `instance.instanceType` pour un élément ajouté à la bibliothèque à l'aide de `library.addItem("video")` est "embedded\_video". Dans Flash 8, la valeur est "video". Pour plus d'informations, consultez [library.addNewItem\(\)](#).

## Exemple

L'exemple suivant montre que le type d'instance d'un clip est "symbol":

```
// Sélectionner un clip, puis exécuter ce script.  
var type = fl.getDocumentDOM().selection[0].instanceType;  
fl.trace("This instance type is " + type);
```

## instance.libraryItem

### Disponibilité

Flash MX 2004.

### Utilisation

instance.libraryItem

### Description

Propriété ; élément de bibliothèque utilisé pour instancier cette instance. Il n'est possible de changer cette propriété qu'en la remplaçant par un autre objet de bibliothèque du même type (par exemple, vous ne pouvez pas faire référence à un bitmap avec une instance de type symbol). Voir [Objet library](#).

## Exemple

L'exemple suivant modifie le symbole sélectionné pour qu'il référence le premier élément de la bibliothèque :

```
fl.getDocumentDOM().selection[0].libraryItem =  
    fl.getDocumentDOM().library.items[0];
```

# Objet Item

## Disponibilité

Flash MX 2004.

## Description

L'objet Item est une classe de base abstraite. Tous les éléments de la bibliothèque dérivent de l'objet Item. Voir aussi [Objet library](#).

## Résumé des méthodes de l'objet Item

Les méthodes suivantes sont disponibles avec l'objet Item :

Méthode	Description
<code>item.addData()</code>	Ajoute les données spécifiées à un élément de la bibliothèque.
<code>item.getData()</code>	Lit la valeur des données spécifiées.
<code>item.hasData()</code>	Détermine si l'élément de bibliothèque comporte les données indiquées.
<code>item.removeData()</code>	Supprime les données persistantes de l'élément de bibliothèque.

## Résumé des propriétés de l'objet Item

Les propriétés suivantes sont disponibles avec l'objet Item :

Propriété	Description
<code>item.itemType</code>	Propriété en lecture seule ; chaîne indiquant le type d'élément.
<code>item.linkageBaseClass</code>	Chaîne spécifiant la classe ActionScript 3.0 qui sera associée au symbole.
<code>item.linkageClassName</code>	Chaîne spécifiant la classe ActionScript 2.0 qui sera associée au symbole.
<code>item.linkageExportForAS</code>	Une valeur booléenne. Si sa valeur est <code>true</code> , l'élément est exporté pour le code ActionScript.
<code>item.linkageExportForRS</code>	Valeur booléenne. Si sa valeur est <code>true</code> , l'élément est exporté pour le partage à l'exécution.



Propriété	Description
<code>item.linkageExportInFirstFrame</code>	Valeur booléenne. Si sa valeur est <code>true</code> , l'élément est exporté dans la première image.
<code>item.linkageIdentifier</code>	Chaîne spécifiant le nom qui permettra à Flash d'identifier la ressource au moment de créer une liaison avec le fichier SWF de destination.
<code>item.linkageImportForRS</code>	Valeur booléenne. Si sa valeur est <code>true</code> , l'élément est importé pour le partage à l'exécution.
<code>item.linkageURL</code>	Chaîne qui spécifie l'URL à laquelle se trouve le fichier SWF contenant l'élément partagé.
<code>item.name</code>	Chaîne qui spécifie le nom de l'élément de bibliothèque, structure de l'arborescence comprise.

## item.addData()

### Disponibilité

Flash MX 2004.

### Utilisation

`item.addData(name, type, data)`

### Paramètres

*nom* Chaîne spécifiant le nom des données.

*type* Chaîne spécifiant le type des données. Les valeurs possibles sont les suivantes : "integer", "integerArray", "double", "doubleArray", "string" et "byteArray".

*data* Données à ajouter à l'élément de bibliothèque spécifié. Le type de données dépend de la valeur du paramètre *type*. Par exemple, si *type* est « `integer` », la valeur des données doit être un entier, etc.

### Valeur renvoyée

Aucune.

### Description

Méthode ; ajoute les données spécifiées à un élément de la bibliothèque.

### Exemple

L'exemple suivant ajoute des données nommées « `myData` », avec une valeur de 12 sur un entier, au premier élément de la bibliothèque :

```
fl.getDocumentDOM().library.items[0].addData("myData", "integer", 12);
```

## item.getData()

### Disponibilité

Flash MX 2004.

### Utilisation

```
item.getData(name)
```

### Paramètres

*name* Chaîne spécifiant le nom des données à lire.

### Valeur renvoyée

Les données spécifiées par le paramètre *name*. Le type renvoyé dépend du type de données enregistrées.

### Description

Méthode ; lit la valeur des données spécifiées.

### Exemple

L'exemple suivant lit les valeurs des données nommées « myData » dans le premier élément de la bibliothèque et les enregistre dans la variable `libData` :

```
var libData = fl.getDocumentDOM().library.items[0].getData("myData");
```

## item.hasData()

### Disponibilité

Flash MX 2004.

### Utilisation

```
item.hasData(name)
```

### Paramètres

*name* Chaîne spécifiant le nom des données à vérifier dans l'élément de bibliothèque.

### Valeur renvoyée

Une valeur booléenne : `true` si les données spécifiées existent, `false` sinon.

### Description

Méthode ; détermine si l'élément de bibliothèque comporte les données indiquées.

## Exemple

L'exemple suivant affiche un message dans le panneau de sortie si le premier élément de la bibliothèque contient un point de coordonnées nommé `myData` :

```
if (fl.getDocumentDOM().library.items[0].hasData("myData")){  
    fl.trace("Yep, it's there!");  
}
```

# item.itemType

## Disponibilité

Flash MX 2004.

## Utilisation

`item.itemType`

## Description

Propriété en lecture seule ; chaîne indiquant le type d'élément. Elle peut avoir l'une des valeurs suivantes : "undefined", "component", "movie clip", "graphic", "button", "folder", "font", "sound", "bitmap", "compiled clip", "screen" et "video". Si cette propriété est "video", vous pouvez déterminer le type de vidéo ; voir [videoItem.videoType](#).

## Exemple

L'exemple suivant affiche le type de l'élément de bibliothèque spécifié dans le panneau de sortie :

```
fl.trace(fl.getDocumentDOM().library.items[0].itemType);
```

# item.linkageBaseClass

## Disponibilité

Flash CS3 Professional.

## Utilisation

`item.linkageBaseClass`

## Description

Propriété ; chaîne spécifiant la classe ActionScript 3.0 qui sera associée au symbole. La valeur spécifiée ici apparaît dans la boîte de dialogue Liaison de l'environnement de programmation, ainsi que dans toutes les autres boîtes de dialogue affichant les contrôles de cette option, par exemple la boîte de dialogue Propriétés du symbole. (Si vous voulez spécifier cette valeur pour une classe ActionScript 2.0, utilisez [item.linkageClassName](#).)

Si la classe de base est la classe par défaut pour le type de symbole (par exemple « `flash.display.MovieClip` » pour les clips, « `flash.display.SimpleButton` » pour les boutons, etc.), la valeur de cette propriété est une chaîne vide (" "). De même, vous définirez cette valeur sur une chaîne vide pour qu'un élément devienne la classe de base par défaut.

Lorsque vous définissez cette valeur, aucune des vérifications prévues par la boîte de dialogue Liaison n'est effectuée, et aucune erreur n'est renvoyée si Flash est dans l'incapacité de définir la valeur de la classe de base sur celle qui est spécifiée. Par exemple, le fait de définir cette valeur dans la boîte de dialogue Liaison oblige le processus de vérification à contrôler la présence de la classe de base dans le chemin d'accès de classe du fichier FLA. Il s'assure qu'ActionScript 3.0 est choisi dans l'onglet Flash de la boîte de dialogue Paramètres de publication, etc. Ces vérifications ne sont pas effectuées lorsque vous définissez cette propriété dans un script.

### Exemple

Les lignes de code suivantes illustrent plusieurs façons d'utiliser cette propriété :

```
// règle la classe de base de l'élément de bibliothèque sur "Sprite"
fl.getDocumentDOM().library.items[0].linkageBaseClass =
    "flash.display.Sprite";
// règle la classe de base de l'élément de bibliothèque sur la valeur
// par défaut pour ce type d'élément
fl.getDocumentDOM().library.items[0].linkageBaseClass = "";
// trouve et affiche la classe de base de l'élément de bibliothèque
fl.trace(fl.getDocumentDOM().library.items[0].linkageBaseClass);
```

### Voir aussi

[document.docClass](#)

## item.linkageClassName

### Disponibilité

Flash MX 2004.

### Utilisation

`item.linkageClassName`

### Description

Propriété ; chaîne spécifiant la classe ActionScript 2.0 qui sera associée au symbole. (Si vous voulez spécifier cette valeur pour une classe ActionScript 3.0, utilisez

[item.linkageBaseClass](#).)

Pour qu'il soit possible de définir cette propriété, les propriétés `item.linkageExportForAS` et/ou `item.linkageExportForRS` doivent être réglées sur `true`, et la propriété `item.linkageImportForRS` sur `false`.

### Exemple

L'exemple suivant spécifie que le nom de classe ActionScript 2.0 associé au premier élément de la bibliothèque est « `myClass` » :

```
fl.getDocumentDOM().library.items[0].linkageClassName = "myClass";
```

## item.linkageExportForAS

### Disponibilité

Flash MX 2004.

### Utilisation

`item.linkageExportForAS`

### Description

Propriété ; valeur booléenne. Si cette propriété a la valeur `true`, l'élément est exporté pour le code ActionScript. Vous pouvez également régler sur `true` les propriétés `item.linkageExportForRS` et `item.linkageExportInFirstFrame`.

Si vous donnez à cette propriété la valeur « `true` », la valeur de la propriété `item.linkageImportForRS` doit être fixée sur `false`. Par ailleurs, il est nécessaire de préciser un identificateur (`item.linkageIdentifier`) et une adresse URL (`item.linkageURL`).

### Exemple

L'exemple suivant définit cette propriété pour l'élément de bibliothèque spécifié :

```
fl.getDocumentDOM().library.items[0].linkageExportForAS = true;
```

# item.linkageExportForRS

## Disponibilité

Flash MX 2004.

## Utilisation

`item.linkageExportForRS`

## Description

Propriété ; valeur booléenne. Si cette propriété a la valeur `true`, l'élément est exporté pour le partage en cours d'exécution. Vous pouvez également régler sur `true` les propriétés `item.linkageExportForAS` et `item.linkageExportInFirstFrame`.

Si vous donnez à cette propriété la valeur « `true` », la valeur de la propriété `item.linkageImportForRS` doit être fixée sur `false`. Par ailleurs, il est nécessaire de préciser un identificateur (`item.linkageIdentifier`) et une adresse URL (`item.linkageURL`).

## Exemple

L'exemple suivant définit cette propriété pour l'élément de bibliothèque spécifié :

```
fl.getDocumentDOM().library.items[0].linkageExportForRS = true;
```

# item.linkageExportInFirstFrame

## Disponibilité

Flash MX 2004.

## Utilisation

`item.linkageExportInFirstFrame`

## Description

Propriété ; valeur booléenne. Si elle vaut `true`, l'élément est exporté dans la première image ; si elle est `false`, l'élément est exporté dans l'image de la première occurrence. Si l'élément n'apparaît pas sur la scène, il n'est pas exporté.

Cette propriété peut être réglée sur `true` uniquement si les valeurs de `item.linkageExportForAS` et/ou de `item.linkageExportForRS` sont également définies sur `true`.

## Exemple

L'exemple suivant spécifie que l'élément de bibliothèque spécifié doit être exporté dans la première image :

```
fl.getDocumentDOM().library.items[0].linkageExportInFirstFrame = true;
```

# item.linkageIdentifier

## Disponibilité

Flash MX 2004.

## Utilisation

`item.linkageIdentifier`

## Description

Propriété ; chaîne spécifiant le nom qui permettra à Flash d'identifier l'élément au moment de créer une liaison vers le fichier SWF de destination. Flash ignore cette propriété si `item.linkageImportForRS`, `item.linkageExportForAS` et `item.linkageExportForRS` ont la valeur `false`. A l'inverse, cette propriété doit être définie si l'une de ces propriétés est réglée sur `true`.

## Exemple

L'exemple suivant spécifie que la chaîne `my_mc` sera utilisée pour identifier l'élément de bibliothèque au moment de le lier au fichier SWF vers lequel il sera exporté :

```
fl.getDocumentDOM().library.items[0].linkageIdentifier = "my_mc";
```

## Voir aussi

[item.linkageURL](#)

# item.linkageImportForRS

## Disponibilité

Flash MX 2004.

## Utilisation

`item.linkageImportForRS`

## Description

Propriété ; valeur booléenne. Si sa valeur est `true`, l'élément est importé pour le partage à l'exécution. Si cette propriété est définie comme `true`, `item.linkageExportForAS` et `item.linkageExportForRS` doivent recevoir la valeur `false`. Par ailleurs il est nécessaire de préciser un identificateur (`item.linkageIdentifier`) et une adresse URL (`item.linkageURL`).

## Exemple

L'exemple suivant définit cette propriété comme `true` pour l'élément de bibliothèque spécifié :

```
fl.getDocumentDOM().library.items[0].linkageImportForRS = true;
```

# item.linkageURL

## Disponibilité

Flash MX 2004.

## Utilisation

`item.linkageURL`

## Description

Propriété ; chaîne qui spécifie l'URL à laquelle se trouve le fichier SWF contenant l'élément partagé. Flash ignore cette propriété si `item.linkageImportForRS`, `item.linkageExportForAS` et `item.linkageExportForRS` ont la valeur `false`. À l'inverse, cette propriété doit être définie si l'une de ces propriétés est réglée sur `true`. Il est possible d'indiquer une adresse de type URL ou un nom de fichier au format d'une plate-forme précise (c'est-à-dire avec une barre oblique normale (/) ou inversée (\), selon la plate-forme).

## Exemple

L'exemple suivant spécifie une adresse URL comme lien vers l'élément de bibliothèque spécifié :

```
fl.getDocumentDOM().library.items[0].linkageURL = "theShareSWF.swf";
```

## Voir aussi

`item.linkageIdentifier`

# item.name

## Disponibilité

Flash MX 2004.

## Utilisation

`item.name`

## Description

Méthode ; chaîne qui spécifie le nom de l'élément de bibliothèque, structure de l'arborescence comprise. Par exemple, si `Symbol_1` se trouve à l'intérieur d'un dossier nommé `Folder_1`, la propriété `name` de `Symbol_1` reçoit la valeur `"Folder_1/Symbol_1"`.

## Exemple

L'exemple suivant affiche le nom de l'élément de bibliothèque spécifié dans le panneau de sortie :

```
fl.trace(fl.getDocumentDOM().library.items[0].name);
```



# item.removeData()

## Disponibilité

Flash MX 2004.

## Utilisation

```
item.removeData(name)
```

## Paramètres

*name* Chaîne spécifiant le nom des données à supprimer de l'élément de bibliothèque.

## Valeur renvoyée

Aucune.

## Description

Propriété ; supprime les données persistantes de l'élément de bibliothèque.

## Exemple

L'exemple suivant supprime les données nommées « myData » du premier élément de la bibliothèque :

```
fl.getDocumentDOM().library.items[0].removeData("myData");
```

# Objet Layer

## Disponibilité

Flash MX 2004.

## Description

L'objet Layer représente un calque dans le scénario. La propriété `timeline.layers` contient un tableau d'objets Layer, auxquels il est possible d'accéder à l'aide de la méthode `fl.getDocumentDOM().getTimeline().layers`.

## Résumé des propriétés de l'objet Layer

Les propriétés suivantes sont disponibles pour l'objet Layer :

Propriété	Description
<code>layer.color</code>	Chaîne, valeur hexadécimale ou entier qui spécifie la couleur utilisée pour indiquer les contours du calque.
<code>layer.frameCount</code>	En lecture seule ; entier précisant le nombre d'images du calque.
<code>layer.frames</code>	Propriété en lecture seule ; tableau des objets Frame.
<code>layer.height</code>	Entier qui spécifie la hauteur du calque en pourcentage ; équivalent du champ Hauteur du calque de la boîte de dialogue Propriétés du calque.
<code>layer.layerType</code>	Chaîne qui spécifie l'utilisation actuelle du calque ; équivalent du champ Type dans la boîte de dialogue Propriétés du calque.
<code>layer.locked</code>	Valeur booléenne spécifiant l'état de verrouillage du calque.
<code>layer.name</code>	Chaîne spécifiant le nom du calque.
<code>layer.outline</code>	Valeur booléenne précisant l'état des contours de tous les objets du calque.
<code>layer.parentLayer</code>	Objet Layer qui représente l'objet parent du calque (dossier de calques, calque de guidage ou calque de masquage).
<code>layer.visible</code>	Valeur booléenne indiquant si les objets du calque sur la scène sont affichés ou masqués.

# layer.color

## Disponibilité

Flash MX 2004.

## Utilisation

`layer.color`

## Description

Propriété ; couleur à utiliser pour délimiter le calque, exprimée dans l'un des formats suivants :

- Chaîne au format "#RRVBBB" ou "#RRVVBBAA"
- Nombre hexadécimal au format 0xRRVVBB
- Entier représentant l'équivalent décimal d'un nombre hexadécimal

Cette propriété est l'équivalent du champ Couleur de contour dans la boîte de dialogue Propriétés du calque.

## Exemple

L'exemple suivant enregistre dans la variable `colorValue` la valeur du premier calque :

```
var colorValue = fl.getDocumentDOM().getTimeline().layers[0].color;
```

L'exemple suivant présente les trois façons de définir comme « rouge » la couleur du premier calque :

```
fl.getDocumentDOM().getTimeline().layers[0].color=16711680;  
fl.getDocumentDOM().getTimeline().layers[0].color="#ff0000";  
fl.getDocumentDOM().getTimeline().layers[0].color=0xFF0000;
```

# layer.frameCount

## Disponibilité

Flash MX 2004.

## Utilisation

`layer.frameCount`

## Description

Propriété en lecture seule ; entier indiquant le nombre d'images du calque.

## Exemple

L'exemple suivant enregistre dans la variable `fcNum` le nombre d'images du premier calque :

```
var fcNum = fl.getDocumentDOM().getTimeline().layers[0].frameCount;
```

# layer.frames

## Disponibilité

Flash MX 2004.

## Utilisation

layer.frames

## Description

Propriété en lecture seule ; tableau des objets Frame (voir [Objet Frame](#)).

## Exemple

L'exemple suivant copie dans la variable `frameArray` le tableau d'objets Frame représentant les images du document actif :

```
var frameArray = fl.getDocumentDOM().getTimeline().layers[0].frames;
```

Pour déterminer si une image est une image-clé, regardez si la propriété `frame.startFrame` correspond à son index dans le tableau, comme dans l'exemple suivant :

```
var frameArray = fl.getDocumentDOM().getTimeline().layers[0].frames;
var n = frameArray.length;
for (i=0; i<n; i++) {
    if (i==frameArray[i].startFrame) {
        alert("Keyframe at: " + i);
    }
}
```

# layer.height

## Disponibilité

Flash MX 2004.

## Utilisation

layer.height

## Description

Propriété ; entier qui spécifie la hauteur du calque en pourcentage ; équivalent du champ Hauteur du calque de la boîte de dialogue Propriétés du calque. Les valeurs possibles représentent un pourcentage de la hauteur par défaut : 100, 200 ou 300.

## Exemple

L'exemple suivant enregistre la valeur de pourcentage de hauteur du premier calque :

```
var layerHeight = fl.getDocumentDOM().getTimeline().layers[0].height;
```

L'exemple suivant fixe la hauteur du premier calque à 300 % :

```
fl.getDocumentDOM().getTimeline().layers[0].height = 300;
```

# layer.layerType

## Disponibilité

Flash MX 2004.

## Utilisation

layer.layerType

## Description

Propriété ; chaîne qui spécifie l'utilisation actuelle du calque ; équivalent du champ Type dans la boîte de dialogue Propriétés du calque. Les valeurs gérées sont les suivantes : "normal", "guide", "guided" (guidé), "mask" (masque), "masked" (masqué) et "folder" (dossier).

## Exemple

L'exemple suivant attribue le type "folder" au premier calque du scénario :

```
fl.getDocumentDOM().getTimeline().layers[0].layerType = "folder";
```

# layer.locked

## Disponibilité

Flash MX 2004.

## Utilisation

layer.locked

## Description

Propriété ; valeur booléenne spécifiant l'état de verrouillage du calque. Si cette propriété est true, le calque est verrouillé. La valeur par défaut est false.

## Exemple

L'exemple suivant enregistre dans la variable lockStatus la valeur booléenne d'état de verrouillage du premier calque :

```
var lockStatus = fl.getDocumentDOM().getTimeline().layers[0].locked;
```

L'exemple suivant définit l'état du premier calque comme verrouillé :

```
fl.getDocumentDOM().getTimeline().layers[0].locked = false;
```

# layer.name

## Disponibilité

Flash MX 2004.

## Utilisation

`layer.name`

## Description

Propriété ; chaîne qui spécifie le nom du calque.

## Exemple

L'exemple suivant définit comme « foreground » le nom du premier calque du document actuel :

```
fl.getDocumentDOM().getTimeline().layers[0].name = "foreground";
```

# layer.outline

## Disponibilité

Flash MX 2004.

## Utilisation

`layer.outline`

## Description

Propriété ; valeur booléenne spécifiant l'état des contours de tous les objets du calque. Si cette propriété est réglée sur `true`, seuls les contours des objets du calque sont affichés. Si elle est réglée sur `false`, les objets apparaissent tels qu'ils ont été créés.

## Exemple

L'exemple suivant fait afficher uniquement les contours des objets du premier calque :

```
fl.getDocumentDOM().getTimeline().layers[0].outline = true;
```

# layer.parentLayer

## Disponibilité

Flash MX 2004.

## Utilisation

layer.parentLayer

## Description

Propriété ; objet Layer qui représente l'objet parent du calque (dossier de calques, calque de guidage ou calque de masquage). Le calque parent doit être un dossier conteneur du calque, un calque de guide ou de masque précédant le calque dans la hiérarchie, ou le calque parent (parentLayer) du calque précédent ou suivant. Le paramétrage de l'objet parentLayer du calque ne modifie pas la position du calque dans la liste ; de ce fait, tenter de définir le parentLayer d'un calque comme un calque qui nécessiterait son déplacement est sans effet. Cette propriété utilise la valeur null pour un calque de niveau supérieur.

## Exemple

L'exemple suivant utilise deux calques de même niveau dans le même scénario. Le premier calque (layers[0]) est converti en dossier, puis défini comme dossier parent du second calque (layers[1]). Cette action place le second calque dans le premier.

```
var parLayer = fl.getDocumentDOM().getTimeline().layers[0];
parLayer.layerType = "folder";
fl.getDocumentDOM().getTimeline().layers[1].parentLayer = parLayer;
```

# layer.visible

## Disponibilité

Flash MX 2004.

## Utilisation

layer.visible

## Description

Propriété ; valeur booléenne indiquant si les objets du calque sur la scène sont affichés ou masqués. Si la valeur de cette propriété est true, tous les objets du calque sont visibles ; si elle est false, ils sont masqués. La valeur par défaut est true.

## Exemple

L'exemple suivant rend invisibles les objets du premier calque :

```
fl.getDocumentDOM().getTimeline().layers[0].visible = false;
```

# Objet library

## Disponibilité

Flash MX 2004.

## Description

L'objet `library` représente le panneau Bibliothèque. C'est une propriété de l'objet `Document` (voir `document.library`) et il est possible d'y accéder à l'aide de la méthode `fl.getDocumentDOM().library`.

L'objet `library` contient un tableau d'éléments de différents types (symboles, bitmaps, sons et vidéo).

## Résumé des méthodes de l'objet library

Les méthodes suivantes sont disponibles avec l'objet `library` :

Méthode	Description
<code>library.addItemToDocument()</code>	Ajoute l'élément actif ou un élément spécifié à la scène, à l'emplacement spécifié.
<code>library.addNewItem()</code>	Crée dans le panneau Bibliothèque un nouvel élément du type spécifié, et active ce nouvel élément.
<code>library.deleteItem()</code>	Supprime du panneau Bibliothèque les éléments actifs ou un élément spécifié.
<code>library.duplicateItem()</code>	Copie l'élément actif ou un élément spécifié.
<code>library.editItem()</code>	Ouvre l'élément actif ou un élément spécifié en mode de modification.
<code>library.expandFolder()</code>	Développe ou réduit le dossier actif ou un dossier spécifié dans la bibliothèque.
<code>library.findItemIndex()</code>	Renvoie l'index (en base zéro) de l'élément dans la bibliothèque.
<code>library.getItemProperty()</code>	Lit la propriété de l'objet sélectionné.
<code>library.getItemType()</code>	Lit le type de l'objet sélectionné ou d'un objet spécifié par son chemin d'accès dans la bibliothèque.
<code>library.getSelectedItems()</code>	Lit le tableau de tous les éléments sélectionnés dans la bibliothèque.
<code>library.importEmbeddedSWF()</code>	Importe un fichier SWF dans la bibliothèque sous la forme d'un clip compilé.
<code>library.itemExists()</code>	Vérifie l'existence de l'élément spécifié dans la bibliothèque.



Méthode	Description
<code>library.moveToFolder()</code>	Déplace l'élément actif ou un élément de bibliothèque spécifié dans un dossier spécifié.
<code>library.newFolder()</code>	Crée un nouveau dossier portant le nom spécifié (ou le nom par défaut "dossier sans nom n°") si le paramètre <code>folderName</code> n'est pas transmis) dans le dossier sélectionné.
<code>library.renameItem()</code>	Renomme l'élément de bibliothèque sélectionné dans le panneau Bibliothèque.
<code>library.selectAll()</code>	Sélectionne ou désélectionne tous les éléments de la bibliothèque.
<code>library.selectItem()</code>	Sélectionne l'élément de bibliothèque spécifié.
<code>library.selectNone()</code>	Retire tous les éléments de bibliothèque de la sélection.
<code>library.setItemProperty()</code>	Définit la propriété de tous les objets de bibliothèque sélectionnés (en ignorant les dossiers).
<code>library.updateItem()</code>	Actualise l'élément de bibliothèque spécifié.

## Résumé des propriétés de l'objet library

La propriété suivante est disponible avec l'objet Library :

Propriété	Description
<code>library.items</code>	Tableau des objets sélectionnés dans la bibliothèque.

## library.addItemToDocument()

### Disponibilité

Flash MX 2004.

### Utilisation

```
library.addItemToDocument(position [, namePath])
```

### Paramètres

*position* Point qui indique la position *x,y* du centre de l'élément sur la scène.

*namePath* Chaîne spécifiant le nom de l'élément. Si l'élément se trouve dans un dossier, il est possible d'indiquer son nom et son chemin d'accès en utilisant la notation avec barre oblique. Si *namePath* n'est pas spécifié, la sélection actuelle dans la bibliothèque est utilisée. Ce paramètre est facultatif.

### Valeur renvoyée

Une valeur booléenne : `true` si l'élément est ajouté au document, `false` sinon.

### Description

Méthode ; ajoute l'élément actif ou un élément spécifié à la scène, à l'emplacement spécifié.

### Exemple

L'exemple suivant ajoute l'élément actif ou un élément spécifié à la scène, à la position (3,60).

```
fl.getDocumentDOM().library.addItemToDocument({x:3, y:60});
```

L'exemple suivant ajoute sur la scène l'élément `Symbol1` (qui se trouve dans le dossier `folder1` de la bibliothèque) à la position (550,485) :

```
fl.getDocumentDOM().library.addItemToDocument({x:550.0, y:485.0}, "folder1/Symbol1");
```

## library.addNewItem()

### Disponibilité

Flash MX 2004.

### Utilisation

```
library.addNewItem(type [, namePath])
```

### Paramètres

*type* Chaîne spécifiant le type d'élément à créer. Les seules valeurs possibles pour *type* sont les suivantes : `"video"`, `"movie clip"`, `"button"`, `"graphic"`, `"bitmap"`, `"screen"` et `"folder"` (cette méthode ne permet donc pas d'ajouter un son à la bibliothèque). L'indication d'un chemin de dossier donne un résultat identique à un appel de `library.newFolder()` avant d'appeler cette méthode.

*namePath* Chaîne spécifiant le nom de l'élément à ajouter. Si l'élément se trouve dans un dossier, il est possible d'indiquer son nom et son chemin d'accès en utilisant la notation avec barre oblique. Ce paramètre est facultatif.

### Valeur renvoyée

Une valeur booléenne : `true` en cas de réussite, `false` sinon.

### Description

Méthode ; crée dans le panneau Bibliothèque un nouvel élément du type spécifié et active ce nouvel élément. Pour plus d'informations sur l'importation d'éléments dans la bibliothèques, par exemple des sons, voir `document.importFile()`.

## Exemple

L'exemple suivant crée un nouvel élément de bouton nommé « start » dans un nouveau dossier nommé « folderTwo » :

```
fl.getDocumentDOM().library.addItem("button", "folderTwo/start");
```

## library.deleteItem()

### Disponibilité

Flash MX 2004.

### Utilisation

```
library.deleteItem([namePath])
```

### Paramètres

*namePath* Chaîne spécifiant le nom de l'élément à supprimer. Si l'élément se trouve dans un dossier, il est possible d'indiquer son nom et son chemin d'accès en utilisant la notation avec barre oblique. Si vous transmettez le nom d'un dossier, ce dossier et tous les éléments qu'il contient sont supprimés. Si aucun nom n'est spécifié, Flash supprime le ou les éléments actuellement sélectionnés. Pour supprimer tous les éléments du panneau Bibliothèque, sélectionnez tous les éléments avant d'utiliser cette méthode. Ce paramètre est facultatif.

### Valeur renvoyée

Une valeur booléenne : `true` en cas de réussite, `false` sinon.

### Description

Méthode ; supprime du panneau Bibliothèque les éléments actifs ou un élément spécifié. Si plusieurs éléments sont sélectionnés, cette méthode les supprime tous.

## Exemple

L'exemple suivant supprime l'élément sélectionné :

```
fl.getDocumentDOM().library.deleteItem();
```

L'exemple suivant supprime l'élément `Symbol_1` du dossier `Folder_1` de la bibliothèque :

```
fl.getDocumentDOM().library.deleteItem("Folder_1/Symbol_1");
```

# library.duplicateItem()

## Disponibilité

Flash MX 2004.

## Utilisation

```
library.duplicateItem( [ namePath ] )
```

## Paramètres

*namePath* Chaîne spécifiant le nom de l'élément à dupliquer. Si l'élément se trouve dans un dossier, il est possible d'indiquer son nom et son chemin d'accès en utilisant la notation avec barre oblique. Ce paramètre est facultatif.

## Valeur renvoyée

Une valeur booléenne : `true` en cas de réussite, `false` sinon. Si vous avez sélectionné plusieurs objets, Flash renvoie `false`.

## Description

Méthode ; copie l'élément actif ou un élément spécifié. Le nouvel élément reçoit un nom par défaut (par exemple `copie d'élément`) et devient l'élément sélectionné. Si vous avez sélectionné plusieurs objets, la commande échoue.

## Exemple

L'exemple suivant crée une copie de l'élément « square » dans le dossier test de la bibliothèque :

```
fl.getDocumentDOM().library.duplicateItem("test/square");
```

# library.editItem()

## Disponibilité

Flash MX 2004.

## Utilisation

```
library.editItem([namePath])
```

## Paramètres

*namePath* Chaîne spécifiant le nom de l'élément. Si l'élément se trouve dans un dossier, il est possible d'indiquer son nom et son chemin d'accès en utilisant la notation avec barre oblique. Si *namePath* n'est pas spécifié, l'élément de bibliothèque sélectionné s'ouvre en mode de modification. Si aucun élément n'est sélectionné, ou si au contraire plusieurs éléments sont sélectionnés, la première séquence du scénario principal est ouverte en mode de modification. Ce paramètre est facultatif.

## Valeur renvoyée

Une valeur booléenne : `true` si l'élément spécifié existe et peut être modifié, `false` sinon.

## Description

Méthode ; ouvre l'élément actif ou un élément spécifié en mode de modification.

## Exemple

L'exemple suivant ouvre pour modification l'élément « circle », qui se trouve dans le dossier test de la bibliothèque :

```
fl.getDocumentDOM().library.editItem("test/circle");
```

# library.expandFolder()

## Disponibilité

Flash MX 2004.

## Utilisation

```
library.expandFolder(bExpand [, bRecurseNestedParents [, namePath]])
```

## Paramètres

*bExpand* Une valeur booléenne : si elle est `true`, le dossier est développé ; si elle est `false` (la valeur par défaut), le dossier est réduit.

*bRecurseNestedParents* Une valeur booléenne : si elle est `true`, tous les sous-dossiers du dossier spécifié sont développés ou réduits, selon la valeur du paramètre *bExpand*. La valeur par défaut est `false`. Ce paramètre est facultatif.

*namePath* Chaîne spécifiant le nom et, éventuellement, le chemin d'accès du dossier à développer ou à réduire. Si ce paramètre n'est pas spécifié, la méthode s'applique au dossier actuellement sélectionné. Ce paramètre est facultatif.

### Valeur renvoyée

Une valeur booléenne : `true` si l'élément a été développé ou réduit, `false` si l'opération a échoué ou si l'élément spécifié n'est pas un dossier.

### Description

Méthode ; développe ou réduit le dossier actif ou un dossier spécifié dans la bibliothèque.

### Exemple

L'exemple suivant réduit le dossier « test », ainsi que tous ses sous-dossiers, le cas échéant :

```
fl.getDocumentDOM().library.expandFolder(false, true, "test");
```

## library.findItemIndex()

### Disponibilité

Flash MX 2004.

### Utilisation

```
library.findItemIndex(namePath)
```

### Paramètres

*namePath* Chaîne spécifiant le nom de l'élément. Si l'élément se trouve dans un dossier, il est possible d'indiquer son nom et son chemin d'accès en utilisant la notation avec barre oblique.

### Valeur renvoyée

Nombre entier représentant l'index (en base zéro) de l'élément dans la bibliothèque.

### Description

Méthode ; renvoie l'index (en base zéro) de l'élément dans la bibliothèque. L'index de la bibliothèque possède une seule dimension, si bien que les dossiers font partie de l'indexation. Il est possible d'utiliser les chemins d'accès des dossiers pour spécifier un élément imbriqué.

### Exemple

L'exemple suivant enregistre dans la variable `sqIndex` la valeur d'index (en base zéro) de l'élément de bibliothèque « square » qui se trouve dans le dossier « test », puis affiche cette valeur dans une boîte de dialogue :

```
var sqIndex = fl.getDocumentDOM().library.findItemIndex("test/square");  
alert(sqIndex);
```

# library.getItemProperty()

## Disponibilité

Flash MX 2004.

## Utilisation

```
library.getItemProperty(property)
```

## Paramètres

*property* Chaîne. La liste des valeurs possibles pour le paramètre *property* figure dans la section [Résumé des propriétés de l'objet Item](#), avec le résumé des propriétés de ses sous-classes.

## Valeur renvoyée

La valeur de chaîne de la propriété.

## Description

Méthode ; lit la propriété de l'objet sélectionné.

## Exemple

L'exemple suivant affiche dans une boîte de dialogue la valeur de l'identifiant de liaison du symbole tel qu'il est référencé en code ActionScript ou pour le partage à l'exécution :

```
alert(fl.getDocumentDOM().library.getItemProperty("linkageIdentifier"));
```

# library.getItemType()

## Disponibilité

Flash MX 2004.

## Utilisation

```
library.getItemType([namePath])
```

## Paramètres

*namePath* Chaîne spécifiant le nom de l'élément. Si l'élément se trouve dans un dossier, il est possible d'indiquer son nom et son chemin d'accès en utilisant la notation avec barre oblique. Si *namePath* n'est pas spécifié, Flash renvoie le type de la sélection actuelle. Si plusieurs éléments sont sélectionnés et si le paramètre *namePath* n'est pas spécifié, Flash ignore la commande. Ce paramètre est facultatif.

## Valeur renvoyée

Une valeur chaîne qui spécifie le type de l'objet. Pour la liste des valeurs pouvant être renvoyées, voir [item.itemType](#).

## Description

Méthode ; lit le type de l'objet sélectionné ou d'un objet spécifié par son chemin d'accès dans la bibliothèque.

## Exemple

L'exemple suivant affiche dans une boîte de dialogue le type de l'élément Symbol\_1, qui se trouve dans le dossier Folder\_1/Folder\_2 :

```
alert(fl.getDocumentDOM().library.getItemType("Folder_1/Folder_2/  
Symbol_1"));
```

# library.getSelectedItems()

## Disponibilité

Flash MX 2004.

## Paramètres

Aucun.

## Valeur renvoyée

Le tableau des valeurs de tous les éléments sélectionnés dans la bibliothèque.

## Description

Méthode ; lit le tableau de tous les éléments sélectionnés dans la bibliothèque.

## Exemple

L'exemple suivant enregistre dans la variable selItems le tableau des éléments de la bibliothèque actuellement sélectionnés (ici, plusieurs fichiers audio), puis donne à la propriété sampleRate du premier fichier audio du tableau la valeur "11 kHz":

```
var selItems = fl.getDocumentDOM().library.getSelectedItems();  
selItems[0].sampleRate = "11 kHz";
```

# library.importEmbeddedSWF()

## Disponibilité

Flash MX 2004.

## Utilisation

```
library.importEmbeddedSWF(linkageName, swfData [, libName])
```



## Paramètres

*linkageName* Chaîne indiquant le nom de la liaison SWF du clip racine.

*swfData* Tableau de données SWF binaires, qui provient d'une bibliothèque externe ou d'une DLL.

*libName* Chaîne spécifiant le nom, dans la bibliothèque, de l'élément créé. Si ce nom est déjà utilisé, la méthode en crée un autre. Ce paramètre est facultatif.

## Valeur renvoyée

Aucune.

## Description

Méthode ; importe un fichier SWF dans la bibliothèque sous la forme d'un clip compilé. À la différence de l'option de menu Fichier > Importer > SWF, cette méthode permet d'incorporer dans la bibliothèque un fichier SWF compilé. Il n'existe dans l'interface utilisateur aucune fonctionnalité correspondante, et cette méthode doit être utilisée avec une bibliothèque externe ou une DLL (voir [Chapitre 3, « Extensibilité de niveau C », à la page 611](#)).

Le fichier SWF que vous importez doit posséder un clip de niveau supérieur qui contient la totalité du contenu. L'identificateur de liaison de ce clip doit être fixé à la même valeur que le paramètre *linkageName* transmis à cette méthode.

## Exemple

L'exemple suivant ajoute à la bibliothèque le fichier SWF ayant la valeur « MyMovie » pour la propriété *linkageName*, sous forme d'un clip compilé nommé « Intro » :

```
fl.getDocumentDOM().library.importEmbeddedSWF("MyMovie", swfData, "Intro");
```

# library.itemExists()

## Disponibilité

Flash MX 2004.

## Utilisation

```
library.itemExists(namePath)
```

## Paramètres

*namePath* Chaîne spécifiant le nom de l'élément. Si l'élément se trouve dans un dossier, il est possible d'indiquer son nom et son chemin d'accès en utilisant la notation avec barre oblique.

### Valeur renvoyée

Une valeur booléenne : `true` si l'élément spécifié existe dans la bibliothèque, `false` sinon.

### Description

Méthode ; vérifie l'existence de l'élément spécifié dans la bibliothèque.

### Exemple

L'exemple suivant affiche `true` ou `false` dans une boîte de dialogue, selon que l'élément `Symbol_1` existe ou non dans le dossier `Folder_1` de la bibliothèque :

```
alert(fl.getDocumentDOM().library.itemExists('Folder_1/Symbol_1'));
```

## library.items

### Disponibilité

Flash MX 2004.

### Utilisation

`library.items`

### Description

Propriété ; tableau des objets sélectionnés dans la bibliothèque.

### Exemple

L'exemple suivant enregistre dans la variable `itemArray` le tableau de tous les éléments contenus dans la bibliothèque :

```
var itemArray = fl.getDocumentDOM().library.items;
```

## library.moveToFolder()

### Disponibilité

Flash MX 2004.

### Utilisation

```
library.moveToFolder(folderPath [, itemToMove [, bReplace]])
```

### Paramètres

*folderPath* Chaîne indiquant le chemin d'accès du dossier sous la forme "nom\_de\_dossier" ou "nom\_de\_dossier1/nom\_de\_dossier2". Pour déplacer un élément au niveau supérieur, passez une chaîne vide (" ") pour le paramètre *folderPath*.

*itemToMove* Chaîne spécifiant le nom de l'élément à déplacer. Si *itemToMove* n'est pas spécifié, c'est la sélection actuelle dans la bibliothèque qui est déplacée. Ce paramètre est facultatif.

*bReplace* Valeur Booléenne. S'il existe déjà un élément du même nom, le fait de spécifier *true* pour le paramètre *bReplace* remplace l'élément existant par l'élément déplacé. Si la valeur de ce paramètre est *false*, le nom de l'élément déplacé est modifié pour être unique. La valeur par défaut est *false*. Ce paramètre est facultatif.

### Valeur renvoyée

Une valeur booléenne : *true* en cas de réussite, *false* sinon.

### Description

Méthode ; déplace l'élément actif ou un élément de bibliothèque spécifié dans un dossier spécifié. Si le paramètre *folderPath* est vide, les éléments sont déplacés vers le niveau supérieur.

### Exemple

L'exemple suivant déplace l'élément « Symbol\_1 » dans le dossier « new » en remplacement de l'élément qui porte déjà ce nom dans ce dossier :

```
fl.getDocumentDOM().library.moveToFolder("new", "Symbol_1", true);
```

## library.newFolder()

### Disponibilité

Flash MX 2004.

### Utilisation

```
library.newFolder([folderPath])
```

### Paramètres

*folderPath* Chaîne spécifiant le nom du dossier à créer. Si cette chaîne contient un chemin et que ce chemin n'existe pas, il est alors créé. Ce paramètre est facultatif.

### Valeur renvoyée

Une valeur booléenne : *true* en cas de réussite, *false* sinon.

### Description

Méthode ; crée un nouveau dossier ayant le nom spécifié (ou le nom par défaut "dossier sans nom n°") si le paramètre *folderName* n'est pas transmis) dans le dossier sélectionné.

### Exemple

L'exemple suivant crée deux dossiers de bibliothèque. Le deuxième est un sous-dossier du premier :

```
fl.getDocumentDOM().library.newFolder("first/second");
```

## library.renameItem()

### Disponibilité

Flash MX 2004.

### Utilisation

```
library.renameItem(name)
```

### Paramètres

*name* Chaîne spécifiant le nouveau nom de l'élément de bibliothèque.

### Valeur renvoyée

Valeur booléenne *true* si le nom de l'élément est modifié, *false* dans le cas contraire. Si plusieurs éléments sont sélectionnés, aucun nom n'est modifié et la valeur renvoyée correspond à *false* (comme dans l'interface utilisateur).

### Description

Méthode ; renomme l'élément de bibliothèque sélectionné dans le panneau Bibliothèque.

### Exemple

L'exemple suivant renomme avec le nom "new name" l'élément de bibliothèque actuellement sélectionné :

```
fl.getDocumentDOM().library.renameItem("new name");
```

## library.selectAll()

### Disponibilité

Flash MX 2004.

### Utilisation

```
library.selectAll([bSelectAll])
```

## Paramètres

*bSelectAll* Valeur booléenne qui indique si tous les éléments de la bibliothèque doivent être sélectionnés ou désélectionnés. Pour sélectionner tous les éléments de la bibliothèque, vous pouvez omettre ce paramètre ou utiliser la valeur par défaut (*true*) ; la valeur *false* désélectionne tous les éléments. Ce paramètre est facultatif.

## Valeur renvoyée

Aucune.

## Description

Méthode ; sélectionne ou désélectionne tous les éléments de la bibliothèque.

## Exemple

Les exemples suivants sélectionnent tous les éléments de la bibliothèque :

```
fl.getDocumentDOM().library.selectAll();  
fl.getDocumentDOM().library.selectAll(true);
```

Les exemples suivants désélectionnent tous les éléments de la bibliothèque :

```
fl.getDocumentDOM().library.selectAll(false);  
fl.getDocumentDOM().library.selectNone();
```

# library.selectItem()

## Disponibilité

Flash MX 2004.

## Utilisation

```
library.selectItem(namePath [, bReplaceCurrentSelection [, bSelect]])
```

## Paramètres

*namePath* Chaîne spécifiant le nom de l'élément. Si l'élément se trouve dans un dossier, il est possible d'indiquer son nom et son chemin d'accès en utilisant la notation avec barre oblique.

*bReplaceCurrentSelection* Valeur booléenne qui indique si le programme doit remplacer la sélection actuelle ou ajouter l'élément à cette sélection. La valeur par défaut est *true* (remplacer la sélection actuelle). Ce paramètre est facultatif.

*bSelect* Valeur booléenne qui indique si l'élément doit être sélectionné ou désélectionné. La valeur par défaut est *true* (sélectionner). Ce paramètre est facultatif.

### Valeur renvoyée

Une valeur booléenne : `true` si l'élément spécifié existe, `false` sinon.

### Description

Méthode ; sélectionne l'élément de bibliothèque spécifié.

### Exemple

L'exemple suivant désélectionne la sélection actuelle et sélectionne dans la bibliothèque l'élément symbol 1, qui se trouve dans le dossier untitled folder 1 :

```
fl.getDocumentDOM().library.selectItem("untitled Folder_1/Symbol_1");
```

L'exemple suivant étend la sélection actuelle dans la bibliothèque en y incluant l'élément symbol 1 qui se trouve dans le dossier untitled folder 1 :

```
fl.getDocumentDOM().library.selectItem("untitled Folder_1/Symbol_1",  
false);
```

L'exemple suivant désélectionne l'élément symbol 1 contenu dans le dossier untitled folder 1 et ne modifie pas les autres éléments sélectionnés :

```
fl.getDocumentDOM().library.selectItem("untitled Folder_1/Symbol_1", true,  
false);
```

## library.selectNone()

### Disponibilité

Flash MX 2004.

### Utilisation

```
library.selectNone()
```

### Paramètres

Aucun.

### Valeur renvoyée

Aucune.

### Description

Méthode ; désélectionne tous les éléments de la bibliothèque.

### Exemple

Les exemples suivants désélectionnent tous les éléments de la bibliothèque :

```
fl.getDocumentDOM().library.selectNone();  
fl.getDocumentDOM().library.selectAll(false);
```

# library.setItemProperty()

## Disponibilité

Flash MX 2004.

## Utilisation

```
library.setItemProperty(property, value)
```

## Paramètres

*property* Chaîne indiquant le nom de la propriété à modifier. La liste des propriétés est indiquée dans la section [Résumé des propriétés de l'objet Item](#) et dans le résumé des propriétés de ses sous-classes. Pour consulter la liste des objets qui sont des sous-classes de l'objet Item, voir [Résumé de la structure du DOM](#).

*value* Valeur à affecter à la propriété spécifiée.

## Valeur renvoyée

Aucune.

## Description

Méthode ; définit la propriété de tous les objets de bibliothèque sélectionnés (en ignorant les dossiers).

## Exemple

L'exemple suivant affecte la valeur « button » à la propriété `symbolType` des éléments de bibliothèque sélectionnés. Dans ce cas précis, l'élément doit être un [Objet SymbolItem](#) ; en effet, `symbolType` est une propriété valide pour les objets `SymbolItem`.

```
fl.getDocumentDOM().library.setItemProperty("symbolType", "button");
```

# library.updateItem()

## Disponibilité

Flash MX 2004.

## Utilisation

```
library.updateItem([namePath])
```

## Paramètres

*namePath* Chaîne spécifiant le nom de l'élément. Si l'élément se trouve dans un dossier, il est possible d'indiquer son nom et son chemin d'accès en utilisant la notation avec barre oblique. Cette opération est identique à un clic de droite sur un élément puis à la sélection de Mettre à jour dans le menu de l'interface utilisateur. Si aucun nom n'est indiqué, c'est la sélection actuelle qui est actualisée. Ce paramètre est facultatif.

## Valeur renvoyée

Une valeur booléenne : `true` si Flash a pu actualiser l'élément, `false` sinon.

## Description

Méthode ; actualise l'élément de bibliothèque spécifié.

## Exemple

L'exemple suivant affiche une boîte de dialogue qui montre si l'élément actuellement sélectionné a été actualisé (`true`) ou non (`false`):

```
alert(fl.getDocumentDOM().library.updateItem());
```



# Objet Math

## Disponibilité

Flash MX 2004.

## Description

L'objet Math est une propriété en lecture seule de l'objet flash ; voir [fl.Math](#). Cet objet comporte des méthodes qui prennent en charge des opérations mathématiques courantes.

## Résumé des méthodes de l'objet Math

Les méthodes suivantes sont disponibles pour l'objet Math :

Méthode	Description
<a href="#">Math.concatMatrix()</a>	Effectue une concaténation matricielle et renvoie le résultat.
<a href="#">Math.invertMatrix()</a>	Renvoie l'inverse de la matrice spécifiée.
<a href="#">Math.pointDistance()</a>	Calcule la distance entre deux points.

## Math.concatMatrix()

### Disponibilité

Flash MX 2004.

### Utilisation

```
Math.concatMatrix(mat1, mat2)
```

### Paramètres

*mat1* et *mat2* Spécifient les objets de tableau à concaténer (voir [Objet Matrix](#)). Chaque paramètre doit être un objet avec des champs a, b, c, d, tx et ty.

### Valeur renvoyée

Un tableau d'objets concaténés.

### Description

Méthode ; effectue une concaténation de tableaux et renvoie le résultat.

## Exemple

L'exemple suivant enregistre dans la variable `elt` l'objet sélectionné, multiplie le tableau des objets par le tableau d'affichage et enregistre cette valeur dans la variable `mat` :

```
var elt = fl.getDocumentDOM().selection[0];  
var mat = fl.Math.concatMatrix( elt.matrix , fl.getDocumentDOM().viewMatrix );
```

# Math.invertMatrix()

## Disponibilité

Flash MX 2004.

## Utilisation

`Math.invertMatrix(mat)`

## Paramètres

*mat* Indique l'objet Matrix à inverser (consultez la section [Objet Matrix](#)). Il doit comporter les champs suivants : `a`, `b`, `c`, `d`, `tx` et `ty`.

## Valeur renvoyée

Un objet Matrix qui est l'inverse du tableau original.

## Description

Méthode ; renvoie l'inverse de la matrice spécifiée.

## Exemple

L'exemple suivant enregistre dans la variable `elt` l'objet sélectionné, assigne ce tableau à la variable `mat`, et enregistre son inverse dans la variable `inv` :

```
var elt = fl.getDocumentDOM().selection[0];  
var mat = elt.matrix;  
var inv = fl.Math.invertMatrix( mat );
```

# Math.pointDistance()

## Disponibilité

Flash MX 2004.

## Utilisation

`Math.pointDistance(pt1, pt2)`

## Paramètres

*pt1* et *pt2* Spécifient les points entre lesquels la distance doit être mesurée.

## Valeur renvoyée

Une valeur en virgule flottante qui représente la distance entre les points.

## Description

Méthode ; calcule la distance entre deux points.

## Exemple

L'exemple suivant enregistre dans la variable `dist` la distance entre *pt1* et *pt2* :

```
var pt1 = {x:10, y:20}  
var pt2 = {x:100, y:200}  
var dist = fl.Math.pointDistance(pt1, pt2);
```

# Objet Matrix

## Disponibilité

Flash MX 2004.

## Description

L'objet Matrix représente une matrice de transformation.

## Résumé des propriétés de l'objet Matrix

Les propriétés suivantes sont disponibles pour l'objet Matrix :

Propriété	Description
<code>matrix.a</code>	Nombre en virgule flottante spécifiant l'élément (0,0) de la matrice de transformation.
<code>matrix.b</code>	Nombre en virgule flottante spécifiant l'élément (0,1) de la matrice.
<code>matrix.c</code>	Nombre en virgule flottante spécifiant l'élément (1,0) de la matrice.
<code>matrix.d</code>	Nombre en virgule flottante spécifiant l'élément (1,1) de la matrice.
<code>matrix.tx</code>	Nombre en virgule flottante spécifiant l'emplacement de l'axe x du point d'alignement d'un symbole ou du centre d'une forme.
<code>matrix.ty</code>	Nombre en virgule flottante spécifiant l'emplacement de l'axe y du point d'alignement d'un symbole ou du centre d'une forme.

## matrix.a

### Disponibilité

Flash MX 2004.

### Utilisation

`matrix.a`

### Description

Propriété ; nombre en virgule flottante spécifiant l'élément (0,0) de la matrice de transformation. Cette valeur représente le facteur de redimensionnement de l'axe *x* de l'objet.

## Exemple

Les propriétés `a` et `d` d'un tableau représentent son redimensionnement. Dans l'exemple suivant, les valeurs sont respectivement définies comme 2 et 3, afin de redimensionner l'objet sélectionné au double en largeur et au triple en hauteur :

```
var mat = fl.getDocumentDOM().selection[0].matrix;  
mat.a = 2;  
mat.d = 3;  
fl.getDocumentDOM().selection[0].matrix = mat;
```

Il est possible de faire pivoter un objet en définissant les propriétés `a`, `b`, `c` et `d` du tableau relativement entre elles, avec  $a = d$  et  $b = -c$ . Par exemple, des valeurs de 0,5, 0,8, -0,8 et 0,5 provoqueront une rotation de l'objet sur 60°:

```
var mat = fl.getDocumentDOM().selection[0].matrix;  
mat.a = 0.5;  
mat.b = 0.8;  
mat.c = 0.8*(-1);  
mat.d = 0.5;  
fl.getDocumentDOM().selection[0].matrix = mat;
```

Les valeurs  $a = d = 1$  et  $c = b = 0$  restaureront la forme originale de l'objet.

## matrix.b

### Disponibilité

Flash MX 2004.

### Utilisation

`matrix.b`

### Description

Propriété ; nombre en virgule flottante spécifiant l'élément (0,1) de la matrice de transformation. Cette valeur représente l'inclinaison verticale d'une forme ; elle provoque le déplacement du côté droit de la forme le long de l'axe vertical.

Les propriétés `matrix.b` et `matrix.c` d'un tableau représentent son inclinaison (consultez la section [matrix.c](#)).

### Exemple

Dans l'exemple suivant, vous pouvez donner à `b` et `c` les valeurs -1 et 0, respectivement ; ces valeurs inclinent l'objet à un angle vertical de 45 ° :

```
var mat = fl.getDocumentDOM().selection[0].matrix;  
mat.b = -1;  
mat.c = 0;  
fl.getDocumentDOM().selection[0].matrix = mat;
```

Pour restaurer la forme originale de l'objet, donnez à `b` et `c` la valeur 0.

Voir aussi l'exemple [matrix.a](#).

## matrix.c

### Disponibilité

Flash MX 2004.

### Utilisation

`matrix.c`

### Description

Propriété ; nombre en virgule flottante spécifiant l'élément (1,0) de la matrice de transformation. Cette valeur provoque une inclinaison de l'objet en déplaçant son bord inférieur le long d'un axe horizontal.

Les propriétés `matrix.b` et `matrix.c` d'un tableau représentent son inclinaison.

### Exemple

Voir l'exemple relatif à [matrix.b](#).

## matrix.d

### Disponibilité

Flash MX 2004.

### Utilisation

`matrix.d`

### Description

Propriété ; nombre en virgule flottante spécifiant l'élément (1,1) de la matrice de transformation. Cette valeur représente le facteur de redimensionnement de l'axe *y* de l'objet.

### Exemple

Voir l'exemple relatif à [matrix.a](#).

# matrix.tx

## Disponibilité

Flash MX 2004.

## Utilisation

`matrix.tx`

## Description

Propriété ; nombre en virgule flottante spécifiant l'emplacement de l'axe *x* du point d'alignement (également *point d'origine* ou *point zéro*) d'un symbole ou du centre d'une forme. Elle définit la translation de la transformation sur l'axe *x*.

Vous pouvez déplacer un objet en définissant ses propriétés `matrix.tx` et `matrix.ty` (consultez la section [matrix.ty](#)).

## Exemple

Dans l'exemple suivant, les paramètres `tx` et `ty` reçoivent la valeur 0 pour amener le point d'alignement de l'objet au point 0,0 du document :

```
var mat = fl.getDocumentDOM().selection[0].matrix;  
mat.tx = 0;  
mat.ty = 0;  
fl.getDocumentDOM().selection[0].matrix = mat;
```

# matrix.ty

## Disponibilité

Flash MX 2004.

## Utilisation

`matrix.ty`

## Description

Propriété ; nombre en virgule flottante spécifiant l'emplacement de l'axe *y* du point d'alignement d'un symbole ou du centre d'une forme. Elle définit la translation de la transformation sur l'axe *y*.

Vous pouvez déplacer un objet en définissant ses propriétés `matrix.tx` et `matrix.ty`.

## Exemple

Voir l'exemple relatif à [matrix.tx](#).

# Objet outputPanel

## Disponibilité

Flash MX 2004.

## Description

L'objet `outputPanel` représente le panneau de sortie, qui affiche des informations facilitant le débogage, par exemple les erreurs de syntaxe. Vous accédez à cet objet via `fl.outputPanel` (ou `flash.outputPanel`). Voir [fl.outputPanel](#).

## Résumé des méthodes de l'objet outputPanel

L'objet `outputPanel` utilise les méthodes suivantes :

Méthode	Description
<code>outputPanel.clear()</code>	Efface le contenu du panneau Sortie.
<code>outputPanel.save()</code>	Enregistre le contenu du panneau Sortie dans un fichier texte local.
<code>outputPanel.trace()</code>	Ajoute une ligne au contenu du panneau Sortie en la terminant par un code de nouvelle ligne.

## outputPanel.clear()

### Disponibilité

Flash MX 2004.

### Utilisation

```
outputPanel.clear()
```

### Paramètres

Aucun.

### Valeur renvoyée

Aucune.

### Description

Méthode ; efface le contenu du panneau de sortie. Vous pouvez utiliser cette méthode dans une application de traitement par lot pour effacer une liste d'erreurs, ou pour les enregistrer de façon incrémentielle en utilisant cette méthode avec `outputPanel.save()`.



## Exemple

L'exemple suivant efface le contenu actuel du panneau de sortie :

```
fl.outputPanel.clear();
```

## outputPanel.save()

### Disponibilité

Flash MX 2004 ; paramètre *bUseSystemEncoding* intégré à Flash 8.

### Utilisation

```
outputPanel.save(fileURI [, bAppendToFile [, bUseSystemEncoding]])
```

### Paramètres

*fileURI* Chaîne de type « fichier URI » qui spécifie le nom du fichier local qui doit recevoir le contenu du panneau Sortie.

*bAppendToFile* Valeur booléenne facultative. Si elle est réglée sur *true*, le contenu du panneau de sortie est ajouté à la fin du fichier de sortie. Si elle est réglée sur *false*, la méthode écrase le fichier de sortie, s'il existe déjà. La valeur par défaut est *false*.

*bUseSystemEncoding* Valeur booléenne facultative. Si elle vaut *true*, la méthode enregistre le texte du panneau Sortie en utilisant le codage système. Si la valeur est *false*, le texte du panneau de sortie est enregistré en codage UTF-8, avec des caractères d'ordre d'octet au début du texte. La valeur par défaut est *false*.

### Valeur renvoyée

Aucune.

### Description

Méthode ; enregistre le contenu du panneau Sortie dans un fichier texte local, soit en remplaçant le fichier, soit en ajoutant les informations dans le fichier existant.

Si *fileURI* n'est pas spécifié ou n'est pas valide, une erreur est renvoyée.

Cette méthode est pratique pour le traitement par lots. Vous pouvez par exemple créer un fichier JSFL regroupant plusieurs composants compilés. Comme toute éventuelle erreur de compilation sera affichée dans le panneau de sortie, vous pouvez utiliser cette méthode pour enregistrer les erreurs dans un fichier texte, qui sera automatiquement analysé par le système de compilation utilisé.

## Exemple

L'exemple suivant enregistre le contenu du panneau de sortie dans le fichier `batch.log`, qui réside dans le dossier `/tests`, et écrase son contenu le cas échéant :

```
fl.outputPanel.save("file:///c:/tests/batch.log");
```

# outputPanel.trace()

## Disponibilité

Flash MX 2004.

## Utilisation

```
outputPanel.trace(message)
```

## Paramètres

*message*     Chaîne qui contient le texte à ajouter dans le panneau de sortie.

## Valeur renvoyée

Aucune.

## Description

Méthode. Envoie une chaîne de texte au panneau Sortie, suivie d'un code de nouvelle ligne, et affiche le panneau Sortie s'il n'est pas déjà visible. Cette méthode est identique à [fl.trace\(\)](#) et fonctionne comme l'instruction `trace()` en ActionScript.

Pour envoyer une ligne vide, utilisez `outputPanel.trace("")` ou `outputPanel.trace("\n")`. Vous pouvez utiliser cette dernière commande en ligne en intégrant `\n` à la chaîne *message*.

## Exemple

L'exemple suivant affiche plusieurs lignes de texte dans le panneau Sortie :

```
fl.outputPanel.clear();
fl.outputPanel.trace("Hello World!!!");
var myPet = "cat";
fl.outputPanel.trace("\nI have a " + myPet);
fl.outputPanel.trace("");
fl.outputPanel.trace("I love my " + myPet);
fl.outputPanel.trace("Do you have a " + myPet + "?");
```

# Objet Oval

**Héritage** [Objet Element](#) > [Objet Shape](#) > objet Oval

## Disponibilité

Flash CS3 Professional.

## Description

L'objet Oval est une forme tracée à l'aide d'une primitive de type ovale. Pour déterminer si un élément est un objet Oval, utilisez [shape.isOvalObject](#).

## Résumé des propriétés de l'objet Oval

Outre les propriétés de l'[Objet Shape](#), les propriétés suivantes sont également disponibles avec l'objet Oval. Pour définir les propriétés d'un objet Oval, utilisez [document.setOvalObjectProperty\(\)](#).

Propriété	Description
<a href="#">OvalObject.closePath</a>	En lecture seule ; valeur booléenne spécifiant si la case à cocher Fermer le tracé, qui se trouve dans l'inspecteur des propriétés, est sélectionnée ou non.
<a href="#">OvalObject.endAngle</a>	En lecture seule ; valeur codée en virgule flottante qui indique l'angle de fin de l'objet Oval.
<a href="#">OvalObject.innerRadius</a>	En lecture seule ; valeur codée en virgule flottante qui précise, en pourcentage, le rayon interne de l'objet Oval.
<a href="#">OvalObject.startAngle</a>	En lecture seule ; valeur codée en virgule flottante qui indique l'angle de début de l'objet Oval.

## OvalObject.closePath

### Disponibilité

Flash CS3 Professional.

### Utilisation

[OvalObject.closePath](#)

## Description

En lecture seule ; valeur booléenne spécifiant si la case à cocher Fermer le tracé, qui se trouve dans l'inspecteur des propriétés, est sélectionnée ou non. Si les valeurs de l'angle de début et de fin sont identiques pour l'objet, la définition de cette propriété n'a aucune incidence tant que ces valeurs demeurent inchangées.

Pour définir cette valeur, utilisez `document.setOvalObjectProperty()`.

## Exemple

L'exemple suivant désélectionne la propriété `OvalObject.closePath` :

```
fl.getDocumentDOM().setOvalObjectProperty("closePath",false);
```

## Voir aussi

[document.setOvalObjectProperty\(\)](#), [shape.isOvalObject](#)

# OvalObject.endAngle

## Disponibilité

Flash CS3 Professional.

## Utilisation

`OvalObject.endAngle`

## Description

Propriété en lecture seule ; valeur codée en virgule flottante qui indique l'angle de fin de l'objet Oval. Les valeurs acceptées sont comprises entre 0 et 360.

Pour définir cette valeur, utilisez `document.setOvalObjectProperty()`.

## Exemple

L'exemple suivant définit sur 270 l'angle de fin des objets Oval sélectionnés :

```
fl.getDocumentDOM().setOvalObjectProperty("endAngle",270);
```

## Voir aussi

[document.setOvalObjectProperty\(\)](#), [OvalObject.startAngle](#), [shape.isOvalObject](#)

# OvalObject.innerRadius

## Disponibilité

Flash CS3 Professional.

## Utilisation

`OvalObject.innerRadius`

## Description

Propriétés en lecture seule ; valeur codée en virgule flottante qui précise, en pourcentage, le rayon interne de l'objet Oval. Les valeurs acceptées sont comprises entre 0 et 99.

Pour définir cette valeur, utilisez `document.setOvalObjectProperty()`.

## Exemple

L'exemple suivant définit sur 50 % le rayon interne des objets Oval sélectionnés :

```
fl.getDocumentDOM().setOvalObjectProperty("innerRadius",50);
```

## Voir aussi

`document.setOvalObjectProperty()`, `shape.isOvalObject`

# OvalObject.startAngle

## Disponibilité

Flash CS3 Professional.

## Utilisation

`OvalObject.startAngle`

## Description

Propriété en lecture seule ; valeur codée en virgule flottante qui indique l'angle de début de l'objet Oval. Les valeurs acceptées sont comprises entre 0 et 360.

Pour définir cette valeur, utilisez `document.setOvalObjectProperty()`.

## Exemple

L'exemple suivant définit sur 270 l'angle de début des objets Oval sélectionnés :

```
fl.getDocumentDOM().setOvalObjectProperty("startAngle",270);
```

## Voir aussi

`document.setOvalObjectProperty()`, `OvalObject.endAngle`, `shape.isOvalObject`

# Objet Parameter

## Disponibilité

Flash MX 2004.

## Description

Le type d'objet `Parameter` figure dans le tableau `screen.parameters` (qui correspond à la fenêtre de l'inspecteur des propriétés dans l'environnement de programmation Flash) ou le tableau `componentInstance.parameters` (qui correspond au composant Inspecteur des propriétés dans l'environnement de programmation). Consultez les sections [screen.parameters](#) et [componentInstance.parameters](#).

## Résumé des méthodes de l'objet Parameter

Les méthodes suivantes sont disponibles pour l'objet `Parameter` :

Méthode	Description
<code>parameter.insertItem()</code>	Insère un élément dans une liste, un objet ou un tableau.
<code>parameter.removeItem()</code>	Supprime un élément du type liste, objet ou tableau d'un paramètre d'un écran ou d'un composant.

## Résumé des propriétés de l'objet Parameter

Les propriétés suivantes sont disponibles pour l'objet `Parameter` :

Propriété	Description
<code>parameter.category</code>	Chaîne indiquant la propriété <code>category</code> pour le paramètre <code>screen</code> ou <code>componentInstance</code> .
<code>parameter.listIndex</code>	Entier spécifiant la valeur de l'élément de liste sélectionné.
<code>parameter.name</code>	Propriété en lecture seule ; chaîne indiquant le nom du paramètre.
<code>parameter.value</code>	Correspond au champ Valeur de l'onglet Paramètres du panneau Inspecteur de composants, de l'onglet Paramètres de l'inspecteur des propriétés ou de la fenêtre Inspecteur des propriétés.
<code>parameter.valueType</code>	Propriété en lecture seule ; chaîne indiquant le type de paramètre d'écran ou de composant.
<code>parameter.verbose</code>	Détermine où le paramètre est affiché.

## parameter.category

### Disponibilité

Flash MX 2004.

### Utilisation

`parameter.category`

### Description

Propriété ; chaîne indiquant la propriété `category` pour le paramètre `screen` ou `componentInstance`. Cette propriété représente une autre méthode pour présenter une liste de paramètres. Cette fonctionnalité n'est pas disponible dans l'interface utilisateur de Flash.

## parameter.insertItem()

### Disponibilité

Flash MX 2004.

### Utilisation

`parameter.insertItem(index, name, value, type)`

### Paramètres

*index* Indice (en base zéro) sur un entier, qui indique si l'élément sera inséré dans la liste, l'objet ou le tableau. Si l'indice est 0, l'élément est inséré au début de la liste. Si l'indice est supérieur à la taille de la liste, l'élément est inséré à la fin du tableau.

*name* Chaîne spécifiant le nom de l'élément à insérer. Ce paramètre est nécessaire pour les paramètres d'objets.

*value* Chaîne spécifiant la valeur de l'élément à insérer.

*type* Chaîne spécifiant le type d'élément à insérer.

### Valeur renvoyée

Aucune.

### Description

Méthode ; insère un élément dans une liste, un objet ou un tableau. Si l'un des paramètres est une liste, un objet ou un tableau, la propriété *value* est un tableau.

## Exemple

L'exemple suivant insère la valeur de "New Value" dans le paramètre `labelPlacement` :

```
// Sélectionner une occurrence d'un composant Bouton sur la scène.  
var parms = fl.getDocumentDOM().selection[0].parameters;  
parms[2].insertItem(0, "name", "New Value", "String");  
var values = parms[2].value;  
for(var prop in values){  
    fl.trace("labelPlacement parameter value = " + values[prop].value);  
}
```

## parameter.listIndex

### Disponibilité

Flash MX 2004.

### Utilisation

`parameter.listIndex`

### Description

Propriété ; valeur de l'élément de liste sélectionné. Cette propriété n'est valide que si le paramètre `valueType` a pour valeur "List".

## Exemple

L'exemple suivant définit le premier paramètre d'un objet diapositive, c'est-à-dire le paramètre `autoKeyNav`. Pour régler le paramètre sur l'une des valeurs gérées (`true`, `false` ou `inherit`) `parameter.listIndex` reçoit la valeur de l'indice de l'élément dans la liste (0 pour `true`, 1 pour `false`, 2 pour `inherit`).

```
var parms = fl.getDocumentDOM().screenOutline.screens[1].parameters;  
parms[0].listIndex = 1;
```

## parameter.name

### Disponibilité

Flash MX 2004.

### Utilisation

`parameter.name`

### Description

Propriété en lecture seule ; chaîne indiquant le nom du paramètre.



## Exemple

L'exemple suivant affiche le nom du cinquième paramètre du composant sélectionné :

```
var parms = fl.getDocumentDOM().selection[0].parameters;  
fl.trace("name: " + parms[4].name);
```

L'exemple suivant affiche le nom du cinquième paramètre de l'écran spécifié :

```
var parms = fl.getDocumentDOM().screenOutline.screens[1].parameters;  
fl.trace("name: " + parms[4].name);
```

# parameter.removeItem()

## Disponibilité

Flash MX 2004.

## Utilisation

```
parameter.removeItem(index)
```

## Paramètres

*index* L'indice (en base zéro) sur un entier de l'élément à supprimer de la propriété d'écran ou de composant.

## Valeur renvoyée

Aucune.

## Description

Méthode ; supprime un élément du type liste, objet ou tableau d'un paramètre d'un écran ou d'un composant.

## Exemple

L'exemple suivant supprime l'élément à l'index 1 depuis le paramètre `labelPlacement` d'un composant :

```
// Sélectionner une occurrence d'un composant Button sur la scène.  
var parms = fl.getDocumentDOM().selection[0].parameters;  
var values = parms[2].value;  
fl.trace("--Original--");  
for(var prop in values){  
    fl.trace("labelPlacement value = " + values[prop].value);  
}  
parms[2].removeItem(1);  
  
var newValues = parms[2].value;  
fl.trace("--After Removing Item--");  
for(var prop in newValues){  
    fl.trace("labelPlacement value = " + newValues[prop].value);  
}
```

L'exemple suivant supprime l'élément à l'index 1 depuis le paramètre `autoKeyNav` d'un écran :

```
// Ouvrir un document de type Présentation.  
var parms = fl.getDocumentDOM().screenOutline.screens[1].parameters;  
var values = parms[0].value;  
fl.trace("--Original--");  
for(var prop in values){  
    fl.trace("autoKeyNav value = " + values[prop].value);  
}  
parms[0].removeItem(1);  
  
var newValues = parms[0].value;  
fl.trace("--After Removing Item--");  
for(var prop in newValues){  
    fl.trace("autoKeyNav value = " + newValues[prop].value);  
}
```

## parameter.value

### Disponibilité

Flash MX 2004.

### Utilisation

`parameter.value`

### Description

Propriété ; correspond au champ Valeur de l'onglet Paramètres du panneau Inspecteur de composants, de l'onglet Paramètres de l'inspecteur des propriétés ou de la fenêtre Inspecteur des propriétés. Le type de la propriété `value` est déterminé par la propriété `valueType` du paramètre (consultez la section [parameter.valueType](#)).

# parameter.valueType

## Disponibilité

Flash MX 2004.

## Utilisation

`parameter.valueType`

## Description

Propriété en lecture seule ; chaîne indiquant le type de paramètre d'écran ou de composant. Ce type peut avoir l'une des valeurs suivantes : "Default" (par défaut), "Array" (tableau), "Object" (objet), "List" (liste), "String" (chaîne), "Number" (nombre), "Boolean" (booléen), "Font Name" (nom de police de caractères), "Color" (couleur), "Collection", "Web Service URL" (adresse de service Web) ou "Web Service Operation" (opération de service web).

## Voir aussi

[parameter.value](#)

# parameter.verbose

## Disponibilité

Flash MX 2004.

## Utilisation

`parameter.verbose`

## Description

Propriété ; détermine où le paramètre est affiché. Si la valeur de cette propriété est 0 (non verbeux), le paramètre est uniquement affiché dans l'inspecteur des composants. Si elle vaut 1 (verbeux), le paramètre est affiché dans l'inspecteur des composants et dans l'onglet Paramètres de l'inspecteur des propriétés.

# Objet Path

## Disponibilité

Flash MX 2004.

## Description

L'objet Path définit une séquence de segments de ligne (droite, courbe ou les deux), qui est particulièrement destinée à la création d'outils extensibles. L'exemple suivant montre le renvoi d'une instance d'un objet Path par l'objet flash :

```
path = fl.drawingLayer.newPath();
```

Voir aussi l'[Objet drawingLayer](#).

## Résumé des méthodes de l'objet Path

Les méthodes suivantes sont disponibles pour l'objet Path :

Méthode	Description
<code>path.addCubicCurve()</code>	Ajoute un segment de courbe de Bézier cubique au trajet.
<code>path.addCurve()</code>	Ajoute un segment de courbe de Bézier quadratique au trajet.
<code>path.addPoint()</code>	Ajoute un point au trajet.
<code>path.clear()</code>	Supprime tous les points du trajet.
<code>path.close()</code>	Ajoute un point à l'emplacement du premier point du trajet, puis étend le trajet jusqu'à ce point, ce qui ferme le trajet.
<code>path.makeShape()</code>	Crée une forme sur la scène en utilisant les paramètres de trait et de remplissage actuels.
<code>path.newContour()</code>	Ajoute le début d'un nouveau contour au trajet.

## Résumé des propriétés de l'objet Path

Les propriétés suivantes sont disponibles pour l'objet Path :

Propriété	Description
<code>path.nPts</code>	En lecture seule ; entier représentant le nombre de points du trajet.

## path.addCubicCurve()

### Disponibilité

Flash MX 2004.

### Utilisation

```
path.addCubicCurve(xAnchor, yAnchor, x2, y2, x3, y3, x4, y4)
```

### Paramètres

*xAnchor* Valeur en virgule flottante indiquant la coordonnée *x* du premier point de contrôle.

*yAnchor* Valeur en virgule flottante indiquant la coordonnée *y* du premier point de contrôle.

*x2* Valeur en virgule flottante indiquant la coordonnée *x* du second point de contrôle.

*y2* Valeur en virgule flottante indiquant la coordonnée *y* du second point de contrôle.

*x3* Valeur en virgule flottante indiquant la coordonnée *x* du troisième point de contrôle.

*y3* Valeur en virgule flottante indiquant la coordonnée *y* du troisième point de contrôle.

*x4* Valeur en virgule flottante indiquant la coordonnée *x* du quatrième point de contrôle.

*y4* Valeur en virgule flottante indiquant la coordonnée *y* du quatrième point de contrôle.

### Valeur renvoyée

Aucune.

### Description

Méthode ; ajoute un segment de courbe de Bézier cubique au trajet.

### Exemple

L'exemple suivant crée un nouveau trajet, l'enregistre dans la variable `myPath`, puis affecte la courbe au trajet :

```
var myPath = fl.drawingLayer.newPath();  
myPath.addCubicCurve(0, 0, 10, 20, 20, 20, 30, 0);
```

## path.addCurve()

### Disponibilité

Flash MX 2004.

### Utilisation

```
path.addCurve(xAnchor, yAnchor, x2, y2, x3, y3)
```

## Paramètres

- xAnchor* Valeur en virgule flottante indiquant la coordonnée *x* du premier point de contrôle.
- yAnchor* Valeur en virgule flottante indiquant la coordonnée *y* du premier point de contrôle.
- x2* Valeur en virgule flottante indiquant la coordonnée *x* du second point de contrôle.
- y2* Valeur en virgule flottante indiquant la coordonnée *y* du second point de contrôle.
- x3* Valeur en virgule flottante indiquant la coordonnée *x* du troisième point de contrôle.
- y3* Valeur en virgule flottante indiquant la coordonnée *y* du troisième point de contrôle.

## Valeur renvoyée

Aucune.

## Description

Méthode ; ajoute un segment de courbe de Bézier quadratique au trajet.

## Exemple

L'exemple suivant crée un nouveau trajet, l'enregistre dans la variable `myPath`, puis affecte la courbe au trajet :

```
var myPath = fl.drawingLayer.newPath();  
myPath.addCurve(0, 0, 10, 20, 20, 0);
```

# path.addPoint()

## Disponibilité

Flash MX 2004.

## Utilisation

```
path.addPoint(x, y)
```

## Paramètres

- x* Nombre en virgule flottante indiquant la coordonnée *x* du point.
- y* Nombre en virgule flottante indiquant la coordonnée *y* du point.

## Valeur renvoyée

Aucune.

## Description

Méthode ; ajoute un point au trajet.

## Exemple

L'exemple suivant crée un nouveau trajet, l'enregistre dans la variable `myPath`, puis affecte le nouveau point au trajet :

```
var myPath = fl.drawingLayer.newPath();  
myPath.addPoint(10, 100);
```

# path.clear()

## Disponibilité

Flash MX 2004.

## Utilisation

```
path.clear()
```

## Paramètres

Aucun.

## Valeur renvoyée

Aucune.

## Description

Méthode ; supprime tous les points du trajet.

## Exemple

L'exemple suivant supprime tous les points d'un trajet enregistré dans la variable `myPath` :

```
var myPath = fl.drawingLayer.newPath();  
myPath.clear();
```

# path.close()

## Disponibilité

Flash MX 2004.

## Utilisation

```
path.close()
```

## Paramètres

Aucun.

## Valeur renvoyée

Aucune.

## Description

Méthode ; ajoute un point à l'emplacement du premier point du trajet, puis étend le trajet jusqu'à ce point, ce qui ferme le trajet. Si le trajet ne comporte pas de points, aucun point n'est ajouté.

## Exemple

L'exemple suivant crée un trajet fermé :

```
var myPath = fl.drawingLayer.newPath();  
myPath.close();
```

# path.makeShape()

## Disponibilité

Flash MX 2004.

## Utilisation

```
path.makeShape([bSupressFill [, bSupressStroke]])
```

## Paramètres

*bSupressFill* Valeur booléenne. Si elle est `true`, la méthode supprime le remplissage qui devrait être appliqué à la forme. La valeur par défaut est `false`. Ce paramètre est facultatif.

*bSupressStroke* Valeur booléenne. Si elle est `true`, la méthode supprime le trait qui devrait être appliqué à la forme. La valeur par défaut est `false`. Ce paramètre est facultatif.

## Valeur renvoyée

Aucune.

## Description

Méthode ; crée une forme sur la scène en utilisant les paramètres de trait et de remplissage actuels. Le trajet est effacé après la création de la forme. Cette méthode comporte deux paramètres facultatifs permettant de supprimer le remplissage et le trait de l'objet forme ainsi obtenu. Si ces paramètres sont omis ou reçoivent la valeur `false`, les valeurs actuelles de remplissage et de trait sont utilisées.

## Exemple

L'exemple suivant crée une forme dotée du remplissage actuel, mais sans trait :

```
var myPath = fl.drawingLayer.newPath();  
myPath.makeShape(false, true);
```



# path.newContour()

## Disponibilité

Flash MX 2004.

## Utilisation

```
path.newContour()
```

## Paramètres

Aucun.

## Valeur renvoyée

Aucune.

## Description

Méthode ; ajoute le début d'un nouveau contour au trajet.

## Exemple

L'exemple suivant crée un carré vide :

```
var myPath = fl.drawingLayer.newPath();
myPath.addPoint(0, 0);
myPath.addPoint(0, 30);
myPath.addPoint(30, 30);
myPath.addPoint(30, 0);
myPath.addPoint(0, 0);

myPath.newContour();
myPath.addPoint(10, 10);
myPath.addPoint(10, 20);
myPath.addPoint(20, 20);
myPath.addPoint(20, 10);
myPath.addPoint(10, 10);

myPath.makeShape();
```

# path.nPts

## Disponibilité

Flash MX 2004.

## Utilisation

path.nPts

## Description

Propriété en lecture seule ; entier représentant le nombre de points du trajet. Un nouveau trajet compte 0 points.

## Exemple

L'exemple suivant affiche dans le panneau de sortie le nombre de points du trajet référencé par la variable myPath :

```
var myPath = fl.drawingLayer.newPath();
var numOfPoints = myPath.nPts;
fl.trace("Number of points in the path: " + numOfPoints);
// Affiche : Nombre de points du trajet : 0
```

# Objet Project

## Disponibilité

Flash 8.

## Description

L'objet Project représente un fichier de projet Flash (FLP). Vous pouvez utiliser les commandes suivantes pour renvoyer un objet Project :

- Pour créer un nouveau fichier de projet, utilisez `fl.createProject()`.
- Pour ouvrir un fichier de projet existant, utilisez `fl.openProject()`.
- Pour renvoyer un objet Project pour le projet actuellement ouvert, utilisez `fl.getProject()`.

## Résumé des méthodes de l'objet Project

Vous pouvez utiliser les méthodes suivantes avec l'objet Project :

Méthode	Description
<code>project.addFile()</code>	Ajoute le fichier spécifié au projet.
<code>project.canPublishProject()</code>	Détermine si le projet peut être publié.
<code>project.canTestProject()</code>	Détermine si le projet peut être testé.
<code>project.findProjectItem()</code>	Recherche le fichier spécifié dans le projet.
<code>project.publishProject()</code>	Publie les fichiers FLA d'un projet.
<code>project.testProject()</code>	Teste le projet.

## Résumé des propriétés de l'objet Project

Vous pouvez utiliser les propriétés suivantes avec l'objet Project :

Propriété	Description
<code>project.defaultItem</code>	Spécifie l' <a href="#">Objet ProjectItem</a> qui représente le document par défaut du projet.
<code>project.items</code>	Propriété en lecture seule ; tableau des objets ProjectItem (consultez la section <a href="#">Objet ProjectItem</a> ) contenus dans le projet.
<code>project.name</code>	Nom du projet tel qu'il apparaît dans le panneau Projet.
<code>project.projectURI</code>	Propriété en lecture seule ; chaîne de type « fichier URI » représentant le chemin d'accès et le nom du fichier de projet.

# project.addFile()

## Disponibilité

Flash 8.

## Utilisation

```
project.addFile(fileURI [ , autoCreateFolder])
```

## Paramètres

*fileURI* Chaîne de type « fichier URI » qui indique le fichier à ajouter au projet.

*autoCreateFolder* Valeur booléenne facultative qui spécifie si des dossiers doivent être automatiquement créés dans le panneau Project pour matérialiser le chemin d'accès indiqué dans *fileURI*. La valeur par défaut correspond à *false*.

## Valeur renvoyée

Si l'opération réussit, la méthode renvoie un objet `ProjectItem`. Dans le cas contraire, elle renvoie la valeur `undefined`. Voir [Objet ProjectItem](#).

## Description

Méthode ; ajoute le fichier spécifié au projet. Il est possible d'utiliser le paramètre *autoCreateFolder* pour déterminer l'emplacement du nouveau fichier dans le panneau Projet :

- Si le paramètre *autoCreateFolder* est omis ou s'il reçoit la valeur *false*, le fichier est ajouté au niveau racine du projet.
- Si vous passez la valeur *true* pour *autoCreateFolder*, et si *fileURI* se trouve à un niveau inférieur à celui du fichier FLP dans la structure hiérarchique du disque, la structure de fichiers est reproduite dans le panneau Project. En d'autres termes, de nouveaux dossiers sont ajoutés dans le panneau Project si nécessaire, afin de refléter l'emplacement du fichier sur le disque dur.
- Si vous passez la valeur *true* pour *autoCreateFolder*, et si *fileURI* se trouve à un niveau supérieur à celui du fichier FLP dans la structure hiérarchique du disque, le fichier est ajouté au niveau racine. En d'autres termes, *autoCreateFolder* n'est pas pris en compte.

## Exemple

L'exemple ci-dessous illustre diverses façons d'utiliser cette commande : Dans cet exemple, le fichier de projet ouvert se trouve dans le répertoire c:\Projects, et les seuls fichiers figurant actuellement dans le projet ont été ajoutés au niveau racine.

```
// Lire l'objet Project.
var myProject = fl.getProject();

// Cette commande crée un dossier nommé « files » sous le niveau racine
// du projet, et place le fichier myFile.fla dans ce dossier.
var newFile = myProject.addFile("file:///C|Projects/files/myFile.fla",
    true)
fl.trace(newFile.isMissing); // false

// Les deux commandes suivantes produisent le même effet : myFile_02.fla
// est placé au niveau racine du projet.
var newFile = myProject.addFile("file:///C|Projects/files/myFile_02.fla" ,
    false)
var newFile = myProject.addFile("file:///C|Projects/files/myFile_02.fla")
fl.trace(newFile.isMissing); // false

// La commande suivante place myFile_03 comme fichier manquant au
// niveau racine du projet.
var newFile = myProject.addFile("file:///C|myFile_03.fla")
fl.trace(newFile.isMissing); // true
```

L'exemple suivant tente d'ajouter un nouveau fichier au projet et affiche, dans le panneau de sortie, un message indiquant la réussite ou l'échec de cette opération :

```
var myProject = fl.getProject();
var newItem = myProject.addFile("file:///C|Projects/files/Integra.fla",
    true);
fl.trace( "Item " + ( newItem ? "was" : "was not" ) + " added!" );
```

## Voir aussi

[fl.getProject\(\)](#), [project.items](#), [Objet ProjectItem](#)

# project.canPublishProject()

## Disponibilité

Flash 8.

## Utilisation

```
project.canPublishProject()
```

## Paramètres

Aucun.

### Valeur renvoyée

Une valeur booléenne indiquant si le projet peut être publié.

### Description

Méthode ; détermine si le projet peut être publié. Un projet peut être publié s'il contient au moins un fichier FLA.

### Exemple

L'exemple suivant affiche un message dans le panneau de sortie s'il est impossible de publier le projet.

```
if (!fl.getProject().canPublishProject()) {  
    fl.trace("Project cannot be published!");  
}
```

### Voir aussi

[fl.getProject\(\)](#), [project.publishProject\(\)](#), [projectItem.canPublish\(\)](#)

## project.canTestProject()

### Disponibilité

Flash 8.

### Utilisation

```
project.canTestProject()
```

### Paramètres

Aucun.

### Valeur renvoyée

Une valeur booléenne indiquant si le projet peut être testé.

### Description

Méthode ; détermine si le projet peut être testé. Un projet peut être testé si un document par défaut a été spécifié.

### Exemple

L'exemple suivant affiche un message dans le panneau de sortie s'il est impossible de tester le projet :

```
if (!fl.getProject().canTestProject()) {  
    fl.trace("Project cannot be tested!");  
}
```

### Voir aussi

[fl.getProject\(\)](#), [project.testProject\(\)](#), [projectItem.canTest\(\)](#)

## project.defaultItem

### Disponibilité

Flash 8.

### Utilisation

`project.defaultItem`

### Description

Propriété ; spécifie l'objet `ProjectItem` qui représente le document par défaut du projet. Vous devez spécifier un élément par défaut si vous souhaitez tester le projet. Voir [Objet ProjectItem](#).

### Exemple

L'exemple suivant définit le fichier `Flower.fl` comme document par défaut du projet :

```
var myProject = fl.getProject();
var item = myProject.findProjectItem("file:///C:/Projects/files/
    Flower.fl");
fl.myProject.defaultItem = item;
```

L'exemple suivant affiche dans le panneau Sortie le nom du document par défaut :

```
fl.trace(fl.getProject().defaultItem.displayName);
```

### Voir aussi

[fl.getProject\(\)](#), [project.findProjectItem\(\)](#), [Objet ProjectItem](#)

## project.findProjectItem()

### Disponibilité

Flash 8.

### Utilisation

`project.findProjectItem(fileURI)`

### Paramètres

*fileURI* Chaîne de type « fichier URI » qui indique le fichier à rechercher dans le projet.

## Valeur renvoyée

Un objet `ProjectItem` pour l'élément en cas de réussite, `false` dans le cas contraire. Voir [Objet ProjectItem](#).

## Description

Méthode ; recherche le fichier spécifié dans le projet.

## Exemple

L'exemple suivant affiche un message d'erreur dans le panneau de sortie si le fichier spécifié n'est pas trouvé dans le projet :

```
var myProject = fl.getProject();
var item = myProject.findProjectItem("file:///C|/Projects/files/
  Integra fla");
if (item == undefined) {
  fl.trace("Integra fla is missing!");
}
```

## Voir aussi

[fl.getProject\(\)](#), [Objet ProjectItem](#), [projectItem.isMissing](#)

# project.items

## Disponibilité

Flash 8.

## Utilisation

`project.items`

## Description

Propriété en lecture seule ; tableau des objets `ProjectItem` (voir [Objet ProjectItem](#)) contenus dans le projet.

## Exemple

L'exemple suivant affiche les noms de tous les éléments du projet. Les noms sont affichés dans le panneau de sortie.

```
for (i = 0; i < fl.getProject().items.length; i++) {
  fl.trace(fl.getProject().items[i].displayName);
}
```

## Voir aussi

[fl.getProject\(\)](#), [Objet ProjectItem](#)



## project.name

### Disponibilité

Flash 8.

### Utilisation

`project.name`

### Description

Propriété ; le nom du projet tel qu'il apparaît dans le panneau Projet.

### Exemple

L'exemple suivant spécifie un nouveau nom à afficher dans le panneau Projet :

```
fl.getProject().name = "New project name";
```

### Voir aussi

[fl.getProject\(\), project.projectURI](#)

## project.projectURI

### Disponibilité

Flash 8.

### Utilisation

`project.projectURI`

### Description

Propriété en lecture seule ; chaîne de type « fichier URI » qui représente le chemin d'accès et le nom du fichier de projet.

### Exemple

L'exemple suivant affiche le chemin d'accès et le nom du projet actuellement ouvert. Ce nom et ce chemin d'accès sont affichés dans le panneau Sortie.

```
fl.trace("Project is located at: " + fl.getProject().projectURI);
```

### Voir aussi

[fl.getProject\(\), project.name](#)

# project.publishProject()

## Disponibilité

Flash 8.

## Utilisation

```
project.publishProject()
```

## Paramètres

Aucun.

## Valeur renvoyée

Une valeur booléenne indiquant si le projet a été publié.

## Description

Méthode ; publie les fichiers FLA d'un projet.

## Exemple

L'exemple suivant publie le projet après avoir vérifié qu'il peut être publié, puis indique dans le panneau de sortie si le projet a bien été publié :

```
if (fl.getProject().canPublishProject()) {  
    var bSucceeded = fl.getProject().publishProject();  
}  
fl.trace(bSucceeded);
```

## Voir aussi

[fl.getProject\(\)](#), [project.canPublishProject\(\)](#), [projectItem.publish\(\)](#)

# project.testProject()

## Disponibilité

Flash 8.

## Utilisation

```
project.testProject()
```

## Paramètres

Aucun.

## Valeur renvoyée

Une valeur booléenne indiquant si le projet a été testé.

## Description

Méthode ; teste le projet. Pour qu'il soit possible de le tester, un projet doit comporter un document par défaut.

## Exemple

L'exemple suivant teste le projet après avoir vérifié qu'il peut être testé, puis indique dans le panneau de sortie si le projet a bien été testé :

```
if (fl.getProject().canTestProject()) {  
    var bSucceeded = fl.getProject().testProject();  
}  
fl.trace(bSucceeded);
```

## Voir aussi

[fl.getProject\(\)](#), [project.canTestProject\(\)](#), [project.defaultItem](#),  
[projectItem.test\(\)](#)

# Objet ProjectItem

## Disponibilité

Flash 8.

## Description

L'objet `ProjectItem` représente un élément (fichier sur disque) qui a été ajouté à un projet.

Cet objet est une propriété de l'objet `Project` (consultez la section [project.items](#)).

Vous pouvez utiliser les commandes suivantes pour renvoyer un objet `ProjectItem` :

- Pour ajouter un nouveau fichier à un projet, utilisez `project.addItem()`.
- Pour localiser un élément qui a déjà été ajouté à un projet, utilisez `project.findProjectItem()`.

## Résumé des méthodes de l'objet ProjectItem

Vous pouvez utiliser les méthodes suivantes avec l'objet `ProjectItem` :

Méthode	Description
<code>projectItem.canPublish()</code>	Détermine si un élément de projet peut être publié.
<code>projectItem.canTest()</code>	Détermine si un élément de projet peut être testé.
<code>projectItem.publish()</code>	Publie un élément de projet.
<code>projectItem.test()</code>	Teste un élément de projet.

## Résumé des propriétés de l'objet ProjectItem

Vous pouvez utiliser les propriétés suivantes avec l'objet `ProjectItem` :

Propriété	Description
<code>projectItem.displayName</code>	Propriété en lecture seule ; chaîne indiquant le nom d'un élément de projet.
<code>projectItem.isMissing</code>	Propriété en lecture seule ; valeur booléenne qui indique si un fichier est introuvable sur le disque.
<code>projectItem.itemURI</code>	Propriété en lecture seule ; chaîne indiquant le chemin et le nom de l'élément de projet.
<code>projectItem.publishProfile</code>	Chaîne qui spécifie le profil de publication à utiliser lors de la publication d'un élément de projet (fichier FLA).

## projectItem.canPublish()

### Disponibilité

Flash 8.

### Utilisation

```
projectItem.canPublish()
```

### Paramètres

Aucun.

### Valeur renvoyée

Valeur booléenne indiquant si un élément de projet peut être publié.

### Description

Méthode ; détermine si un élément peut être publié. Un élément ne peut être publié que s'il s'agit d'un fichier FLA.

### Exemple

L'exemple suivant affiche un message dans le panneau de sortie s'il est impossible de publier le premier élément du projet :

```
var item = fl.getProject().items[0];
if (!item.canPublish()) {
    fl.trace(item.displayName + " cannot be published!");
}
```

### Voir aussi

[fl.getProject\(\)](#), [project.canPublishProject\(\)](#), [project.items](#),  
[projectItem.publish\(\)](#)

## projectItem.canTest()

### Disponibilité

Flash 8.

### Utilisation

```
projectItem.canTest()
```

### Paramètres

Aucun.

## Valeur renvoyée

Valeur booléenne indiquant si un élément de projet peut être testé.

## Description

Méthode ; détermine si un élément peut être testé. Un élément ne peut être testé que s'il s'agit d'un fichier FLA ou HTML.

## Exemple

L'exemple suivant affiche un message dans le panneau de sortie s'il est impossible de tester le premier élément du projet :

```
var item = fl.getProject().items[0];
if (!item.canTest()) {
    fl.trace(item.name + " cannot be tested!");
}
```

## Voir aussi

[fl.getProject\(\)](#), [project.canTestProject\(\)](#), [project.items](#), [projectItem.test\(\)](#)

# projectItem.displayName

## Disponibilité

Flash 8.

## Utilisation

`projectItem.displayName`

## Description

Propriété en lecture seule ; chaîne spécifiant le nom d'un élément de projet, tel que « fichier.fl ».

## Exemple

L'exemple suivant affiche dans le panneau de sortie les noms de tous les fichiers du projet :

```
fl.trace( "These are all the files in the project: ");
var files = fl.getProject().items;
for (i = 0; i < files.length; i++) {
    fl.trace(files[i].displayName + " ");
}
```

## Voir aussi

[fl.getProject\(\)](#), [project.items](#), [projectItem.itemURI](#)

# projectItem.isMissing

## Disponibilité

Flash 8.

## Utilisation

`projectItem.isMissing`

## Description

Propriété en lecture seule ; valeur booléenne spécifiant si un fichier est absent du disque (par exemple, si cet élément a été déplacé, supprimé ou renommé).

## Exemple

L'exemple suivant affiche dans le panneau de sortie un message qui précise si un fichier particulier se trouve dans le dossier prévu sur le disque :

```
var item = fl.getProject().findProjectItem("file:///C:/Projects/files/
    DynamicHighAscii.fla");
fl.trace("DynamicHighAscii.fla is missing: " + item.isMissing);
```

## Voir aussi

[fl.getProject\(\)](#), [project.findProjectItem\(\)](#), [project.items](#)

# projectItem.itemURI

## Disponibilité

Flash 8.

## Utilisation

`projectItem.itemURI`

## Description

Propriété en lecture seule ; chaîne de type « fichier URI » qui représente le chemin d'accès et le nom de l'élément de projet. Les éléments de dossier contiennent une chaîne vide ("").

## Exemple

L'exemple suivant affiche dans le panneau de sortie le chemin d'accès et le nom de chaque élément du projet :

```
files = fl.getProject().items;
for (i = 0; i < files.length; i++) {
    fl.trace(files[i].itemURI);
}
```

## Voir aussi

[fl.getProject\(\)](#), [projectItem.displayName](#), [project.items](#)

# projectItem.publish()

## Disponibilité

Flash 8.

## Utilisation

```
projectItem.publish()
```

## Paramètres

Aucun.

## Valeur renvoyée

Valeur booléenne `true` si l'opération a réussi, ou `false` sinon.

## Description

Méthode ; publie un élément de projet. Seuls les fichiers FLA peuvent être publiés.

## Exemple

L'exemple suivant publie tous les éléments publiables du projet :

```
for (var i in fl.getProject().items) {  
    var item = fl.getProject().items[i];  
    if (item.canPublish()) {  
        item.publish();  
    }  
}
```

## Voir aussi

[fl.getProject\(\)](#), [project.canPublishProject\(\)](#), [project.items](#),  
[projectItem.canPublish\(\)](#), [projectItem.publishProfile](#)



# projectItem.publishProfile

## Disponibilité

Flash 8.

## Utilisation

`projectItem.publishProfile`

## Description

Propriété ; chaîne qui spécifie le profil de publication à utiliser lors de la publication d'un élément de projet (fichier FLA). Le profil de publication doit être un profil existant dans l'élément, faute de quoi un appel ultérieur à `projectItem.publish()` échouera.

Voir [projectItem.publish\(\)](#).

Si l'élément n'est pas un fichier FLA, cette propriété est une chaîne vide ("") et il sera impossible de lui donner une valeur.

## Exemple

L'exemple suivant définit, sur la base d'un profil spécifié qui existe déjà dans l'élément, le profil de publication de tous les éléments du projet, puis publie chaque élément. Si le profil n'existe pas dans un fichier, ce fichier n'est pas publié.

```
var items = fl.getProject().items;
for ( i = 0 ; i < items.length ; i++ ) {
    items[i].publishProfile = "mySpecialProfile";
    items[i].publish();
}
```

## Voir aussi

[fl.getProject\(\)](#), [project.canPublishProject\(\)](#), [project.items](#),  
[projectItem.canPublish\(\)](#), [projectItem.publish\(\)](#)

# projectItem.test()

## Disponibilité

Flash 8.

## Utilisation

```
projectItem.test()
```

## Paramètres

Aucun.

## Valeur renvoyée

Une valeur booléenne indiquant si l'élément a été testé ou non.

## Description

Méthode ; teste un élément de projet. Si le test échoue parce que l'élément n'est pas un fichier FLA ou HTML, cette méthode renvoie `false`.

## Exemple

L'exemple suivant teste tous les fichiers FLA et HTML du projet :

```
for (var i in fl.getProject().items) {  
    var item = fl.getProject().items[i];  
    if (item.canTest()) {  
        item.test();  
    }  
}
```

## Voir aussi

[fl.getProject\(\)](#), [project.canTestProject\(\)](#), [project.items](#),  
[projectItem.canTest\(\)](#)

# Objet Rectangle

**Héritage** [Objet Element](#) > [Objet Shape](#) > objet Rectangle

## Disponibilité

Flash CS3 Professional.

## Description

L'objet Rectangle est une forme tracée à l'aide d'une primitive de type rectangle.

Pour déterminer si un élément est un objet Rectangle, utilisez [shape.isRectangleObject](#).

## Résumé des propriétés de l'objet Rectangle

Outre les propriétés de l'[Objet Shape](#), les propriétés suivantes sont également disponibles pour l'objet Rectangle. Si vous voulez définir les propriétés d'un objet Rectangle, utilisez [document.setRectangleObjectProperty\(\)](#).

Propriété	Description
<a href="#">RectangleObject.bottomLeftRadius</a>	En lecture seule ; valeur codée en virgule flottante qui définit le rayon de l'angle inférieur gauche de l'objet Rectangle.
<a href="#">RectangleObject.bottomRightRadius</a>	En lecture seule ; valeur codée en virgule flottante qui définit le rayon de l'angle inférieur droit de l'objet Rectangle.
<a href="#">RectangleObject.lockFlag</a>	En lecture seule ; valeur booléenne déterminant si les valeurs des rayons des différents angles du rectangle peuvent être différentes.
<a href="#">RectangleObject.topLeftRadius</a>	En lecture seule ; valeur codée en virgule flottante qui définit le rayon de tous les angles du rectangle, ou seulement le rayon de l'angle supérieur gauche de l'objet Rectangle.
<a href="#">RectangleObject.topRightRadius</a>	En lecture seule ; valeur codée en virgule flottante qui définit le rayon de l'angle supérieur droit de l'objet Rectangle.

# RectangleObject.bottomLeftRadius

## Disponibilité

Flash CS3 Professional.

## Utilisation

`RectangleObject.bottomLeftRadius`

## Description

Propriété en lecture seule ; valeur codée en virgule flottante qui définit le rayon de l'angle inférieur gauche de l'objet Rectangle. Si la valeur de `RectangleObject.lockFlag` est true, la définition de cette valeur n'a aucun effet.

Pour définir cette valeur, utilisez `document.setRectangleObjectProperty()`.

## Voir aussi

`document.setRectangleObjectProperty()`, `RectangleObject.bottomRightRadius`, `RectangleObject.lockFlag`, `RectangleObject.topLeftRadius`, `RectangleObject.topRightRadius`

# RectangleObject.bottomRightRadius

## Disponibilité

Flash CS3 Professional.

## Utilisation

`RectangleObject.bottomRightRadius`

## Description

Propriété en lecture seule ; valeur codée en virgule flottante qui définit le rayon de l'angle inférieur droit de l'objet Rectangle. Si la valeur de `RectangleObject.lockFlag` est true, la définition de cette valeur n'a aucun effet.

Pour définir cette valeur, utilisez `document.setRectangleObjectProperty()`.

## Voir aussi

`document.setRectangleObjectProperty()`, `RectangleObject.bottomLeftRadius`, `RectangleObject.lockFlag`, `RectangleObject.topLeftRadius`, `RectangleObject.topRightRadius`

# RectangleObject.lockFlag

## Disponibilité

Flash CS3 Professional.

## Utilisation

`RectangleObject.lockFlag`

## Description

Propriété en lecture seule ; valeur booléenne déterminant si les valeurs des rayons des différents angles du rectangle peuvent être différentes. Si cette valeur est `true`, les angles prennent tous la valeur `RectangleObject.topLeftRadius`. Si elle est `false`, chaque rayon d'angle peut être défini séparément.

Pour définir cette valeur, utilisez `document.setRectangleObjectProperty()`.

## Voir aussi

`document.setRectangleObjectProperty()`, `RectangleObject.bottomLeftRadius`, `RectangleObject.bottomRightRadius`, `RectangleObject.topLeftRadius`, `RectangleObject.topRightRadius`

# RectangleObject.topLeftRadius

## Disponibilité

Flash CS3 Professional.

## Utilisation

`RectangleObject.topLeftRadius`

## Description

Propriété en lecture seule ; valeur codée en virgule flottante qui définit le rayon de tous les angles du rectangle (si la valeur de `RectangleObject.lockFlag` est `true`), ou seulement le rayon de l'angle supérieur gauche (si la valeur de `RectangleObject.lockFlag` est `false`).

Pour définir cette valeur, utilisez `document.setRectangleObjectProperty()`.

## Voir aussi

`document.setRectangleObjectProperty()`, `RectangleObject.bottomLeftRadius`, `RectangleObject.bottomRightRadius`, `RectangleObject.lockFlag`, `RectangleObject.topRightRadius`

# RectangleObject.topRightRadius

## Disponibilité

Flash CS3 Professional.

## Utilisation

`RectangleObject.topRightRadius`

## Description

Propriété en lecture seule ; valeur codée en virgule flottante qui définit le rayon de l'angle supérieur droit de l'objet Rectangle. Si la valeur de `RectangleObject.lockFlag` est `true`, la définition de cette valeur n'a aucun effet.

Pour définir cette valeur, utilisez `document.setRectangleObjectProperty()`.

## Voir aussi

`document.setRectangleObjectProperty()`, `RectangleObject.bottomLeftRadius`, `RectangleObject.bottomRightRadius`, `RectangleObject.lockFlag`, `RectangleObject.topLeftRadius`

# Objet Screen

## Disponibilité

Flash MX 2004.

## Description

L'objet Screen représente un écran séparé dans une diapositive ou un formulaire. Cet objet contient les propriétés propres à la diapositive ou au formulaire. Pour accéder au tableau de tous les objets Screen du document, utilisez le code suivant :

```
fl.getDocumentDOM().screenOutline.screens
```

## Résumé des propriétés de l'objet Screen

L'objet Screen possède les propriétés suivantes :

Propriété	Description
<code>screen.accName</code>	Chaîne équivalente au champ Nom du panneau Accessibilité.
<code>screen.childScreens</code>	Propriété en lecture seule ; tableau des écrans enfants de cet écran. Ce tableau est vide s'il n'existe pas d'écrans enfants.
<code>screen.description</code>	Chaîne équivalente au champ Description du panneau Accessibilité.
<code>screen.forceSimple</code>	Valeur booléenne qui active et désactive l'accessibilité aux enfants de l'objet.
<code>screen.hidden</code>	Valeur booléenne indiquant si l'écran est visible.
<code>screen.instanceName</code>	Propriété en lecture seule ; chaîne représentant le nom de l'instance utilisée pour accéder à l'objet en ActionScript.
<code>screen.name</code>	Propriété en lecture seule ; chaîne représentant le nom de l'écran.
<code>screen.nextScreen</code>	Propriété en lecture seule ; objet représentant l'écran de même niveau suivant dans le tableau <code>childScreens</code> de l'écran parent.
<code>screen.parameters</code>	En lecture seule ; tableau contenant les propriétés ActionScript 2.0 accessibles à l'aide de l'Inspecteur de propriétés.
<code>screen.parentScreen</code>	Propriété en lecture seule ; chaîne représentant l'écran parent.
<code>screen.prevScreen</code>	Propriété en lecture seule ; objet représentant l'écran de même niveau précédent dans le tableau <code>childScreens</code> de l'écran parent.
<code>screen.silent</code>	Valeur booléenne indiquant si l'objet est accessible.
<code>screen.tabIndex</code>	Équivalent au champ Index de tabulation du panneau Accessibilité.
<code>screen.timeline</code>	Propriété en lecture seule ; l'objet Scénario de l'écran. Pour plus d'informations, consultez <a href="#">Objet Timeline</a> .

## screen.accName

### Disponibilité

Flash MX 2004.

### Utilisation

`screen.accName`

### Description

Propriété ; chaîne équivalente au champ Nom du panneau Accessibilité. Les logiciels de lecture vocale identifient l'objet en prononçant ce nom.

### Exemple

L'exemple suivant enregistre dans la variable `theName` la valeur du nom de l'objet :

```
var theName = fl.getDocumentDOM().screenOutline.screens[1].accName;
```

L'exemple suivant définit comme "Home Button" le nom de l'objet :

```
fl.getDocumentDOM().screenOutline.screens[1].accName = 'Home Button';
```

## screen.childScreens

### Disponibilité

Flash MX 2004.

### Utilisation

`screen.childScreens`

### Description

Propriété en lecture seule ; tableau des écrans enfants de cet écran. Ce tableau est vide s'il n'existe pas d'écrans enfants.

### Exemple

L'exemple suivant vérifie que le document actuel est une diapositive ou un formulaire et, si c'est le cas, enregistre dans la variable `myChildren` le tableau des écrans enfants avant d'afficher leurs noms dans le panneau de sortie :

```
var myChildren = new Array();
if(fl.getDocumentDOM().allowScreens) {
    var myParent = fl.getDocumentDOM().screenOutline.rootScreen.name
    for (i in fl.getDocumentDOM().screenOutline.rootScreen.childScreens) {
        myChildren.push("
        "+fl.getDocumentDOM().screenOutline.rootScreen.childScreens[i].name);
    }
    fl.trace(" The child screens of "+myParent+" are "+myChildren+" . ");
}
```



## screen.description

### Disponibilité

Flash MX 2004.

### Utilisation

screen.description

### Description

Propriété ; chaîne équivalente au champ Description du panneau Accessibilité. Cette description est lue par les logiciels de lecture vocale.

### Exemple

L'exemple suivant enregistre dans la variable theDescription la description de l'objet :

```
var theDescription =  
    fl.getDocumentDOM().screenOutline.screens[1].description;
```

L'exemple suivant définit comme "This is Screen 1" la description de l'objet :

```
fl.getDocumentDOM().screenOutline.screens[1].description = "This is Screen  
1"
```

## screen.forceSimple

### Disponibilité

Flash MX 2004.

### Utilisation

screen.forceSimple

### Description

Propriété ; valeur booléenne qui active ou désactive l'accessibilité aux enfants de l'objet. Cette propriété est équivalente à l'inverse du paramètre Rendre les objets enfant accessibles du panneau Accessibilité. Ainsi, si forceSimple est true, son effet est similaire à la désactivation de l'option Rendre les objets enfant accessibles. Si forceSimple est false, son effet est similaire à l'activation de l'option Rendre les objets enfant accessibles.

### Exemple

L'exemple suivant donne à la variable areChildrenAccessible la valeur de la propriété forceSimple ; une valeur false signifie que les enfants sont accessibles :

```
var areChildrenAccessible =  
    fl.getDocumentDOM().screenOutline.screens[1].forceSimple
```

L'exemple suivant rend les enfants de l'objet accessibles :

```
fl.getDocumentDOM().screenOutline.screens[1].forceSimple = false;
```

## screen.hidden

### Disponibilité

Flash MX 2004.

### Utilisation

screen.hidden

### Description

Propriété ; valeur booléenne indiquant si l'écran est visible. Un écran dont la propriété `hidden` a la valeur `true` n'est pas visible dans un autre écran.

### Exemple

L'exemple suivant teste si le premier écran est caché, et change sa visibilité en conséquence. Un message s'affiche ensuite dans le panneau de sortie pour indiquer la visibilité de l'écran avant le changement :

```
if (fl.getDocumentDOM().screenOutline.screens[0].hidden) {  
    fl.getDocumentDOM().screenOutline.setScreenProperty("hidden", false);  
    fl.trace(fl.getDocumentDOM().screenOutline.screens[0].name+" had its  
    'hidden' property set to 'false'");  
}  
else {  
    fl.getDocumentDOM().screenOutline.setScreenProperty("hidden", true);  
    fl.trace(fl.getDocumentDOM().screenOutline.screens[0].name+" had its  
    'hidden' property set to 'true'");  
}
```

## screen.instanceName

### Disponibilité

Flash MX 2004.

### Utilisation

screen.instanceName

### Description

Propriété en lecture seule ; chaîne représentant le nom de l'instance utilisée pour accéder à l'objet en ActionScript.

## Exemple

L'exemple suivant vérifie si le document actif autorise les écrans (c'est-à-dire s'il est de type diapositive ou formulaire). Il affecte ensuite la valeur `instanceName` du premier écran enfant du tableau à la variable `myInstanceName` et ouvre le panneau de sortie pour afficher le nom d'occurrence de l'écran :

```
var myChildren = new Array();
if(fl.getDocumentDOM().allowScreens) {
    var myInstanceName =
        fl.getDocumentDOM().screenOutline.rootScreen.childScreens[0].instanceName;
    fl.trace(" The instanceName is "+myInstanceName+". ");
}
```

## screen.name

### Disponibilité

Flash MX 2004.

### Utilisation

`screen.name`

### Description

Propriété en lecture seule ; chaîne représentant le nom de l'écran.

## Exemple

L'exemple suivant vérifie si le document actif autorise les écrans (c'est-à-dire s'il est de type diapositive ou formulaire). Il affecte ensuite la valeur `name` du premier écran enfant du tableau à la variable `myName`, et ouvre le panneau de sortie pour afficher le nom de l'écran :

```
var myChildren = new Array();
if(fl.getDocumentDOM().allowScreens) {
    var myName =
        fl.getDocumentDOM().screenOutline.rootScreen.childScreens[0].name;
    fl.trace("The name of the screen is "+myName+". ");
}
```

# screen.nextScreen

## Disponibilité

Flash MX 2004.

## Utilisation

`screen.nextScreen`

## Description

Propriété en lecture seule ; objet représentant l'écran de même niveau suivant dans le tableau `childScreens` de l'écran parent. Autrement dit, il faut parcourir un tableau d'écrans enfants jusqu'au prochain écran du tableau pour connaître la valeur de `screen.nextScreen`.

Voir [screen.prevScreen](#).

S'il n'existe pas d'écrans de même niveau, la valeur est `null`.

## Exemple

L'exemple suivant vérifie que le document actuel est une diapositive ou un formulaire et, si c'est le cas, il affiche la séquence des écrans dans le panneau de sortie :

```
if(fl.getDocumentDOM().allowScreens) {  
    var myCurrent =  
        fl.getDocumentDOM().screenOutline.rootScreen.childScreens[0].name;  
    var myNext =  
        fl.getDocumentDOM().screenOutline.rootScreen.childScreens[0].nextScreen.  
        name;  
    fl.trace(" The next screen to "+myCurrent+" is "+myNext+". ");  
}
```

# screen.parameters

## Disponibilité

Flash MX 2004.

## Utilisation

`screen.parameters`

## Description

Propriété en lecture seule ; tableau contenant les propriétés ActionScript 2.0 accessibles à l'aide de l'Inspecteur de propriétés.

## Exemple

L'exemple suivant enregistre les paramètres du second écran dans la variable `parms`, puis affecte la valeur "some value" à la première propriété :

```
var parms = fl.getDocumentDOM().screenOutline.screens[1].parameters;  
parms[0].value = "some value";
```

## Voir aussi

[Objet Parameter](#)

# screen.parentScreen

## Disponibilité

Flash MX 2004.

## Utilisation

`screen.parentScreen`

## Description

Propriété en lecture seule ; objet représentant l'écran parent. Si `parentScreen` est `null`, l'écran est un écran de niveau supérieur.

## Exemple

L'exemple suivant enregistre les valeurs des propriétés `childScreen` et `parentScreen` dans des variables, puis affiche ces valeurs et leurs relations parent/enfant dans le panneau de sortie :

```
if(fl.getDocumentDOM().allowScreens) {  
    var myCurrent =  
        fl.getDocumentDOM().screenOutline.rootScreen.childScreens[1].name;  
    var myParent =  
        fl.getDocumentDOM().screenOutline.rootScreen.childScreens[1].parentScreen.name;  
    fl.trace(" The parent screen to "+myCurrent+" is "+myParent+". ");  
}
```

# screen.prevScreen

## Disponibilité

Flash MX 2004.

## Utilisation

`screen.prevScreen`

## Description

Propriété en lecture seule ; objet représentant l'écran de même niveau précédent dans le tableau `childScreens` de l'écran parent. S'il n'existe pas d'écrans de même niveau, la valeur est `null`. Voir aussi `screen.nextScreen`.

## Exemple

L'exemple suivant vérifie que le document actuel est une diapositive ou un formulaire et, si c'est le cas, il affiche la séquence des écrans dans le panneau de sortie :

```
if(fl.getDocumentDOM().allowScreens) {  
    var myCurrent =  
        fl.getDocumentDOM().screenOutline.rootScreen.childScreens[1].name;  
    var myNext =  
        fl.getDocumentDOM().screenOutline.rootScreen.childScreens[1].prevScreen.  
        name;  
    fl.trace(" The previous screen to "+myCurrent+" is "+myNext+". ");  
}
```

# screen.silent

## Disponibilité

Flash MX 2004.

## Utilisation

`screen.silent`

## Description

Propriété ; valeur booléenne indiquant si l'objet est accessible. Cette propriété est équivalente à l'inverse du paramètre `Rendre l'objet accessible` du panneau `Accessibilité`. Ainsi, si `silent` est `true`, son effet est similaire à la désactivation de l'option `Rendre l'objet accessible` dans le panneau `Accessibilité`. Si `silent` est `false`, son effet est similaire à l'activation de l'option `Rendre l'objet accessible` dans le panneau `Accessibilité`.

## Exemple

L'exemple suivant lit la valeur de la propriété `silent` de l'objet (la valeur `false` signifie que l'objet est accessible):

```
var isSilent = fl.getDocumentDOM().screenOutline.screens[1].silent;
```

L'exemple suivant rend l'objet accessible :

```
fl.getDocumentDOM().screenOutline.screens[1].silent = false;
```

# screen.tabIndex

## Disponibilité

Flash MX 2004.

## Utilisation

`screen.tabIndex`

## Description

Propriété ; entier équivalent au champ Index de tabulation du panneau Accessibilité. Cette valeur permet de déterminer l'ordre dans lequel les objets seront sélectionnés lorsque l'utilisateur appuiera sur la touche de tabulation.

## Exemple

L'exemple suivant lit l'index de tabulation de l'objet :

```
var theTabIndex = fl.getDocumentDOM().screenOutline.screens[1].tabIndex;
```

L'exemple suivant met l'index de tabulation de l'objet à 1 :

```
fl.getDocumentDOM().screenOutline.screens[1].tabIndex = 1;
```

# screen.timeline

## Disponibilité

Flash MX 2004.

## Utilisation

`screen.timeline`

## Description

Propriété en lecture seule ; [Objet Timeline](#) de l'écran.

## Exemple

L'exemple suivant lit la propriété `screenOutline` du document actif, affecte le tableau des propriétés de `timeline` du premier écran à la variable `myArray`, et affiche ces propriétés dans le panneau de sortie :

```
myArray = new Array();
if(fl.getDocumentDOM().screenOutline) {
    for(i in fl.getDocumentDOM().screenOutline.screens[0].timeline) {
        myArray.push(" "+i+" : "+fl.getDocumentDOM().screenOutline.screens[0].timeline[i]+" " );
    }
    fl.trace("Here are the properties of the screen named "+
        fl.getDocumentDOM().screenOutline.screens[0].name+": "+myArray);
}
```

# Objet ScreenOutline

## Disponibilité

Flash MX 2004.

## Description

L'objet ScreenOutline représente le groupe d'écrans d'une diapositive ou d'un formulaire. Cet objet est accessible à l'aide de `fl.getDocumentDOM().screenOutline`.

L'objet ScreenOutline n'existe que si le document est une diapositive ou un formulaire. C'est pourquoi, avant d'y accéder, il est nécessaire d'utiliser `document.allowScreens()` pour vérifier qu'il existe bien un document Screens, comme dans l'exemple suivant :

```
if(fl.getDocumentDOM().allowScreens) {  
    var myName =  
        fl.getDocumentDOM().screenOutline.rootScreen.childScreens[0].name;  
    fl.trace("The name of the screen is " + myName + ".");  
}
```

## Résumé des méthodes de l'objet ScreenOutline

Vous pouvez utiliser les méthodes suivantes avec l'objet ScreenOutline :

Méthode	Description
<code>screenOutline.copyScreenFromFile()</code>	Insère sous l'écran actuellement sélectionné tous les écrans (ou un écran spécifié et ses enfants) d'un document spécifié.
<code>screenOutline.deleteScreen()</code>	Supprime les écrans actuellement sélectionnés ou un écran spécifié, ainsi que leurs enfants.
<code>screenOutline.duplicateScreen()</code>	Duplique les écrans actuellement sélectionnés ou un écran spécifié.
<code>screenOutline.getSelectedScreens()</code>	Renvoie un tableau des objets Screen actuellement sélectionnés dans le groupe d'écrans.
<code>screenOutline.insertNestedScreen()</code>	Insère un écran imbriqué du type spécifié à un emplacement particulier du groupe d'écrans.
<code>screenOutline.insertScreen()</code>	Insère un nouvel écran vierge du type spécifié à un emplacement spécifique du document.
<code>screenOutline.moveScreen()</code>	Déplace l'écran spécifié en fonction de la valeur du paramètre <code>referenceScreen</code> , soit avant, après, comme premier enfant ou comme dernier enfant.



Méthode	Description
<code>screenOutline.renameScreen()</code>	Donne un nouveau nom à l'écran portant le nom spécifié.
<code>screenOutline.setCurrentScreen()</code>	Affecte l'écran spécifié à la sélection actuelle dans le groupe d'écrans.
<code>screenOutline.setScreenProperty()</code>	Définit la propriété spécifiée avec la valeur indiquée pour les écrans sélectionnés.
<code>screenOutline.setSelectedScreens()</code>	Sélectionne les écrans spécifiés dans le panneau Contour de l'écran.

## Résumé des propriétés de l'objet ScreenOutline

Vous pouvez utiliser les propriétés suivantes avec l'objet ScreenOutline :

Propriété	Description
<code>screenOutline.currentScreen</code>	Un <a href="#">Objet Screen</a> ; l'écran actuellement sélectionné.
<code>screenOutline.rootScreen</code>	Propriété en lecture seule ; premier écran du groupe d'écrans.
<code>screenOutline.screens</code>	Propriété en lecture seule ; tableau des objets Screen de niveau supérieur (voir <a href="#">Objet Screen</a> ) contenus dans le document.

## screenOutline.copyScreenFromFile()

### Disponibilité

Flash MX 2004.

### Utilisation

```
screenOutline.copyScreenFromFile(fileURI [, screenName])
```

### Paramètres

*fileURI* Chaîne de type « fichier URI » qui spécifie le nom du fichier FLA qui contient les écrans à copier dans le document.

*screenName* Nom de l'écran à copier. Si le paramètre *screenName* est présent, Flash copie cet écran et ses enfants. Si le paramètre *screenName* est omis, Flash copie tout le document.

Ce paramètre est facultatif.

## Valeur renvoyée

Aucune. Si le fichier est introuvable, ou si ce n'est pas un fichier FLA valide, ou encore si l'écran spécifié est introuvable, une erreur est affichée et l'exécution du script est interrompue.

## Description

Méthode ; insère sous l'écran actuellement sélectionné tous les écrans (ou un écran spécifié et ses enfants) d'un document spécifié. Si plusieurs écrans sont sélectionnés, les nouveaux écrans sont insérés sous le dernier écran sélectionné, au même niveau.

## Exemple

L'exemple suivant copie l'écran « slide1 » depuis le fichier myTarget.flas (situé sur le Bureau) dans le document actif (remplacez la chaîne *userName* par votre nom d'utilisateur) :

```
fl.getDocumentDOM().screenOutline.copyScreenFromFile("file:///C:/Documents  
and Settings/userName/Desktop/myTarget.flas", "slide1");
```

# screenOutline.currentScreen

## Disponibilité

Flash MX 2004.

## Utilisation

screenOutline.currentScreen

## Description

Propriété ; un objet Screen, l'écran actuellement sélectionné (consultez la section [Objet Screen](#)).

## Exemple

L'exemple suivant enregistre dans la variable myScreen l'objet currentScreen, puis affiche le nom de cet écran dans le panneau de sortie :

```
var myScreen = fl.getDocumentDOM().screenOutline.currentScreen;  
fl.trace(myScreen.name);
```

## screenOutline.deleteScreen()

### Disponibilité

Flash MX 2004.

### Utilisation

```
screenOutline.deleteScreen([screenName])
```

### Paramètres

*screenName* Chaîne spécifiant le nom de l'écran à supprimer. Si le paramètre *screenName* est omis, le ou les écrans actuellement sélectionnés, ainsi que leurs enfants, sont supprimés. Ce paramètre est facultatif.

### Valeur renvoyée

Aucune.

### Description

Méthode ; supprime les écrans actuellement sélectionnés ou un écran spécifié, ainsi que leurs enfants.

### Exemple

L'exemple suivant supprime l'écran nommé « apple » et tous ses enfants :

```
fl.getDocumentDOM().screenOutline.deleteScreen("apple");
```

## screenOutline duplicateScreen()

### Disponibilité

Flash MX 2004.

### Utilisation

```
screenOutline.duplicateScreen([screenName])
```

### Paramètres

*screenName* Chaîne spécifiant le nom de l'écran à dupliquer. Si le paramètre *screenName* est omis, le ou les écrans actuellement sélectionnés sont dupliqués. Ce paramètre est facultatif.

### Valeur renvoyée

Une valeur booléenne : `true` en cas de réussite, `false` sinon.

## Description

Méthode ; duplique les écrans actuellement sélectionnés ou un écran spécifié. Les écrans dupliqués reçoivent un nom par défaut, qui est créé en ajoutant `_copy` à la fin du nom de l'écran original, par exemple `ecran_copy`, `ecran_copy2`, etc. Si vous dupliquez plusieurs écrans, ces duplicatas sont placés directement sous l'écran sélectionné ayant le niveau le plus bas dans la hiérarchie du groupe d'écrans.

## Exemple

L'exemple suivant duplique un écran nommé « apple » :

```
fl.getDocumentDOM().screenOutline duplicateScreen("apple");
```

# screenOutline.getSelectedScreens()

## Disponibilité

Flash MX 2004.

## Utilisation

```
screenOutline.getSelectedScreens()
```

## Paramètres

Aucun.

## Valeur renvoyée

Un tableau des objets Screen sélectionnés (voir [Objet Screen](#)).

## Description

Méthode ; renvoie un tableau des objets Screen actuellement sélectionnés dans le groupe d'écrans.

## Exemple

L'exemple suivant enregistre dans la variable `myArray` les objets Screen actuellement sélectionnés, puis affiche le nom de ces écrans dans le panneau de sortie :

```
var myArray = fl.getDocumentDOM().screenOutline.getSelectedScreens();
for (var i in myArray) {
    fl.trace(myArray[i].name)
}
```

# screenOutline.insertNestedScreen()

## Disponibilité

Flash MX 2004.

## Utilisation

```
screenOutline.insertNestedScreen([name [, referenceScreen  
[, screenTypeName]])
```

## Paramètres

*name* Chaîne spécifiant le nom de l'écran à insérer. Si cette chaîne est vide, le nouvel écran inséré recevra un nom par défaut, par exemple Slide *n* ou Form *n* (où *n* est le premier chiffre unique disponible). Ce paramètre est facultatif.

*referenceScreen* Une chaîne qui indique le nom de l'écran sous lequel le nouvel écran doit être inséré en tant qu'enfant. Si ce paramètre n'est pas spécifié, le nouvel écran est inséré en tant qu'enfant de l'écran actuellement sélectionné. Ce paramètre est facultatif.

*screenTypeName* Chaîne spécifiant le type d'écran à affecter au nouvel écran imbriqué. Le type d'écran et son nom de classe sont alors définis pour cet écran. Les valeurs acceptables sont "Form" et "Slide". Ce paramètre est facultatif. Si ce paramètre est omis, le type est hérité de l'écran parent.

## Valeur renvoyée

Un [Objet Screen](#)

## Description

Méthode ; insère un écran imbriqué du type spécifié à un emplacement particulier du groupe d'écrans.

## Exemple

L'exemple suivant insère l'écran slide2 comme enfant de slide1 :

```
fl.getDocumentDOM().screenOutline.insertNestedScreen("slide2", "slide1",  
"Slide");
```

# screenOutline.insertScreen()

## Disponibilité

Flash MX 2004.

## Utilisation

```
screenOutline.insertScreen([name [, referenceScreen [, screenTypeName]])
```

## Paramètres

*name* Chaîne spécifiant le nom de l'écran à insérer. Si ce paramètre est omis, la méthode insère un nouvel écran avec un nom par défaut, par exemple Slide *n* ou Form *n* (où *n* est le premier chiffre unique disponible). Ce paramètre est facultatif.

*referenceScreen* Chaîne spécifiant le nom de l'écran situé avant le nouvel écran. Si ce paramètre n'est pas spécifié, le nouvel écran est inséré après l'écran actuellement sélectionné. Si le paramètre *referenceScreen* identifie un écran enfant, le nouvel écran sera de même niveau que cet écran enfant, donc un enfant du même écran parent. Ce paramètre est facultatif.

*screenTypeName* Chaîne spécifiant le type d'écran à affecter au nouvel écran. Le type d'écran et son nom de classe sont alors définis pour cet écran. Les valeurs acceptables sont "Form" et "Slide". Ce paramètre est facultatif.

## Valeur renvoyée

Un [Objet Screen](#)

## Description

Méthode ; insère un nouvel écran vierge du type spécifié à un emplacement spécifique du document.

## Exemple

L'exemple suivant insère un formulaire nommé « slide2 » après l'écran nommé « slide1 » :

```
fl.getDocumentDOM().screenOutline.insertScreen("slide2","slide1","Form");
```

L'exemple suivant insère une diapositive nommée « slide4 » après l'écran nommé « slide3 » :

```
fl.getDocumentDOM().screenOutline.insertScreen("slide4","slide3","Slide");
```

# screenOutline.moveScreen()

## Disponibilité

Flash MX 2004.

## Utilisation

`screenOutline.moveScreen(screenToMove, referenceScreen, position)`

## Paramètres

*screenToMove* Chaîne indiquant le nom de l'écran à déplacer.

*referenceScreen* Chaîne indiquant l'écran à côté duquel l'écran indiqué par le paramètre *screenToMove* sera placé.

*position* Chaîne indiquant l'emplacement où l'écran doit être déplacé, par rapport à l'écran indiqué par le paramètre *referenceScreen*. Les valeurs possibles sont les suivantes : "before" (avant), "after" (après), "firstChild" (premier enfant) et "lastChild" (dernier enfant).

## Valeur renvoyée

Une valeur booléenne : `true` en cas de réussite, `false` sinon.

## Description

Méthode ; déplace l'écran spécifié en fonction de la valeur du paramètre *referenceScreen*, soit avant, après, comme premier enfant ou comme dernier enfant.

## Exemple

L'exemple suivant déplace l'écran `slide1` en en faisant le premier enfant de l'écran `slide2` :

```
fl.getDocumentDOM().screenOutline.moveScreen("slide1", "slide2",  
    "firstChild");
```

# screenOutline.renameScreen()

## Disponibilité

Flash MX 2004.

## Utilisation

`screenOutline.renameScreen(newScreenName [, oldScreenName  
 [, bDisplayError]])`

## Paramètres

*newScreenName* Chaîne spécifiant le nouveau nom de l'écran.

*oldScreenName* Chaîne spécifiant le nom de l'écran à modifier. Si ce paramètre est omis, le nom de l'écran actuellement sélectionné est modifié. Ce paramètre est facultatif.

*bDisplayError* Valeur booléenne. Si sa valeur est `true`, un message d'erreur est affiché en cas d'erreur, par exemple si un écran porte déjà le nom passé dans le paramètre *newScreenName*. La valeur par défaut est `false`.

## Valeur renvoyée

Une valeur booléenne : `true` en cas de réussite, `false` sinon.

## Description

Méthode ; donne un nouveau nom à l'écran ayant le nom spécifié.

## Exemple

L'exemple suivant remplace par « Intro » le nom de l'écran « slide1 » :

```
fl.getDocumentDOM().screenOutline.renameScreen("Intro", "slide1");
```

# screenOutline.rootScreen

## Disponibilité

Flash MX 2004.

## Utilisation

```
screenOutline.rootScreen
```

## Description

Propriété en lecture seule ; premier écran du groupe d'écrans. Vous pouvez utiliser `screenOutline.rootScreen` comme raccourci de `screenOutline.screens[0]`.

## Exemple

L'exemple suivant affiche le nom du premier enfant du premier écran du groupe d'écrans :

```
var n = fl.getDocumentDOM().screenOutline.rootScreen.childScreens[0].name;  
fl.trace(n);
```



## screenOutline.screens

### Disponibilité

Flash MX 2004.

### Utilisation

`screenOutline.screens`

### Description

Propriété en lecture seule ; tableau des objets Screen de niveau supérieur (voir [Objet Screen](#)) contenus dans le document.

### Exemple

L'exemple suivant enregistre dans la variable `myArray` le tableau des objets Screen, puis affiche le nom de ces écrans dans le panneau de sortie :

```
var myArray = new Array();
if(fl.getDocumentDOM().allowScreens) {
    for(var i in fl.getDocumentDOM().screenOutline.screens) {
        myArray.push(" "+fl.getDocumentDOM().screenOutline.screens[i].name);
    }
    fl.trace(2"The screens array contains objects whose names are:
        "+myArray+". ");
}
```

## screenOutline.setCurrentScreen()

### Disponibilité

Flash MX 2004.

### Utilisation

`screenOutline.setCurrentScreen(name)`

### Paramètres

*name* Chaîne indiquant l'écran qui doit devenir sélectionné. Si cet écran est un enfant d'un autre écran, il n'est pas nécessaire d'indiquer un chemin ou une hiérarchie.

### Valeur renvoyée

Aucune.

### Description

Méthode ; affecte l'écran spécifié à la sélection actuelle dans le groupe d'écrans.

## Exemple

L'exemple suivant définit l'écran nommé ChildOfSlide\_1 comme écran actif :

```
fl.getDocumentDOM().screenOutline.setCurrentScreen("ChildOfSlide_1");
```

# screenOutline.setScreenProperty()

## Disponibilité

Flash MX 2004.

## Utilisation

```
screenOutline.setScreenProperty(property, value)
```

## Paramètres

*property* Chaîne définissant la propriété à modifier.

*value* Nouvelle valeur de la propriété. Le type de valeur dépend de la propriété à définir.

La liste complète des propriétés et de leurs valeurs figure dans la section [Résumé des propriétés de l'objet Screen](#).

## Valeur renvoyée

Aucune.

## Description

Méthode ; affecte la propriété spécifiée, avec la valeur spécifiée, aux écrans sélectionnés.

## Exemple

L'exemple suivant change la visibilité des écrans actuellement sélectionnés, en la faisant passer de « hidden » à « visible » :

```
fl.getDocumentDOM().screenOutline.setScreenProperty("hidden", false);
```

# screenOutline.setSelectedScreens()

## Disponibilité

Flash MX 2004.

## Utilisation

```
screenOutline.setSelectedScreens(selection [, bReplaceCurrentSelection])
```

## Paramètres

*selection*    Tableau des noms des écrans à sélectionner dans le groupe d'écrans.

*bReplaceCurrentSelection*    Valeur booléenne. Si elle est *true*, la sélection actuelle peut être désélectionnée. La valeur par défaut est *true*. Si elle est *false*, Flash étend la sélection actuelle aux écrans spécifiés. Ce paramètre est facultatif.

## Valeur renvoyée

Aucune.

## Description

Méthode ; sélectionne les écrans spécifiés dans le groupe d'écrans. Si plusieurs écrans sont spécifiés, celui qui a la dernière valeur d'index dans le tableau de la sélection reçoit le focus sur la scène.

## Exemple

L'exemple suivant désélectionne tous les écrans actuellement sélectionnés, puis sélectionne les écrans *slide1*, *slide2*, *slide3* et *slide4* dans le groupe d'écrans :

```
myArray = new Array("slide1", "slide2", "slide3", "slide4");  
fl.getDocumentDOM().screenOutline.setSelectedScreens(myArray, true);
```

# Objet Shape

Héritage [Objet Element](#) > objet Shape

## Disponibilité

Flash MX 2004.

## Description

L'objet Shape est une sous-classe de l'objet Element. L'objet Shape autorise un contrôle plus précis que les API de dessin pour la manipulation ou la création de formes géométriques sur la scène. Cette précision de contrôle est nécessaire pour permettre aux scripts de créer des effets utiles et autres commandes de dessin (voir [Objet Element](#)).

Pour fonctionner correctement, toutes les méthodes et propriétés de Shape qui modifient une forme ou l'un de ses composants doivent être placées entre des appels de `shape.beginEdit()` et `shape.endEdit()`.

## Résumé des méthodes pour l'objet Shape

Outre les méthodes de l'[Objet Element](#), les méthodes suivantes sont disponibles avec l'objet Shape :

Méthode	Description
<code>shape.beginEdit()</code>	Définit le début d'une session de modification.
<code>shape.deleteEdge()</code>	Supprime le contour spécifié.
<code>shape.endEdit()</code>	Définit la fin d'une session de modification.

## Résumé des propriétés de l'objet Shape

Outre les propriétés de l'[Objet Element](#), les propriétés suivantes sont disponibles avec l'objet Shape :

Propriété	Description
<code>shape.contours</code>	Propriété en lecture seule ; tableau des objets Contour de la forme (voir <a href="#">Objet Contour</a> ).
<code>shape.edges</code>	Propriété en lecture seule ; tableau des objets Edge (voir <a href="#">Objet Edge</a> ).
<code>shape.isDrawingObject</code>	Propriété en lecture seule ; si elle est <code>true</code> , la forme est un objet de dessin.

Propriété	Description
<code>shape.isGroup</code>	Propriété en lecture seule ; si elle est <code>true</code> , la forme est un groupe.
<code>shape.isOvalObject</code>	En lecture seule ; si la valeur est <code>true</code> , la forme est un objet Oval de type primitif (créé à l'aide de l'outil Oval).
<code>shape.isRectangleObject</code>	En lecture seule ; si la valeur est <code>true</code> , la forme est un objet Rectangle de type primitif (créé à l'aide de l'outil Rectangle).
<code>shape.vertices</code>	Propriété en lecture seule ; tableau des objets Vertex (consultez la section <a href="#">Objet Vertex</a> ).

## shape.beginEdit()

### Disponibilité

Flash MX 2004.

### Utilisation

`shape.beginEdit()`

### Paramètres

Aucun.

### Valeur renvoyée

Aucune.

### Description

Méthode ; définit le début d'une session de modification. Cette méthode doit être appelée avant toute commande modifiant l'objet Shape ou l'un de ses composants.

### Exemple

L'exemple suivant supprime le premier contour du tableau des contours de la forme actuellement sélectionnée :

```
var shape = fl.getDocumentDOM().selection[0];
shape.beginEdit();
shape.deleteEdge(0);
shape.endEdit();
```

# shape.contours

## Disponibilité

Flash MX 2004.

## Utilisation

shape.contours

## Description

Propriété en lecture seule ; tableau des objets Contour de la forme (consultez la section [Objet Contour](#)).

## Exemple

L'exemple suivant enregistre dans la variable *c* le premier contour du tableau des contours, puis enregistre l'[Objet halfEdge](#) de ce contour dans la variable *he* :

```
var c = fl.getDocumentDOM().selection[0].contours[0];  
var he = c.getHalfEdge();
```

# shape.deleteEdge()

## Disponibilité

Flash MX 2004.

## Utilisation

shape.deleteEdge(*index*)

## Paramètres

*index*    Index (en base zéro) qui spécifie le contour à supprimer du tableau [shape.edges](#). Cette méthode modifie la longueur du tableau [shape.edges](#).

## Valeur renvoyée

Aucune.

## Description

Méthode ; supprime le contour spécifié. Il est nécessaire d'appeler [shape.beginEdit\(\)](#) avant d'utiliser cette méthode.

## Exemple

L'exemple suivant supprime le premier contour du tableau des contours de la forme actuellement sélectionnée :

```
var shape = fl.getDocumentDOM().selection[0];
shape.beginEdit();
shape.deleteEdge(0);
shape.endEdit();
```

## shape.edges

### Disponibilité

Flash MX 2004.

### Utilisation

shape.edges

### Description

Propriété en lecture seule ; tableau des objets Edge (consultez la section [Objet Edge](#)).

## shape.endEdit()

### Disponibilité

Flash MX 2004.

### Utilisation

shape.endEdit()

### Paramètres

Aucun.

### Valeur renvoyée

Aucune.

### Description

Méthode ; définit la fin d'une session de modification. Toutes les modifications apportées à l'objet Shape ou à l'un de ses composants sont appliquées à la forme. Cette méthode doit être appelée après toute commande modifiant l'objet Shape ou l'un de ses composants.

## Exemple

L'exemple suivant supprime le premier contour du tableau des contours dans la forme actuellement sélectionnée :

```
var shape = fl.getDocumentDOM().selection[0];
shape.beginEdit();
shape.deleteEdge(0);
shape.endEdit();
```

## shape.isDrawingObject

### Disponibilité

Flash 8.

### Utilisation

shape.isDrawingObject

### Description

Propriété en lecture seule ; si elle est `true`, la forme est un objet de dessin.

## Exemple

L'exemple suivant enregistre dans la variable `sel` le premier objet `item` sélectionné, puis utilise les propriétés `element.elementType` et `shape.isDrawingObject` pour déterminer si l'élément sélectionné est un objet de dessin :

```
var sel = fl.getDocumentDOM().selection[0];
var shapeDrawingObject = (sel.elementType == "shape") &&
    sel.isDrawingObject;
fl.trace(shapeDrawingObject);
```

### Voir aussi

[document.crop\(\)](#), [document.deleteEnvelope\(\)](#), [document.intersect\(\)](#),  
[document.punch\(\)](#), [document.union\(\)](#), [shape.isGroup](#)



# shape.isGroup

## Disponibilité

Flash MX 2004.

## Utilisation

`shape.isGroup`

## Description

Propriété en lecture seule ; si elle est `true`, la forme est un groupe.

## Exemple

L'exemple suivant enregistre dans la variable `sel` le premier objet `item` sélectionné, puis utilise les propriétés `element.elementType` et `shape.isGroup` pour déterminer si l'élément sélectionné est un groupe :

```
var sel = fl.getDocumentDOM().selection[0];
var shapeGroup = (sel.elementType == "shape") && sel.isGroup;
fl.trace(shapeGroup);
```

## Voir aussi

[shape.isDrawingObject](#)

# shape.isOvalObject

## Disponibilité

Flash CS3 Professional.

## Utilisation

`shape.isOvalObject`

## Description

Propriété en lecture seule ; si la valeur est `true`, la forme est un objet Oval de type primitif (créé à l'aide d'une primitive de type ovale).

## Exemple

L'exemple suivant affiche `"true"` si le premier élément sélectionné est un objet Oval de type primitif, `"false"` dans le cas contraire :

```
var sel = fl.getDocumentDOM().selection[0];
fl.trace(sel.isOvalObject);
```

## Voir aussi

[shape.isRectangleObject](#)

# shape.isRectangleObject

## Disponibilité

Flash CS3 Professional.

## Utilisation

`shape.isRectangleObject`

## Description

Propriété en lecture seule ; si la valeur est `true`, la forme est un objet Rectangle de type primitif (créé à l'aide d'une primitive de type rectangle).

## Exemple

L'exemple suivant affiche "`true`" si le premier élément sélectionné est un objet Rectangle de type primitif, "`false`" dans le cas contraire :

```
var sel = fl.getDocumentDOM().selection[0];  
fl.trace(sel.isRectangleObject);
```

## Voir aussi

[shape.isOvalObject](#)

# shape.vertices

## Disponibilité

Flash MX 2004.

## Utilisation

`shape.vertices`

## Description

Propriété en lecture seule ; tableau des objets Vertex (consultez la section [Objet Vertex](#)).

## Exemple

L'exemple suivant enregistre dans la variable `someShape` l'objet `item`, puis affiche le nombre de sommets de cet objet dans le panneau Sortie :

```
var someShape = fl.getDocumentDOM().selection[0];  
fl.trace("The shape has " + someShape.vertices.length + " vertices.");
```

# Objet SoundItem

**Héritage** [Objet Item](#) > objet SoundItem

## Disponibilité

Flash MX 2004.

## Description

L'objet SoundItem est une sous-classe de l'objet Item. Il représente un élément de la bibliothèque utilisé pour créer un son. Voir aussi [frame.soundLibraryItem](#) et [Objet Item](#).

## Résumé des propriétés de l'objet SoundItem

Outre les propriétés de l'[Objet Item](#), les propriétés suivantes sont disponibles avec l'objet SoundItem :

Propriété	Description
<a href="#">soundItem.bitRate</a>	Chaîne spécifiant la vitesse de transfert d'un son figurant dans la bibliothèque. Uniquement disponible pour la compression MP3.
<a href="#">soundItem.bits</a>	Chaîne spécifiant la vitesse de transfert d'un son enregistré avec compression en ADPCM et figurant dans la bibliothèque.
<a href="#">soundItem.compressionType</a>	Chaîne spécifiant le type de compression d'un son figurant dans la bibliothèque.
<a href="#">soundItem.convertStereoToMono</a>	Valeur booléenne uniquement disponible pour les types de compression MP3 et Raw.
<a href="#">soundItem.quality</a>	Chaîne spécifiant la qualité de lecture d'un son figurant dans la bibliothèque. Uniquement disponible pour la compression MP3.
<a href="#">soundItem.sampleRate</a>	Chaîne spécifiant la fréquence d'échantillonnage du clip audio.
<a href="#">soundItem.useImportedMP3Quality</a>	Valeur booléenne ; si elle vaut <code>true</code> , toutes les autres propriétés sont ignorées, et la qualité du fichier MP3 importé est utilisée.

# soundItem.bitRate

## Disponibilité

Flash MX 2004.

## Utilisation

`soundItem.bitRate`

## Description

Propriété ; chaîne spécifiant la vitesse de transfert d'un son figurant dans la bibliothèque. Cette propriété est uniquement disponible pour la compression MP3. Les valeurs gérées sont "8 Kbits/s", "16 Kbits/s", "20 Kbits/s", "24 Kbits/s", "32 Kbits/s", "48 Kbits/s", "56 Kbits/s", "64 Kbits/s", "80 Kbits/s", "112 Kbits/s", "128 Kbits/s" et "160 Kbits/s". Les sons stéréo exportés à 8 ou 16 Kbits/s sont convertis en mono. Cette propriété a la valeur `undefined` pour les autres types de compression.

### REMARQUE

Si vous avez l'intention de spécifier une valeur pour cette propriété, réglez `soundItem.useImportedMP3Quality` sur `false`.

## Exemple

L'exemple suivant affiche la valeur de la propriété `bitRate` dans le panneau de sortie si l'élément spécifié possède le type de compression MP3 :

```
alert(fl.getDocumentDOM().library.items[0].bitRate);
```

## Voir aussi

[soundItem.compressionType](#), [soundItem.convertStereoToMono](#)

# soundItem.bits

## Disponibilité

Flash MX 2004.

## Utilisation

`soundItem.bits`

## Description

Propriété ; chaîne spécifiant la vitesse de transfert d'un son enregistré avec compression ADPCM et figurant dans la bibliothèque. Les valeurs gérées sont les suivantes : "2 bit", "3 bit", "4 bit" et "5 bit".

REMARQUE

Si vous avez l'intention de spécifier une valeur pour cette propriété, réglez `soundItem.useImportedMP3Quality` sur `false`.

## Exemple

L'exemple suivant affiche le nombre de bits de transfert dans le panneau de sortie si l'élément actuellement sélectionné dans la bibliothèque possède le type de compression ADPCM :

```
alert(fl.getDocumentDOM().library.items[0].bits);
```

## Voir aussi

[soundItem.compressionType](#)

# soundItem.compressionType

## Disponibilité

Flash MX 2004.

## Utilisation

`soundItem.compressionType`

## Description

Propriété ; chaîne spécifiant le type de compression d'un son figurant dans la bibliothèque. Les valeurs possibles sont les suivantes : "Default", "ADPCM", "MP3", "Raw" (brut) et "Speech" (parole).

REMARQUE

Si vous avez l'intention de spécifier une valeur pour cette propriété, réglez `soundItem.useImportedMP3Quality` sur `false`.

## Exemple

L'exemple suivant modifie le type de compression d'un élément de la bibliothèque en lui donnant la valeur « Raw » :

```
fl.getDocumentDOM().library.items[0].compressionType = "Raw";
```

L'exemple suivant modifie le type de compression d'un élément sélectionné en lui donnant la valeur « Speech » :

```
fl.getDocumentDOM().library.getSelectedItems()[0].compressionType =  
    "Speech";
```

## soundItem.convertStereoToMono

### Disponibilité

Flash MX 2004.

### Utilisation

```
soundItem.convertStereoToMono
```

### Description

Propriété ; valeur booléenne uniquement disponible pour les types de compression MP3 et Raw. Si la valeur de cette propriété est `true`, les sons stéréo sont convertis en mono ; si elle vaut `false` le son stéréo est préservé. Pour le type de compression MP3, si `soundItem.bitRate` est inférieur à 20 Kbps/s, cette propriété n'est pas prise en compte et prend la valeur `true` (consultez la section [soundItem.bitRate](#)).

#### REMARQUE

Si vous avez l'intention de spécifier une valeur pour cette propriété, réglez `soundItem.useImportedMP3Quality` sur `false`.

### Exemple

L'exemple suivant convertit en mono un élément de la bibliothèque si cet élément possède le type de compression MP3 ou Raw :

```
fl.getDocumentDOM().library.items[0].convertStereoToMono = true;
```

### Voir aussi

[soundItem.compressionType](#)

## soundItem.quality

### Disponibilité

Flash MX 2004.

### Utilisation

`soundItem.quality`

### Description

Propriété ; chaîne spécifiant la qualité de lecture d'un son figurant dans la bibliothèque. Cette propriété est uniquement disponible pour la compression MP3. Les valeurs gérées sont les suivantes : "Fast" (rapide), "Medium" (moyenne) et "Best" (optimale).

REMARQUE

Si vous avez l'intention de spécifier une valeur pour cette propriété, réglez `soundItem.useImportedMP3Quality` sur `false`.

### Exemple

L'exemple suivant définit comme « Best » la qualité de lecture d'un élément de la bibliothèque si cet élément possède le type de compression MP3 :

```
fl.getDocumentDOM().library.items[0].quality = "Best";
```

### Voir aussi

[soundItem.compressionType](#)

## soundItem.sampleRate

### Disponibilité

Flash MX 2004.

### Utilisation

`soundItem.sampleRate`

## Description

Propriété ; chaîne spécifiant la fréquence d'échantillonnage du clip audio. Cette propriété est réservée aux types de compression ADPCM, Raw et Speech. Les valeurs gérées sont "5 kHz", "11 kHz", "22 kHz" et "44 kHz".

### REMARQUE

Si vous avez l'intention de spécifier une valeur pour cette propriété, réglez `soundItem.useImportedMP3Quality` sur `false`.

## Exemple

L'exemple suivant définit le taux d'échantillonnage d'un élément de la bibliothèque sur 5 kHz si cet élément possède le type de compression ADPCM, Raw ou Speech :

```
fl.getDocumentDOM().library.items[0].sampleRate = "5 kHz";
```

## Voir aussi

[soundItem.compressionType](#)

# soundItem.useImportedMP3Quality

## Disponibilité

Flash MX 2004.

## Utilisation

```
soundItem.useImportedMP3Quality
```

## Description

Propriété ; valeur booléenne. Si elle vaut `true`, toutes les autres propriétés sont ignorées et la qualité du fichier MP3 importé est utilisée.

## Exemple

L'exemple suivant modifie la qualité d'un élément de la bibliothèque en utilisant la qualité du fichier MP3 importé :

```
fl.getDocumentDOM().library.items[0].useImportedMP3Quality = true;
```

## Voir aussi

[soundItem.compressionType](#)



# Objet Stroke

## Disponibilité

Flash MX 2004.

## Description

L'objet Stroke contient tous les paramètres d'un trait, y compris les paramètres personnalisés. Cet objet représente les informations contenues dans l'inspecteur Propriétés. L'utilisation conjointe de l'objet Stroke et de la méthode `document.setCustomStroke()` permet de modifier les paramètres de trait du panneau Outils, de l'Inspecteur de propriétés et de la sélection actuelle. Il est également possible de lire les paramètres de trait du panneau Outils, de l'Inspecteur de propriétés ou de la sélection actuelle à l'aide de la méthode `document.getCustomStroke()`.

Cet objet possède toujours les quatre propriétés suivantes : `style`, `thickness` (épaisseur), `color` (couleur) et `breakAtCorners` (angles saillants). (Dans Flash CS3, la propriété `stroke.joinType` est préférée à `breakAtCorners`.) Il est possible de définir d'autres propriétés en fonction de la valeur de la propriété `stroke.style`.

## Résumé des propriétés de l'objet Stroke

Les propriétés suivantes sont disponibles pour l'objet Stroke :

Propriété	Description
<code>stroke.breakAtCorners</code>	Valeur booléenne, identique au paramètre « Angles saillants » de la boîte de dialogue personnalisée Style de trait.
<code>stroke.capType</code>	Chaîne qui spécifie le type d'extrémité du trait.
<code>stroke.color</code>	Valeur hexadécimale ou entier représentant la couleur du trait.
<code>stroke.curve</code>	Chaîne spécifiant le type de trame du trait.
<code>stroke.dash1</code>	Entier spécifiant la longueur de la partie pleine d'une ligne pointillée.
<code>stroke.dash2</code>	Entier spécifiant la longueur de la partie vide d'une ligne pointillée.
<code>stroke.density</code>	Chaîne spécifiant la densité d'une ligne pointillée fine.
<code>stroke.dotSize</code>	Chaîne spécifiant la taille des points d'une ligne pointillée fine.
<code>stroke.dotSpace</code>	Entier spécifiant l'espacement des points d'une ligne pointillée.
<code>stroke.hatchThickness</code>	Chaîne spécifiant l'épaisseur d'une ligne tramée.

Propriété	Description
<code>stroke.jiggle</code>	Chaîne spécifiant la propriété de déplacement d'une ligne tramée.
<code>stroke.joinType</code>	Chaîne qui spécifie le type de jointure du trait.
<code>stroke.length</code>	Chaîne spécifiant la longueur d'une ligne tramée.
<code>stroke.miterLimit</code>	Nombre en virgule flottante spécifiant l'angle au-dessus duquel l'extrémité de la pointe sera tronquée par un segment.
<code>stroke.pattern</code>	Chaîne spécifiant le motif d'une ligne irrégulière.
<code>stroke.rotate</code>	Chaîne spécifiant la rotation d'une ligne tramée.
<code>stroke.scaleType</code>	Chaîne spécifiant le type de redimensionnement à appliquer au trait.
<code>stroke.shapeFill</code>	<b>Objet Fill</b> représentant les paramètres de remplissage du trait.
<code>stroke.space</code>	Chaîne spécifiant l'espacement d'une ligne tramée.
<code>stroke.strokeHinting</code>	Valeur booléenne indiquant si les repères de trait doivent être activés pour ce trait.
<code>stroke.style</code>	Chaîne décrivant le style de trait.
<code>stroke.thickness</code>	Nombre entier spécifiant la taille de trait.
<code>stroke.variation</code>	Chaîne spécifiant la variation d'une ligne pointillée fine.
<code>stroke.waveHeight</code>	Chaîne spécifiant la hauteur d'onde d'une ligne irrégulière.
<code>stroke.waveLength</code>	Chaîne spécifiant la longueur d'onde d'une ligne irrégulière.

## stroke.breakAtCorners

### Disponibilité

Flash MX 2004. Délaissée dans Flash CS3 au profit de `stroke.joinType`.

### Utilisation

`stroke.breakAtCorners`

### Description

Propriété ; valeur booléenne. Cette propriété est identique au paramètre « Angles saillants » de la boîte de dialogue Style de trait.

### Exemple

L'exemple suivant donne à la propriété `breakAtCorners` la valeur `true` :

```
var myStroke = fl.getDocumentDOM().getCustomStroke();
myStroke.breakAtCorners = true;
fl.getDocumentDOM().setCustomStroke(myStroke);
```

# stroke.capType

## Disponibilité

Flash 8.

## Utilisation

`stroke.capType`

## Description

Propriété ; chaîne qui spécifie le type d'extrémité du trait. Les valeurs acceptables sont "none" (aucune), "round" (arrondie) et "square" (carrée).

## Exemple

L'exemple suivant donne la valeur "round" au type d'extrémité du trait :

```
var myStroke = fl.getDocumentDOM().getCustomStroke();
myStroke.capType = "round";
fl.getDocumentDOM().setCustomStroke(myStroke);
```

# stroke.color

## Disponibilité

Flash MX 2004. Dans Flash 8 et les versions ultérieures, cette propriété est délaissée au profit de `stroke.shapeFill.color`.

## Utilisation

`stroke.color`

## Description

Propriété ; couleur du trait, exprimée dans l'un des formats suivants :

- Chaîne au format "#RRVBBB" ou "#RRVBBBAA"
- Nombre hexadécimal au format 0xRRVBBB
- Entier représentant l'équivalent décimal d'un nombre hexadécimal

## Exemple

L'exemple suivant définit la couleur du trait :

```
var myStroke = fl.getDocumentDOM().getCustomStroke();
myStroke.color = "#000000";
fl.getDocumentDOM().setCustomStroke(myStroke);
```

## Voir aussi

[stroke.shapeFill](#)

# stroke.curve

## Disponibilité

Flash MX 2004.

## Utilisation

stroke.curve

## Description

Propriété ; chaîne spécifiant le type de trame du trait. Cette propriété ne peut être définie que si la propriété `stroke.style` est définie sur "hatched" (consultez la section [stroke.style](#)). Les valeurs possibles sont les suivantes : "straight" (droite), "slight curve" (courbe légère), "medium curve" (courbe moyenne) et "very curved" (courbe très forte).

## Exemple

L'exemple suivant définit la propriété de courbe, entre autres, d'un trait ayant le style "hatched" :

```
var myStroke = fl.getDocumentDOM().getCustomStroke();
myStroke.style = "hatched";
myStroke.curve = "straight";
myStroke.space = "close";
myStroke.jiggle = "wild";
myStroke.rotate = "free";
myStroke.length = "slight";
myStroke.hatchThickness = "thin";
fl.getDocumentDOM().setCustomStroke(myStroke);
```

# stroke.dash1

## Disponibilité

Flash MX 2004.

## Utilisation

stroke.dash1

## Description

Propriété ; entier spécifiant la longueur de la partie pleine d'une ligne pointillée. Cette propriété n'est disponible que si la propriété `stroke.style` est réglée sur "dashed" (consultez la section [stroke.style](#)).

## Exemple

L'exemple suivant définit les propriétés `dash1` et `dash2` pour le style de trait `dashed` :

```
var myStroke = fl.getDocumentDOM().getCustomStroke();
myStroke.style = "dashed";
myStroke.dash1 = 1;
myStroke.dash2 = 2;
fl.getDocumentDOM().setCustomStroke(myStroke);
```

# stroke.dash2

## Disponibilité

Flash MX 2004.

## Utilisation

`stroke.dash2`

## Description

Propriété ; entier spécifiant la longueur de la partie vide d'une ligne pointillée.

Cette propriété n'est disponible que si la propriété `stroke.style` est réglée sur "dashed" (voir [stroke.style](#)).

## Exemple

Voir [stroke.dash1](#).

# stroke.density

## Disponibilité

Flash MX 2004.

## Utilisation

`stroke.density`

## Description

Propriété ; chaîne spécifiant la densité d'une ligne pointillée fine. Cette propriété n'est disponible que si la propriété `stroke.style` est réglée sur "stipple" (consultez la section [stroke.style](#)). Les valeurs possibles sont les suivantes : "very dense", "dense", "sparse" (clairsemée) et "very sparse" (très clairsemée).

## Exemple

L'exemple suivant définit comme "sparse" la propriété de densité du style de trait stipple :

```
var myStroke = fl.getDocumentDOM().getCustomStroke();
myStroke.style = "stipple";
myStroke.dotSpace= 3;
myStroke.variation = "random sizes";
myStroke.density = "sparse";
fl.getDocumentDOM().setCustomStroke(myStroke);
```

## stroke.dotSize

### Disponibilité

Flash MX 2004.

### Utilisation

stroke.dotSize

### Description

Propriété ; chaîne spécifiant la taille des points d'une ligne pointillée fine. Cette propriété n'est disponible que si la propriété stroke.style est réglée sur "stipple" (consultez la section [stroke.style](#)). Les valeurs possibles sont les suivantes : "tiny" (minuscule), "small" (petite), "medium" (moyenne) et "large".

L'exemple suivant définit comme "tiny" la propriété dotSize du style de trait stipple :

```
var myStroke = fl.getDocumentDOM().getCustomStroke();
myStroke.style = "stipple";
myStroke.dotSpace= 3;
myStroke.dotsize = "tiny";
myStroke.variation = "random sizes";
myStroke.density = "sparse";
fl.getDocumentDOM().setCustomStroke(myStroke);
```

## stroke.dotSpace

### Disponibilité

Flash MX 2004.

### Utilisation

stroke.dotSpace

## Description

Propriété ; entier spécifiant l'espacement des points d'une ligne pointillée. Cette propriété n'est disponible que si la propriété `stroke.style` a pour valeur "dotted". Voir la section [stroke.style](#).

## Exemple

L'exemple suivant donne la valeur 3 à la propriété `dotSpace` pour le style de trait `dotted` :

```
var myStroke = fl.getDocumentDOM().getCustomStroke();
myStroke.style = "dotted";
myStroke.dotSpace = 3;
fl.getDocumentDOM().setCustomStroke(myStroke);
```

# stroke.hatchThickness

## Disponibilité

Flash MX 2004.

## Utilisation

`stroke.hatchThickness`

## Description

Propriété ; chaîne spécifiant l'épaisseur d'une ligne tramée. Cette propriété n'est disponible que si la propriété `stroke.style` est réglée sur "hatched" (consultez la section [stroke.style](#)). Les valeurs possibles sont les suivantes : "hairline" (ultrafine), "thin" (fine), "medium" (moyenne) et "thick" (épaisse).

## Exemple

L'exemple suivant donne la valeur "thin" à la propriété `hatchThickness` du style de trait `hatched` :

```
var myStroke = fl.getDocumentDOM().getCustomStroke();
myStroke.style = "hatched";
myStroke.curve = "straight";
myStroke.space = "close";
myStroke.jiggle = "wild";
myStroke.rotate = "free";
myStroke.length = "slight";
myStroke.hatchThickness = "thin";
fl.getDocumentDOM().setCustomStroke(myStroke);
```

# stroke.jiggle

## Disponibilité

Flash MX 2004.

## Utilisation

stroke.jiggle

## Description

Propriété ; chaîne spécifiant la propriété de déplacement d'une ligne tramée. Cette propriété n'est disponible que si la propriété `stroke.style` est réglée sur "hatched" (consultez la section [stroke.style](#)). Les valeurs possibles sont les suivantes : "none" (aucun), "bounce" (rebond), "loose" (espacé) et "wild" (désordonné).

## Exemple

L'exemple suivant donne la valeur "wild" à la propriété `jiggle` du style de trait `hatched` :

```
var myStroke = fl.getDocumentDOM().getCustomStroke();
myStroke.style = "hatched";
myStroke.curve = "straight";
myStroke.space = "close";
myStroke.jiggle = "wild";
myStroke.rotate = "free";
myStroke.length = "slight";
myStroke.hatchThickness = "thin";
fl.getDocumentDOM().setCustomStroke(myStroke);
```

# stroke.joinType

## Disponibilité

Flash 8.

## Utilisation

stroke.joinType

## Description

Propriété ; chaîne qui spécifie le type de jointure du trait. Les valeurs gérées sont "miter" (pointe), "round" (arrondie) et "bevel" (biseau).

## Voir aussi

[stroke.capType](#)



# stroke.length

## Disponibilité

Flash MX 2004.

## Utilisation

`stroke.length`

## Description

Propriété ; chaîne spécifiant la longueur d'une ligne tramée. Cette propriété n'est disponible que si la propriété `stroke.style` est réglée sur "hatched" (consultez la section [stroke.style](#)). Les valeurs possibles sont les suivantes : "equal" (égale), "slight" (légère), "variation" (variation légère), "medium variation" (variation moyenne) et "random" (aléatoire).

## Exemple

L'exemple suivant donne la valeur "slight" à la propriété `length` du style de trait hatched :

```
var myStroke = fl.getDocumentDOM().getCustomStroke();
myStroke.style = "hatched";
myStroke.curve = "straight";
myStroke.space = "close";
myStroke.jiggle = "wild";
myStroke.rotate = "free";
myStroke.length = "slight";
myStroke.hatchThickness = "thin";
fl.getDocumentDOM().setCustomStroke(myStroke);
```

# stroke.miterLimit

## Disponibilité

Flash 8.

## Utilisation

`stroke.miterLimit`

## Description

Propriété ; nombre en virgule flottante spécifiant l'angle au-dessus duquel l'extrémité de la pointe sera tronquée par un segment. Cela signifie que la pointe ne sera tronquée que si son angle est supérieur à la valeur transmise dans `miterLimit`.

## Exemple

L'exemple suivant donne la valeur 3 à la limite de pointe du trait. Si l'angle de la pointe est supérieur à 3, la pointe est tronquée.

```
var myStroke = fl.getDocumentDOM().getCustomStroke();  
myStroke.miterLimit = 3;  
var myStroke = fl.getDocumentDOM().setCustomStroke();
```

## stroke.pattern

### Disponibilité

Flash MX 2004.

### Utilisation

stroke.pattern

### Description

Propriété ; chaîne spécifiant le motif d'une ligne irrégulière. Cette propriété n'est disponible que si la propriété `stroke.style` est réglée sur "ragged" (consultez la section [stroke.style](#)). Les valeurs possibles sont les suivantes : "solid" (continu), "simple", "random" (aléatoire), "dotted" (pointillé), "random dotted" (pointillé aléatoire), "triple dotted" (pointillé triple) et "random triple dotted" (pointillé triple aléatoire).

### Exemple

L'exemple suivant donne la valeur "random" à la propriété `pattern` du style de trait `ragged` :

```
var myStroke = fl.getDocumentDOM().getCustomStroke();  
myStroke.style = "ragged";  
myStroke.pattern = "random";  
fl.getDocumentDOM().setCustomStroke(myStroke);
```

# stroke.rotate

## Disponibilité

Flash MX 2004.

## Utilisation

`stroke.rotate`

## Description

Propriété ; chaîne spécifiant la rotation d'une ligne tramée. Cette propriété n'est disponible que si la propriété `stroke.style` est réglée sur "hatched" (consultez la section [stroke.style](#)). Les valeurs possibles sont les suivantes : "none" (aucune), "slight" (légère), "medium" (moyenne) et "free" (libre)

## Exemple

L'exemple suivant donne la valeur "free" à la propriété `rotate` du style de trait `hatched` :

```
var myStroke = fl.getDocumentDOM().getCustomStroke();
myStroke.style = "hatched";
myStroke.curve = "straight";
myStroke.space = "close";
myStroke.jiggle = "wild";
myStroke.rotate = "free";
myStroke.length = "slight";
myStroke.hatchThickness = "thin";
```

# stroke.scaleType

## Disponibilité

Flash 8.

## Utilisation

`stroke.scaleType`

## Description

Propriété ; chaîne qui spécifie le type de redimensionnement à appliquer au trait. Les valeurs gérées sont "normal", "horizontal", "vertical" et "none".

## Exemple

L'exemple suivant donne au type de dimensionnement du trait la valeur "horizontal" :

```
var myStroke = fl.getDocumentDOM().getCustomStroke();
myStroke.scaleType = "horizontal";
fl.getDocumentDOM().setCustomStroke(myStroke);
```

# stroke.shapeFill

## Disponibilité

Flash 8.

## Utilisation

stroke.shapeFill

## Description

Propriété ; [Object Fill](#) représentant les paramètres de remplissage du trait.

## Exemple

L'exemple suivant spécifie les paramètres de remplissage et les applique au trait :

```
var fill = fl.getDocumentDOM().getCustomFill();
fill.linearGradient = true;
fill.colorArray = [ 00ff00, ff0000, ffffff ];
var stroke = fl.getDocumentDOM().getCustomStroke();
stroke.shapeFill = fill;
fl.getDocumentDOM().setCustomStroke(stroke);
```

# stroke.space

## Disponibilité

Flash MX 2004.

## Utilisation

stroke.space

## Description

Propriété ; chaîne spécifiant l'espacement d'une ligne tramée. Cette propriété n'est disponible que si la propriété `stroke.style` est réglée sur "hatched" (consultez la section [stroke.style](#)). Les valeurs possibles sont les suivantes : "very close" (très faible), "close" (faible), "distant" (important) et "very distant" (très important).

## Exemple

L'exemple suivant donne la valeur "close" à la propriété `space` du style de trait hatched :

```
var myStroke = fl.getDocumentDOM().getCustomStroke();
myStroke.style = "hatched";
myStroke.curve = "straight";
myStroke.space = "close";
myStroke.jiggle = "wild";
myStroke.rotate = "free";
myStroke.length = "slight";
myStroke.hatchThickness = "thin";
fl.getDocumentDOM().setCustomStroke(myStroke);
```

# stroke.strokeHinting

## Disponibilité

Flash 8.

## Utilisation

`stroke.strokeHinting`

## Description

Propriété ; valeur booléenne indiquant si les repères de trait doivent être activés pour ce trait.

## Exemple

L'exemple suivant active les repères de trait pour ce trait :

```
var myStroke = fl.getDocumentDOM().getCustomStroke();
myStroke.strokeHinting = true;
fl.getDocumentDOM().setCustomStroke(myStroke);
```

# stroke.style

## Disponibilité

Flash MX 2004.

## Utilisation

`stroke.style`

## Description

Propriété ; chaîne décrivant le style de trait. Les valeurs possibles sont les suivantes : "noStroke", (pas de trait), "solid" (uni), "dashed" (discontinu), "dotted" (pointillé), "ragged" (irrégulier), "stipple" (pointillé fin) et "hatched" (hachuré). Certaines de ces valeurs nécessitent que des propriétés complémentaires de l'objet Stroke soient définies, comme décrit dans la liste suivante :

- Si cette propriété a la valeur "solid" ou "noStroke", il n'existe pas d'autre propriété.
- Si cette propriété a la valeur "dashed", il existe deux propriétés complémentaires : "dash1" et "dash2".
- Si cette propriété a la valeur "dotted", il existe une propriété complémentaire : "dotSpace".
- Si cette propriété a la valeur "ragged", il existe trois propriétés complémentaires : "pattern", "waveHeight" et "waveLength".

- Si cette propriété a la valeur "stipple", il existe trois propriétés complémentaires : "dotSize", "variation" et "density".
- Si cette propriété a la valeur "hatched", il existe six propriétés complémentaires : "hatchThickness", "space", "jiggle", "rotate", "curve" et "length".

### Exemple

L'exemple suivant donne la valeur "ragged" au style de trait :

```
var myStroke = fl.getDocumentDOM().getCustomStroke();
myStroke.style = "ragged";
fl.getDocumentDOM().setCustomStroke(myStroke);
```

## stroke.thickness

### Disponibilité

Flash MX 2004.

### Utilisation

stroke.thickness

### Description

Propriété ; nombre entier spécifiant la taille de trait.

### Exemple

L'exemple suivant donne la valeur 2 à la propriété thickness du trait :

```
var myStroke = fl.getDocumentDOM().getCustomStroke();
myStroke.thickness = 2;
fl.getDocumentDOM().setCustomStroke(myStroke);
```

## stroke.variation

### Disponibilité

Flash MX 2004.

### Utilisation

stroke.variation

## Description

Propriété ; chaîne spécifiant la variation d'une ligne pointillée fine. Cette propriété n'est disponible que si la propriété `stroke.style` est réglée sur "stipple" (consultez la section [stroke.style](#)). Les valeurs possibles sont les suivantes : "one size" (taille unique), "small variation" (variation minime), "varied sizes" (différentes tailles) et "random sizes" (tailles aléatoires).

## Exemple

L'exemple suivant donne la valeur "random sizes" à la propriété de variation du style de trait stipple :

```
var myStroke = fl.getDocumentDOM().getCustomStroke();
myStroke.style = "stipple";
myStroke.dotSpace= 3;
myStroke.variation = "random sizes";
myStroke.density = "sparse";
fl.getDocumentDOM().setCustomStroke(myStroke);
```

# stroke.waveHeight

## Disponibilité

Flash MX 2004.

## Utilisation

`stroke.waveHeight`

## Description

Propriété ; chaîne spécifiant la hauteur d'onde d'une ligne irrégulière. Cette propriété n'est disponible que si la propriété `stroke.style` est réglée sur "ragged" (consultez la section [stroke.style](#)). Les valeurs possibles sont les suivantes : "flat" (régulière), "wavy" (ondulée), "very wavy" (très ondulée) et "wild" (désordonnée).

## Exemple

L'exemple suivant donne la valeur "flat" à la propriété `waveHeight` du style de trait `ragged` :

```
var myStroke = fl.getDocumentDOM().getCustomStroke();
myStroke.style = "ragged";
myStroke.pattern = "random";
myStroke.waveHeight = "flat";
myStroke.waveLength = "short";
fl.getDocumentDOM().setCustomStroke(myStroke);
```

# stroke.waveLength

## Disponibilité

Flash MX 2004.

## Utilisation

stroke.waveLength

## Description

Propriété ; chaîne spécifiant la longueur d'onde d'une ligne irrégulière. Cette propriété n'est disponible que si la propriété `stroke.style` est réglée sur "ragged" (consultez la section [stroke.style](#)). Les valeurs possibles sont les suivantes : "very short" (très courte), "short" (courte), "medium" (moyenne) et "long" (longue).

## Exemple

L'exemple suivant donne la valeur "short" à la propriété `waveLength` du style de trait `ragged` :

```
var myStroke = fl.getDocumentDOM().getCustomStroke();
myStroke.style = "ragged";
myStroke.pattern = "random";
myStroke.waveHeight = 'flat';
myStroke.waveLength = "short";
fl.getDocumentDOM().setCustomStroke(myStroke);
```



# Objet SymbolInstance

**Héritage** [Objet Element](#) > [Objet Instance](#) > objet SymbolInstance

## Disponibilité

Flash MX 2004.

## Description

L'objet SymbolInstance est une sous-classe de l'objet Instance et représente un symbole dans une image (consultez la section [Objet Instance](#)).

## Résumé des propriétés de l'objet SymbolInstance

Outre les propriétés de l'[Objet Instance](#), l'objet SymbolInstance possède les propriétés suivantes :

Propriété	Description
<a href="#">symbolInstance.accName</a>	Chaîne équivalente au champ Nom du panneau Accessibilité.
<a href="#">symbolInstance.actionScript</a>	Chaîne qui spécifie les actions affectées au symbole.
<a href="#">symbolInstance.blendMode</a>	Chaîne qui spécifie le mode de fondu à appliquer à un symbole de clip.
<a href="#">symbolInstance.buttonTracking</a>	Chaîne qui définit, pour les symboles de bouton uniquement, la même propriété que le menu contextuel Traiter comme bouton ou Traiter comme élément de menu dans l'inspecteur des propriétés.
<a href="#">symbolInstance.cacheAsBitmap</a>	Valeur booléenne spécifiant si la mise en cache des bitmaps à l'exécution est activée ou non.
<a href="#">symbolInstance.colorAlphaAmount</a>	Entier spécifiant une partie de la transformation de couleur de l'occurrence, en indiquant la valeur des paramètres avancés de l'effet Alpha ; équivalent à l'utilisation de l'entrée de menu Couleur > Paramètres avancés de l'inspecteur des propriétés, suivie du réglage des contrôles qui apparaissent à droite dans la boîte de dialogue.

Propriété	Description
<code>symbolInstance.colorAlphaPercent</code>	Entier spécifiant certaines parties de la transformation de couleur de l'occurrence ; équivalent à l'utilisation de l'entrée de menu Couleur > Paramètres avancés de l'inspecteur des propriétés, suivie du réglage des contrôles de pourcentage qui apparaissent à gauche dans la boîte de dialogue.
<code>symbolInstance.colorBlueAmount</code>	Entier spécifiant une partie de la transformation de couleur de l'occurrence ; équivalent à l'utilisation de l'entrée de menu Couleur > Paramètres avancés de l'inspecteur des propriétés.
<code>symbolInstance.colorBluePercent</code>	Entier spécifiant une partie de la transformation de couleur de l'occurrence ; équivalent à l'utilisation de l'entrée de menu Couleur > Paramètres avancés de l'inspecteur des propriétés, suivie du réglage des contrôles de pourcentage qui apparaissent à gauche dans la boîte de dialogue.
<code>symbolInstance.colorGreenAmount</code>	Entier spécifiant une partie de la transformation de couleur de l'occurrence ; équivalent à l'utilisation de l'entrée de menu Couleur > Paramètres avancés de l'inspecteur des propriétés. Les valeurs autorisées sont comprises entre -255 et 255.
<code>symbolInstance.colorGreenPercent</code>	Partie de la transformation de couleur de l'occurrence ; équivalent à l'utilisation de l'entrée de menu Couleur > Paramètres avancés de l'inspecteur des propriétés, suivie du réglage des contrôles de pourcentage qui apparaissent à gauche dans la boîte de dialogue.
<code>symbolInstance.colorMode</code>	Chaîne spécifiant le mode colorimétrique tel qu'il est identifié dans le menu contextuel Couleurs de l'inspecteur des propriétés du symbole.
<code>symbolInstance.colorRedAmount</code>	Entier spécifiant une partie de la transformation de couleur de l'occurrence ; équivalent à l'utilisation de l'entrée de menu Couleur > Paramètres avancés de l'inspecteur des propriétés.
<code>symbolInstance.colorRedPercent</code>	Partie de la transformation de couleur de l'occurrence ; équivalent à l'utilisation de l'entrée de menu Couleur > Paramètres avancés de l'inspecteur des propriétés, suivie du réglage des contrôles de pourcentage qui apparaissent à gauche dans la boîte de dialogue.
<code>symbolInstance.description</code>	Chaîne équivalente au champ Description du panneau Accessibilité.

Propriété	Description
<code>symbolInstance.filters</code>	Tableau des objets Filter (consultez la section <a href="#">Objet Filter</a> ).
<code>symbolInstance.firstFrame</code>	Entier en base zéro indiquant la première image qui doit apparaître dans le scénario du graphisme.
<code>symbolInstance.forceSimple</code>	Valeur booléenne qui active et désactive l'accessibilité aux enfants de l'objet ; équivalente à l'inverse de la fonction « Rendre les objets enfants accessibles » dans le panneau Accessibilité.
<code>symbolInstance.loop</code>	Chaîne qui définit, pour les symboles graphiques, la même propriété que le menu contextuel Boucle de l'inspecteur des propriétés.
<code>symbolInstance.shortcut</code>	Chaîne équivalente à la touche de raccourci associée au symbole, ainsi qu'au champ Raccourci dans le panneau Accessibilité.
<code>symbolInstance.silent</code>	Valeur booléenne qui active et désactive l'accessibilité à l'objet ; équivalente à l'inverse de la fonction Rendre l'objet accessible dans le panneau Accessibilité.
<code>symbolInstance.symbolType</code>	Chaîne spécifiant le type de symbole, équivalente de la valeur du champ Comportement dans les boîtes de dialogue Créer un nouveau symbole et Convertir en symbole.
<code>symbolInstance.tabIndex</code>	Entier équivalent au champ Index de tabulation du panneau Accessibilité.

# symbolInstance.accName

## Disponibilité

Flash MX 2004.

## Utilisation

`symbolInstance.accName`

## Description

Propriété ; chaîne équivalente au champ Nom du panneau Accessibilité. Les logiciels de lecture vocale identifient l'objet en prononçant ce nom. Cette propriété n'est pas disponible pour les symboles graphiques.

## Exemple

L'exemple suivant enregistre dans la variable `theName` la valeur du nom de l'objet dans le panneau Accessibilité :

```
var theName = fl.getDocumentDOM().selection[0].accName;
```

L'exemple suivant donne la valeur "Home Button" au nom de l'objet dans le panneau Accessibilité :

```
fl.getDocumentDOM().selection[0].accName = "Home Button";
```

# symbolInstance.actionScript

## Disponibilité

Flash MX 2004.

## Utilisation

`symbolInstance.actionScript`

## Description

Propriété ; chaîne qui spécifie les actions affectées au symbole. Elle ne s'applique qu'aux occurrences de clips et de boutons. Pour une occurrence de symbole graphique, la valeur renvoyée est `undefined`.

## Exemple

L'exemple suivant affecte une action `onClipEvent` au premier élément de la première image du premier calque du scénario :

```
fl.getDocumentDOM().getTimeline().layers[0].frames[0].elements[0].actionScript  
= "onClipEvent(enterFrame) {trace('movie clip enterFrame');}";
```

# symbolInstance.blendMode

## Disponibilité

Flash 8.

## Utilisation

`symbolInstance.blendMode`

## Description

Propriété ; chaîne qui spécifie le mode de fondu à appliquer à un symbole de clip. Les valeurs gérées sont "normal", "layer", "multiply", "screen", "overlay", "hardlight", "lighten", "darken", "difference", "add", "subtract", "invert", "alpha" et "erase".

## Exemple

L'exemple suivant règle le mode de fondu du premier symbole de clip dans la première image du premier niveau sur "add" :

```
fl.getDocumentDOM().getTimeline().layers[0].frames[0].elements[0].blendMode  
= "add";
```

## Voir aussi

[document.setBlendMode\(\)](#)

# symbolInstance.buttonTracking

## Disponibilité

Flash MX 2004.

## Utilisation

`symbolInstance.buttonTracking`

## Description

Propriété ; chaîne qui définit, pour les symboles de bouton uniquement, la même propriété que le menu contextuel Traiter comme bouton ou Traiter comme élément de menu dans l'inspecteur des propriétés. Pour les autres types de symboles, cette propriété est ignorée. Les valeurs gérées sont "button" ou "menu".

## Exemple

L'exemple suivant donne au premier symbole de la première image du premier calque du scénario la valeur Traiter comme élément de menu, dans la mesure où ce symbole est un bouton :

```
fl.getDocumentDOM().getTimeline().layers[0].frames[0].elements[0].buttonTra  
cking = "menu";
```

# symbolInstance.cacheAsBitmap

## Disponibilité

Flash 8.

## Utilisation

`symbolInstance.cacheAsBitmap`

## Description

Propriété ; valeur booléenne spécifiant si la mise en cache des bitmaps à l'exécution est activée ou non.

## Exemple

L'exemple suivant active la mise en cache des bitmaps à l'exécution pour le premier élément de la première image du premier calque :

```
fl.getDocumentDOM().getTimeline().layers[0].frames[0].elements[0].cacheAsBi  
tmap = true;
```

# symbolInstance.colorAlphaAmount

## Disponibilité

Flash MX 2004.

## Utilisation

`symbolInstance.colorAlphaAmount`

## Description

Propriété ; entier spécifiant une partie de la transformation de couleur de l'occurrence, en indiquant la valeur des paramètres avancés de l'effet Alpha. Cette propriété équivaut à l'utilisation de l'entrée de menu Couleur > Paramètres avancés de l'inspecteur des propriétés, suivie du réglage des contrôles qui apparaissent à droite dans la boîte de dialogue. Cette valeur diminue ou augmente d'un niveau constant les valeurs d'alpha et de teinte. Cette valeur est ajoutée à la valeur actuelle. Cette propriété est surtout utile en combinaison avec [symbolInstance.colorAlphaPercent](#). Les valeurs autorisées sont comprises entre -255 et 255.

## Exemple

L'exemple suivant soustrait la valeur 100 du paramètre alpha de l'occurrence de symbole sélectionnée :

```
fl.getDocumentDOM().selection[0].colorAlphaAmount = -100;
```

# symbolInstance.colorAlphaPercent

## Disponibilité

Flash MX 2004.

## Utilisation

`symbolInstance.colorAlphaPercent`

## Description

Propriété ; entier spécifiant une partie de la transformation de couleur de l'occurrence. Cette propriété équivaut à l'utilisation de la commande Couleur > Paramètres avancés de l'inspecteur des propriétés, suivie du réglage des contrôles de pourcentage qui apparaissent à gauche dans la boîte de dialogue. Cette valeur spécifie le nouveau pourcentage de teinte et d'alpha. Les valeurs autorisées sont comprises entre -100 et 100. Consultez également la section [symbolInstance.colorAlphaAmount](#).

## Exemple

L'exemple suivant fixe à 80 la valeur de la propriété `colorAlphaPercent` de l'occurrence de symbole sélectionnée :

```
fl.getDocumentDOM().selection[0].colorAlphaPercent = 80;
```

# symbolInstance.colorBlueAmount

## Disponibilité

Flash MX 2004.

## Utilisation

`symbolInstance.colorBlueAmount`

## Description

Propriété ; entier spécifiant une partie de la transformation de couleur de l'occurrence. Cette propriété a un résultat équivalent à celui de la commande Couleur > Paramètres avancés de l'inspecteur des propriétés de l'occurrence. Les valeurs autorisées sont comprises entre -255 et 255.

# symbolInstance.colorBluePercent

## Disponibilité

Flash MX 2004.

## Utilisation

`symbolInstance.colorBluePercent`

## Description

Propriété ; entier spécifiant une partie de la transformation de couleur de l'occurrence. Cette propriété équivaut à l'utilisation de la commande Couleur > Paramètres avancés de l'inspecteur des propriétés, suivie du réglage des contrôles de pourcentage qui apparaissent à gauche dans la boîte de dialogue. Cette valeur spécifie le nouveau pourcentage de bleu. Les valeurs autorisées sont comprises entre -100 et 100.

## Exemple

L'exemple suivant fixe à 80 la valeur de la propriété `colorBluePercent` de l'occurrence de symbole sélectionnée :

```
fl.getDocumentDOM().selection[0].colorBluePercent = 80;
```

# symbolInstance.colorGreenAmount

## Disponibilité

Flash MX 2004.

## Utilisation

`symbolInstance.colorGreenAmount`

## Description

Propriété ; entier spécifiant une partie de la transformation de couleur de l'occurrence. Cette propriété a un résultat équivalent à celui de la commande Couleur > Paramètres avancés de l'inspecteur des propriétés de l'occurrence. Les valeurs autorisées sont comprises entre -255 et 255.



# symbolInstance.colorGreenPercent

## Disponibilité

Flash MX 2004.

## Utilisation

`symbolInstance.colorGreenPercent`

## Description

Propriété ; entier spécifiant une partie de la transformation de couleur de l'occurrence. Cette propriété équivaut à l'utilisation de la commande Couleur > Paramètres avancés de l'inspecteur des propriétés, suivie du réglage des contrôles de pourcentage qui apparaissent à gauche dans la boîte de dialogue. Cette valeur spécifie le nouveau pourcentage de vert. Les valeurs autorisées sont comprises entre -100 et 100.

## Exemple

L'exemple suivant fixe à 70 la valeur de la propriété `colorGreenPercent` de l'occurrence de symbole sélectionnée :

```
fl.getDocumentDOM().selection[0].colorGreenPercent = 70;
```

# symbolInstance.colorMode

## Disponibilité

Flash MX 2004.

## Utilisation

`symbolInstance.colorMode`

## Description

Propriété ; chaîne spécifiant le mode colorimétrique tel qu'il est identifié dans le menu contextuel Couleurs de l'inspecteur des propriétés du symbole. Les valeurs gérées sont "none" (aucun), "brightness" (luminosité), "tint" (teinte), "alpha" et "advanced" (avancé).

## Exemple

L'exemple suivant donne la valeur "alpha" à la propriété `colorMode` du premier élément de la première image du premier calque du scénario :

```
fl.getDocumentDOM().getTimeline().layers[0].frames[0].elements[0].colorMode  
= "alpha";
```

# symbolInstance.colorRedAmount

## Disponibilité

Flash MX 2004.

## Utilisation

`symbolInstance.colorRedAmount`

## Description

Propriété ; entier spécifiant une partie de la transformation de couleur de l'occurrence. Cette propriété a un résultat équivalent à celui de la commande Couleur > Paramètres avancés de l'inspecteur des propriétés de l'occurrence. Les valeurs autorisées sont comprises entre -255 et 255.

## Exemple

L'exemple suivant fixe à 255 la valeur de la propriété `colorRedAmount` de l'occurrence de symbole sélectionnée :

```
fl.getDocumentDOM().selection[0].colorRedAmount = 255;
```

# symbolInstance.colorRedPercent

## Disponibilité

Flash MX 2004.

## Utilisation

`symbolInstance.colorRedPercent`

## Description

Propriété ; entier spécifiant une partie de la transformation de couleur de l'occurrence. Cette propriété équivaut à l'utilisation de la commande Couleur > Paramètres avancés de l'inspecteur des propriétés, suivie du réglage des contrôles de pourcentage qui apparaissent à gauche dans la boîte de dialogue. Cette valeur spécifie le nouveau pourcentage de rouge. Les valeurs autorisées sont comprises entre -100 et 100.

## Exemple

L'exemple suivant fixe à 10 la valeur de la propriété `colorRedPercent` de l'occurrence de symbole sélectionnée :

```
fl.getDocumentDOM().selection[0].colorRedPercent = 10;
```

# symbolInstance.description

## Disponibilité

Flash MX 2004.

## Utilisation

`symbolInstance.description`

## Description

Propriété ; chaîne équivalente au champ Description du panneau Accessibilité. Cette description est lue par les logiciels de lecture vocale. Cette propriété n'est pas disponible pour les symboles graphiques.

## Exemple

L'exemple suivant enregistre dans la variable `theDescription` la valeur de la description de l'objet dans le panneau Accessibilité :

```
var theDescription = fl.getDocumentDOM().selection[0].description;
```

L'exemple suivant affecte la chaîne "Click the home button to go to home" à la description de l'objet dans le panneau Accessibilité :

```
fl.getDocumentDOM().selection[0].description= "Click the home button to go  
to home";
```

# symbolInstance.filters

## Disponibilité

Flash 8.

## Utilisation

`symbolInstance.filters`

## Description

Propriété ; tableau des objets Filter (consultez la section [Objet Filter](#)). Pour modifier les propriétés relatives aux filtres, n'entrez pas directement de données dans ce tableau. Vous devez en effet lire le tableau, régler chaque propriété, puis définir le tableau de sorte à refléter les nouvelles propriétés.

## Exemple

L'exemple suivant recherche le nom du filtre à l'index 0. S'il s'agit d'un filtre de type Néon, la propriété `blurX` correspondante est réglée sur 100 et la nouvelle valeur est entrée dans le tableau des filtres.

```
var filterName =  
    fl.getDocumentDOM().getTimeline().layers[0].frames[0].elements[0].filters[0].name;  
fl.trace(filterName);  
var filterArray =  
    fl.getDocumentDOM().getTimeline().layers[0].frames[0].elements[0].filters;  
if (filterName == 'glowFilter'){  
    filterArray[0].blurX = 100;  
}  
fl.getDocumentDOM().getTimeline().layers[0].frames[0].elements[0].filters =  
    filterArray;
```

# symbolInstance.firstFrame

## Disponibilité

Flash MX 2004.

## Utilisation

`symbolInstance.firstFrame`

## Description

Propriété ; entier en base zéro indiquant la première image qui doit apparaître dans le scénario du graphisme. Cette propriété ne s'applique qu'aux symboles graphiques et définit le même paramètre que le champ Première image de l'inspecteur des propriétés. Pour les autres types de symboles, cette propriété a la valeur `undefined`.

## Exemple

L'exemple suivant spécifie que l'image 10 doit être la première à apparaître dans le scénario de l'élément spécifié :

```
fl.getDocumentDOM().getTimeline().layers[0].frames[0].elements[0].firstFrame  
    = 10;
```

# symbolInstance.forceSimple

## Disponibilité

Flash MX 2004.

## Utilisation

`symbolInstance.forceSimple`

## Description

Propriété ; valeur booléenne qui active et désactive l'accessibilité aux enfants de l'objet. Cette propriété est équivalente à l'inverse du paramètre Rendre les objets enfant accessibles du panneau Accessibilité. Par exemple, si `forceSimple` est `true`, son effet est similaire à la désactivation de l'option Rendre les objets enfant accessibles. Si `forceSimple` est `false`, son effet est similaire à l'activation de l'option Rendre les objets enfant accessibles.

Cette propriété est uniquement disponible pour les objets MovieClip.

## Exemple

L'exemple suivant vérifie que les enfants de l'objet sont accessibles ; si la valeur renvoyée est `false`, les enfants de l'objet sont accessibles :

```
var areChildrenAccessible = fl.getDocumentDOM().selection[0].forceSimple;
```

L'exemple suivant rend les enfants de l'objet accessibles :

```
fl.getDocumentDOM().selection[0].forceSimple = false;
```

# symbolInstance.loop

## Disponibilité

Flash MX 2004.

## Utilisation

`symbolInstance.loop`

## Description

Propriété ; chaîne qui définit, pour les symboles graphiques, la même propriété que le menu contextuel Boucle de l'inspecteur des propriétés. Pour les autres types de symboles, cette propriété a la valeur `undefined`. Les valeurs gérées sont les suivantes : "loop" (boucle), "play once" (lire une seule fois) et "single frame" (image unique) ; l'animation du graphisme est exécutée en conséquence.

## Exemple

L'exemple suivant donne au premier symbole de la première image du premier calque du scénario la valeur « Single Frame » (afficher une image spécifique du scénario), dans la mesure où ce symbole est un graphisme :

```
fl.getDocumentDOM().getTimeline().layers[0].frames[0].elements[0].loop =  
    'single frame';
```

# symbolInstance.shortcut

## Disponibilité

Flash MX 2004.

## Utilisation

`symbolInstance.shortcut`

## Description

Propriété ; chaîne équivalente à la touche de raccourci associée au symbole. Cette propriété est équivalente au champ Raccourci du panneau Accessibilité. Cette touche est lue par les logiciels de lecture vocale. Cette propriété n'est pas disponible pour les symboles graphiques.

## Exemple

L'exemple suivant enregistre dans la variable `theShortcut` la valeur de la touche de raccourci de l'objet dans le panneau Accessibilité :

```
var theShortcut = fl.getDocumentDOM().selection[0].shortcut;
```

L'exemple suivant définit comme "Ctrl+i" la touche de raccourci de l'objet :

```
fl.getDocumentDOM().selection[0].shortcut = "Ctrl+i";
```

# symbolInstance.silent

## Disponibilité

Flash MX 2004.

## Utilisation

`symbolInstance.silent`

## Description

Propriété ; valeur booléenne qui active et désactive l'accessibilité à l'objet. Cette propriété est équivalente à l'inverse du paramètre Rendre l'objet accessible du panneau Accessibilité.

Par exemple, si `silent` est `true`, son effet est similaire à la désactivation de l'option Rendre l'objet accessible. Si `silent` est `false`, son effet est similaire à l'activation de l'option Rendre l'objet accessible.

Cette propriété n'est pas disponible pour les objets graphiques.

## Exemple

L'exemple suivant vérifie que l'objet est accessible ; si la valeur renvoyée est `false`, l'objet est accessible :

```
var isSilent = fl.getDocumentDOM().selection[0].silent;
```

L'exemple suivant rend l'objet accessible :

```
fl.getDocumentDOM().selection[0].silent = false;
```

# symbolInstance.symbolType

## Disponibilité

Flash MX 2004.

## Utilisation

`symbolInstance.symbolType`

## Description

Propriété ; chaîne qui spécifie le type de symbole. Cette propriété est l'équivalent de la valeur du champ Comportement dans les boîtes de dialogue Créer un nouveau symbole et Convertir en symbole. Les valeurs possibles sont "button" (bouton), "movie clip" (clip) et "graphic".

## Exemple

L'exemple suivant donne le comportement de symbole graphique au premier symbole de la première image du premier calque du scénario du document actif :

```
fl.getDocumentDOM().getTimeline().layers[0].frames[0].elements[0].symbolType = "graphic";
```

# symbolInstance.tabIndex

## Disponibilité

Flash MX 2004.

## Utilisation

`symbolInstance.tabIndex`

## Description

Propriété ; entier équivalent au champ Index de tabulation du panneau Accessibilité. Cette propriété crée l'ordre de tabulation dans lequel les objets seront sélectionnés lorsque l'utilisateur appuiera sur la touche de tabulation. Cette propriété n'est pas disponible pour les symboles graphiques.

## Exemple

L'exemple suivant donne la valeur 3 à la propriété `tabIndex` de l'objet « `mySymbol` », puis affiche cette valeur dans le panneau de sortie :

```
var mySymbol = fl.getDocumentDOM().selection[0];  
mySymbol.tabIndex = 3;  
fl.trace(mySymbol.tabIndex);
```



# Objet SymbolItem

**Héritage** [Objet Item](#) > objet SymbolItem

## Disponibilité

Flash MX 2004.

## Description

L'objet SymbolItem est une sous-classe de l'[Objet Item](#).

## Résumé des méthodes de l'objet SymbolItem

Outre les méthodes de l'[Objet Item](#), les méthodes suivantes sont disponibles avec l'objet SymbolItem :

Méthode	Description
<code>symbolItem.convertToCompiledClip()</code>	Convertit en clip compilé un élément de symbole dans la bibliothèque.
<code>symbolItem.exportSWC()</code>	Exporte le symbole dans un fichier SWC.
<code>symbolItem.exportSWF()</code>	Exporte le symbole dans un fichier SWF.

## Résumé des propriétés de l'objet SymbolItem

Outre les propriétés de l'[Objet Item](#), les propriétés suivantes sont disponibles avec l'objet SymbolItem :

Propriété	Description
<code>symbolItem.scalingGrid</code>	Valeur booléenne qui spécifie si échelle à 9 découpes est activée pour l'élément.
<code>symbolItem.scalingGridRect</code>	Objet Rectangle spécifiant les emplacements des quatre repères pour les 9-découpes.
<code>symbolItem.sourceAutoUpdate</code>	Valeur booléenne qui spécifie si l'élément est actualisé lors de la publication du fichier FLA.
<code>symbolItem.sourceFilePath</code>	Chaîne indiquant le chemin d'accès du fichier source FLA, sous la forme « fichier:/// URI ».
<code>symbolItem.sourceLibraryName</code>	Chaîne qui spécifie le nom de l'élément dans la bibliothèque du fichier source.
<code>symbolItem.symbolType</code>	Chaîne qui spécifie le type de symbole.
<code>symbolItem.timeline</code>	Propriété en lecture seule ; <a href="#">Objet Timeline</a> .

# symbolItem.convertToCompiledClip()

## Disponibilité

Flash MX 2004.

## Utilisation

```
symbolItem.convertToCompiledClip()
```

## Paramètres

Aucun.

## Valeur renvoyée

Aucune.

## Description

Méthode ; convertit en clip compilé un élément de symbole dans la bibliothèque.

## Exemple

L'exemple suivant convertit en clip compilé un élément dans la bibliothèque.

```
fl.getDocumentDOM().library.items[3].convertToCompiledClip();
```

# symbolItem.exportSWC()

## Disponibilité

Flash MX 2004.

## Utilisation

```
symbolItem.exportSWC(outputURI)
```

## Paramètres

*outputURI* Chaîne de type « fichier:/// URI » qui spécifie le fichier SWC dans lequel la méthode doit exporter le symbole. L'adresse *outputURI* doit faire référence à un fichier local. Flash ne crée pas de dossier si le fichier *outputURI* n'existe pas.

## Valeur renvoyée

Aucune.

## Description

Méthode ; exporte le symbole dans un fichier SWC.

## Exemple

L'exemple suivant exporte un élément de la bibliothèque dans un fichier SWC nommé «mySymbol.swc» et situé dans le dossier de tests :

```
fl.getDocumentDOM().library.selectItem("mySymbol");  
var currentSelection = fl.getDocumentDOM().library.getSelectedItems();  
currentSelection[0].exportSWC("file:///Macintosh HD/SWCDirectory/  
mySymbol.swc");
```

## symbolItem.exportSWF()

### Disponibilité

Flash MX 2004.

### Utilisation

`symbolItem.exportSWF(outputURI)`

### Paramètres

*outputURI* Chaîne de type « fichier:/// URI » qui spécifie le fichier SWF dans lequel la méthode doit exporter le symbole. L'adresse *outputURI* doit faire référence à un fichier local. Flash ne crée pas de dossier si *outputURI* n'existe pas.

### Valeur renvoyée

Aucune.

### Description

Méthode ; exporte le symbole dans un fichier SWF.

### Exemple

L'exemple suivant exporte un élément de la bibliothèque dans le fichier « my.swf » et situé dans le dossier « tests » :

```
fl.getDocumentDOM().library.items[0].exportSWF("file:///c:/tests/my.swf");
```

## symbolItem.scalingGrid

### Disponibilité

Flash 8.

### Utilisation

`symbolItem.scalingGrid`

## Description

Propriété ; valeur booléenne spécifiant si échelle à 9 découpes est activée pour l'élément.

## Exemple

L'exemple suivant active l'échelle à 9 découpes pour un élément de la bibliothèque :

```
fl.getDocumentDOM().library.items[0].scalingGrid = true;
```

## Voir aussi

[symbolItem.scalingGridRect](#)

# symbolItem.scalingGridRect

## Disponibilité

Flash 8.

## Utilisation

`symbolItem.scalingGridRect`

## Description

Propriété ; objet Rectangle spécifiant les emplacements des quatre repères pour les 9-découpes. Pour plus d'informations sur le format du rectangle, consultez la section [document.addNewRectangle\(\)](#).

## Exemple

L'exemple suivant spécifie les emplacements des repères pour les 9-découpes :

```
fl.getDocumentDOM().library.items[0].scalingGridRect = {left:338, top:237,  
    right:3859, bottom:713};
```

## Voir aussi

[symbolItem.scalingGrid](#)

# symbolItem.sourceAutoUpdate

## Disponibilité

Flash MX 2004.

## Utilisation

`symbolItem.sourceAutoUpdate`

## Description

Propriété ; valeur booléenne qui spécifie si l'élément est actualisé lors de la publication du fichier FLA. La valeur par défaut est `false`. Cette propriété est utilisée pour les symboles de la bibliothèque partagée.

## Exemple

L'exemple suivant définit la propriété `sourceAutoUpdate` pour un élément de la bibliothèque :

```
fl.getDocumentDOM().library.items[0].sourceAutoUpdate = true;
```

# symbolItem.sourceFilePath

## Disponibilité

Flash MX 2004.

## Utilisation

`symbolItem.sourceFilePath`

## Description

Propriété ; chaîne indiquant le chemin d'accès du fichier source FLA, sous la forme « fichier:/// URI ». Cette adresse doit être un chemin absolu, et non pas un chemin relatif. Cette propriété est utilisée pour les symboles de la bibliothèque partagée.

## Exemple

L'exemple suivant affiche la valeur de la propriété `sourceFilePath` dans le panneau de sortie :

```
fl.trace(fl.getDocumentDOM().library.items[0].sourceFilePath);
```

# symbolItem.sourceLibraryName

## Disponibilité

Flash MX 2004.

## Utilisation

`symbolItem.sourceLibraryName`

## Description

Propriété ; chaîne qui spécifie le nom de l'élément dans la bibliothèque du fichier source. Cette propriété est utilisée pour les symboles de la bibliothèque partagée.

## Exemple

L'exemple suivant affiche la valeur de la propriété `sourceLibraryName` dans le panneau de sortie :

```
fl.trace(fl.getDocumentDOM().library.items[0].sourceLibraryName);
```

# symbolItem.symbolType

## Disponibilité

Flash MX 2004.

## Utilisation

`symbolItem.symbolType`

## Description

Propriété ; chaîne qui spécifie le type de symbole. Les valeurs possibles sont "button" (bouton), "movie clip" (clip) et "graphic".

## Exemple

L'exemple suivant affiche la valeur actuelle de la propriété `symbolType`, puis lui donne la valeur "button" avant de l'afficher à nouveau :

```
alert(fl.getDocumentDOM().library.items[0].symbolType);  
fl.getDocumentDOM().library.items[0].symbolType = "button";  
alert(fl.getDocumentDOM().library.items[0].symbolType);
```

# symbolItem.timeline

## Disponibilité

Flash MX 2004.

## Utilisation

`symbolItem.timeline`

## Description

Propriété en lecture seule ; [Objet Timeline](#).

## Exemple

L'exemple suivant lit et affiche le nombre de calques du clip sélectionné dans la bibliothèque :

```
var tl = fl.getDocumentDOM().library.getSelectedItems()[0].timeline;  
alert(tl.layerCount);
```

# Objet Text

Héritage [Objet Element](#) > objet Text

## Disponibilité

Flash MX 2004.

## Description

L'objet Text représente un élément de texte dans un document. Toutes les propriétés du texte concernent l'ensemble du bloc de texte.

Pour savoir comment définir les propriétés d'un segment de texte à l'intérieur du champ texte, consultez la section « [Résumé des propriétés de l'objet TextRun](#) », à la page 542. Pour modifier les propriétés d'une sélection à l'intérieur d'un champ texte, vous pouvez utiliser `document.setTextAttr()` en spécifiant une plage de texte ou la sélection actuelle.

Pour modifier les propriétés du texte du champ texte sélectionné, utilisez `document.setProperty()`. L'exemple suivant affecte à la variable `textVar` le texte actuellement sélectionné :

```
fl.getDocumentDOM().setProperty("variableName", "textVar");
```

## Résumé des méthodes de l'objet Text

Outre les méthodes de l'[Objet Element](#), les méthodes suivantes sont disponibles avec l'objet Text :

Méthode	Description
<code>text.getTextAttr()</code>	Lit l'attribut spécifié pour le texte identifié par les paramètres facultatifs <code>startIndex</code> et <code>endIndex</code> .
<code>text.getTextString()</code>	Lit la plage de texte spécifiée.
<code>text.setTextAttr()</code>	Définit l'attribut spécifié pour le texte identifié par les paramètres <code>startIndex</code> et <code>endIndex</code> .
<code>text.setTextString()</code>	Modifie la chaîne de texte dans cet objet Text.

## Résumé des propriétés pour l'objet Texte

Outre les propriétés de l'[Objet Element](#), les propriétés suivantes sont disponibles avec l'objet Text :

Propriété	Description
<code>text.accName</code>	Chaîne équivalente au champ Nom du panneau Accessibilité.
<code>text.antiAliasSharpness</code>	Nombre en virgule flottante spécifiant la netteté de l'anti-aliasing du texte.
<code>text.antiAliasThickness</code>	Nombre en virgule flottante spécifiant l'épaisseur de l'anti-aliasing du texte.
<code>text.autoExpand</code>	Valeur booléenne qui contrôle l'expansion de la largeur du cadre de délimitation des champs texte statique, ou la largeur et la hauteur du cadre de délimitation des champs texte dynamique ou de saisie.
<code>text.border</code>	Valeur booléenne qui détermine si Flash affiche ( <code>true</code> ) ou masque ( <code>false</code> ) une bordure autour des champs texte dynamique ou de saisie.
<code>text.description</code>	Chaîne équivalente au champ Description du panneau Accessibilité.
<code>text.embeddedCharacters</code>	Chaîne qui spécifie les caractères à incorporer. Cette propriété est l'équivalent d'une saisie de texte dans la boîte de dialogue Options des caractères.
<code>text.embedRanges</code>	Chaîne composée d'entiers délimités correspondant aux éléments qui peuvent être sélectionnés dans la boîte de dialogue Options des caractères.
<code>text.fontRenderingMode</code>	Chaîne qui spécifie le mode de rendu du texte.
<code>text.length</code>	En lecture seule ; entier représentant le nombre de caractères de l'objet Text.
<code>text.lineType</code>	Chaîne définissant le type de ligne comme "single line" (une seule ligne), "multiline" (multiligne), "multiline no wrap" (multiligne sans retour) ou "password" (mot de passe).
<code>text.maxCharacters</code>	Entier spécifiant le nombre maximal de caractères que l'utilisateur peut indiquer dans cet objet Text.
<code>text.orientation</code>	Chaîne spécifiant l'orientation du champ texte.
<code>text.renderAsHTML</code>	Valeur booléenne qui détermine si Flash affiche le texte en HTML en interprétant les balises HTML incorporées.



Propriété	Description
<code>text.scrollable</code>	Valeur booléenne qui indique si le texte peut défiler ( <code>true</code> ) ou non ( <code>false</code> ).
<code>text.selectable</code>	Valeur booléenne qui indique si le texte peut être sélectionné ( <code>true</code> ) ou non ( <code>false</code> ). Le texte de saisie peut toujours être sélectionné.
<code>text.selectionEnd</code>	Entier en base zéro indiquant le décalage de la fin d'une sous-sélection de texte.
<code>text.selectionStart</code>	Entier en base zéro indiquant le décalage du début d'une sous-sélection de texte.
<code>text.shortcut</code>	Chaîne équivalente au champ Raccourci du panneau Accessibilité.
<code>text.silent</code>	Valeur booléenne indiquant si l'objet est accessible.
<code>text.tabIndex</code>	Entier équivalent au champ Index de tabulation du panneau Accessibilité.
<code>text.textRuns</code>	Propriété en lecture seule ; tableau des objets <code>TextRun</code> .
<code>text.textType</code>	Chaîne qui spécifie le type de champ texte. Les valeurs possibles sont les suivantes : <code>"static"</code> (statique), <code>"dynamic"</code> (dynamique) et <code>"input"</code> (saisie).
<code>text.useDeviceFonts</code>	Une valeur booléenne. Si elle a la valeur <code>true</code> , Flash trace le texte à l'aide des polices de périphérique.
<code>text.variableName</code>	Chaîne présentant le contenu de l'objet <code>Text</code> .

# text.accName

## Disponibilité

Flash MX 2004.

## Utilisation

`text.accName`

## Description

Propriété ; chaîne équivalente au champ Nom du panneau Accessibilité. Les logiciels de lecture vocale identifient l'objet en prononçant ce nom. Cette propriété ne peut pas être utilisée avec du texte dynamique.

## Exemple

L'exemple suivant lit le nom de l'objet :

```
var doc = fl.getDocumentDOM();  
var theName = doc.getTimeline().layers[0].frames[0].elements[0].accName;
```

L'exemple suivant définit le nom de l'objet actuellement sélectionné :

```
fl.getDocumentDOM().selection[0].accName = "Home Button";
```

# text.antiAliasSharpness

## Disponibilité

Flash 8.

## Utilisation

`text.antiAliasSharpness`

## Description

Propriété ; nombre en virgule flottante spécifiant la netteté de l'anti-aliasing du texte. Cette propriété contrôle la netteté selon laquelle le texte est tracé ; une valeur plus élevée correspond à du texte plus net. Une valeur de 0 représente une netteté normale. Cette propriété n'est disponible que si `text.fontRenderingMode` est réglé sur `"customThicknessSharpness"`.

## Exemple

Voir `text.fontRenderingMode`.

## Voir aussi

`text.antiAliasThickness`, `text.fontRenderingMode`

# text.antiAliasThickness

## Disponibilité

Flash 8.

## Utilisation

`text.antiAliasThickness`

## Description

Propriété ; nombre en virgule flottante spécifiant l'épaisseur de l'anti-aliasing du texte. Cette propriété détermine l'épaisseur selon laquelle le texte est traité, une valeur plus élevée représentant un texte plus épais. Une valeur de 0 représente une épaisseur normale. Cette propriété n'est disponible que si `text.fontRenderingMode` est réglé sur "customThicknessSharpness".

## Exemple

Voir `text.fontRenderingMode`.

## Voir aussi

`text.antiAliasSharpness`, `text.fontRenderingMode`

# text.autoExpand

## Disponibilité

Flash MX 2004.

## Utilisation

`text.autoExpand`

## Description

Propriété ; valeur booléenne. Pour les champs texte statique, la valeur `true` provoque une expansion en largeur du cadre de délimitation afin d'afficher tout le texte. Pour les champs texte dynamique ou de saisie, la valeur `true` provoque une expansion en largeur et en hauteur du cadre de délimitation afin d'afficher tout le texte.

## Exemple

L'exemple suivant donne à la propriété `autoExpand` la valeur `true` :

```
fl.getDocumentDOM().selection[0].autoExpand = true;
```

## text.border

### Disponibilité

Flash MX 2004.

### Utilisation

text.border

### Description

Propriété ; valeur booléenne. Si elle a la valeur true, Flash trace une bordure autour du texte.

### Exemple

L'exemple suivant donne à la propriété border la valeur true :

```
fl.getDocumentDOM().getTimeline().layers[0].frames[0].elements[0].border =  
true;
```

## text.description

### Disponibilité

Flash MX 2004.

### Utilisation

text.description

### Description

Propriété ; chaîne équivalente au champ Description du panneau Accessibilité.

Cette description est lue par les logiciels de lecture vocale.

### Exemple

L'exemple suivant lit la description de l'objet :

```
var doc = fl.getDocumentDOM();  
var desc = doc.getTimeline().layers[0].frames[0].elements[0].description;
```

L'exemple suivant définit la description de l'objet :

```
var doc = fl.getDocumentDOM();  
doc.getTimeline().layers[0].frames[0].elements[0].description= "Enter your  
name here";
```

# text.embeddedCharacters

## Disponibilité

Flash MX 2004.

## Utilisation

`text.embeddedCharacters`

## Description

Propriété ; chaîne qui spécifie les caractères à incorporer. Cette propriété est l'équivalent d'une saisie de texte dans la boîte de dialogue Options des caractères.

Elle est disponible uniquement avec les champs texte dynamique ou de saisie, et déclenche un avertissement avec un autre type de champ texte.

## Exemple

L'exemple suivant donne à la propriété `embeddedCharacters` la valeur "abc" :

```
fl.getDocumentDOM().selection[0].embeddedCharacters = "abc";
```

# text.embedRanges

## Disponibilité

Flash MX 2004.

## Utilisation

`text.embedRanges`

## Description

Propriété ; chaîne composée d'entiers délimités correspondant aux éléments qui peuvent être sélectionnés dans la boîte de dialogue Options des caractères. Cette propriété est disponible uniquement avec les champs texte dynamique ou de saisie, et est ignorée avec du texte statique.

REMARQUE

Cette propriété correspond au fichier XML du dossier Configuration/Font Embedding.

## Exemple

L'exemple suivant donne à la propriété `embedRanges` la valeur "1|3|7" :

```
var doc = fl.getDocumentDOM();
doc.getTimeline().layers[0].frames[0].elements[0].embedRanges = "1|3|7";
```

L'exemple suivant réinitialise la propriété :

```
var doc = fl.getDocumentDOM();
doc.getTimeline().layers[0].frames[0].elements[0].embedRanges = "";
```

# text.fontRenderingMode

## Disponibilité

Flash 8.

## Utilisation

`text.fontRenderingMode`

## Description

Propriété ; chaîne qui spécifie le mode de rendu du texte. Cette propriété affecte l'affichage du texte sur la scène et dans Flash Player. Les valeurs gérées sont décrites dans le tableau suivant :

Valeur de la propriété	Rendu du texte
<code>périphérique</code>	Le texte est rendu dans les polices du périphérique.
<code>bitmap</code>	Le texte aliasé est rendu sous la forme d'un bitmap ou comme le ferait une police en mode pixel.
<code>standard</code>	Le texte est rendu par le biais de la méthode d'anti-aliasing standard utilisée par Flash MX 2004. Ce paramètre est particulièrement adapté au texte animé, de très grande taille ou incliné.
<code>advanced</code>	Le texte est restitué par le biais de la technologie de rendu des polices, l'anti-alias avancé mis en oeuvre dans Flash 8, elle produit un meilleur anti-aliasing et améliore la lisibilité, en particulier dans le cas de texte de petite taille.
<code>customThicknessSharpness</code>	Permet de définir des paramètres personnalisés pour la netteté et l'épaisseur du texte en cas d'emploi de la technologie de rendu des polices par l'anti-alias avancé mis en oeuvre dans Flash 8.

## Exemple

L'exemple suivant illustre une utilisation de la valeur `customThicknessSharpness` pour spécifier la netteté et l'épaisseur du texte :

```
fl.getDocumentDOM().setElementProperty("fontRenderingMode",  
    "customThicknessSharpness");  
fl.getDocumentDOM().setElementProperty("antiAliasSharpness", 400);  
fl.getDocumentDOM().setElementProperty("antiAliasThickness", -200);
```

## Voir aussi

[text.antiAliasSharpness](#), [text.antiAliasThickness](#)

# text.getTextAttr()

## Disponibilité

Flash MX 2004.

## Utilisation

```
text.getTextAttr(attrName [, startIndex [, endIndex]])
```

## Paramètres

*attrName* Chaîne spécifiant le nom de la propriété TextAttrs à renvoyer.

REMARQUE

La liste des valeurs possibles de *attrName* figure dans la section [Résumé des propriétés de l'objet TextAttrs](#).

*startIndex* Nombre entier qui spécifie l'index du premier caractère. Ce paramètre est facultatif.

*endIndex* Nombre entier spécifiant la fin de la plage de texte, de *startIndex* compris à *endIndex* non compris. Ce paramètre est facultatif.

## Valeur renvoyée

La valeur de l'attribut spécifié par le paramètre *attrName*.

## Description

Méthode ; lit l'attribut spécifié par le paramètre *attrName* pour le texte identifié par les paramètres facultatifs *startIndex* et *endIndex*. Si cet attribut n'est pas logique pour la plage spécifiée, Flash renvoie la valeur *undefined*. Si vous ne spécifiez pas les paramètres facultatifs *startIndex* et *endIndex*, la méthode utilise la plage de texte entière. Si vous ne précisez que *startIndex*, la plage devient le caractère qui occupe cette position. Si *startIndex* et *endIndex* sont tous deux spécifiés, la plage de caractères commence à *startIndex* et se termine au dernier caractère avant *endIndex*.

## Exemple

L'exemple suivant lit et affiche la taille de caractères du champ texte sélectionné :

```
var TheTextSize = fl.getDocumentDOM().selection[0].getTextAttr("size");
fl.trace(TheTextSize);
```

L'exemple suivant lit la couleur de remplissage du champ texte sélectionné :

```
var TheFill = fl.getDocumentDOM().selection[0].getTextAttr("fillColor");
fl.trace(TheFill);
```

L'exemple suivant lit la taille du troisième caractère du champ texte sélectionné :

```
var Char3 = fl.getDocumentDOM().selection[0].getTextAttr("size", 2);
fl.trace(Char3);
```

L'exemple suivant lit la couleur du champ texte sélectionné entre le troisième et le huitième caractères :

```
fl.getDocumentDOM().selection[0].getTextAttr("fillColor", 2, 8);
```

# text.getTextString()

## Disponibilité

Flash MX 2004.

## Utilisation

```
text.getTextString([startIndex [, endIndex]])
```

## Paramètres

*startIndex* Nombre entier qui spécifie l'index (en base zéro) du premier caractère.

Ce paramètre est facultatif.

*endIndex* Nombre entier spécifiant la fin de la plage de texte, de *startIndex* compris à *endIndex* non compris. Ce paramètre est facultatif.

## Valeur renvoyée

La chaîne de texte de la plage spécifiée.



## Description

Méthode ; lit la plage de texte spécifiée. Si vous omettez les paramètres facultatifs *startIndex* et *endIndex*, la méthode renvoie tout le texte. Si seul le paramètre *startIndex* est transmis, la méthode renvoie la chaîne qui commence à cet index et se termine à la fin du champ.

Si *startIndex* et *endIndex* sont tous deux spécifiés, la méthode renvoie la plage de caractères qui commence à *startIndex* et se termine au dernier caractère avant *endIndex*.

## Exemple

L'exemple suivant lit le texte compris entre le cinquième caractère et la fin de la sélection :

```
var myText = fl.getDocumentDOM().selection[0].getTextString(4);  
fl.trace(myText);
```

L'exemple suivant lit le texte compris entre le quatrième et le neuvième caractères à partir du début du texte sélectionné :

```
var myText = fl.getDocumentDOM().selection[0].getTextString(3, 9);  
fl.trace(myText);
```

# text.length

## Disponibilité

Flash MX 2004.

## Utilisation

text.length

## Description

Propriété en lecture seule ; entier indiquant le nombre de caractères dans l'objet Text.

## Exemple

L'exemple suivant renvoie le nombre de caractères du texte sélectionné :

```
var textLength = fl.getDocumentDOM().selection[0].length;
```

## text.lineType

### Disponibilité

Flash MX 2004.

### Utilisation

`text.lineType`

### Description

Propriété ; chaîne qui spécifie le type de ligne. Les valeurs possibles sont les suivantes : "single line" (une seule ligne), "multiline" (multiligne), "multiline no wrap" (multiligne sans retour) ou "password" (mot de passe).

Cette propriété est uniquement disponible avec les champs texte dynamique ou de saisie, et déclenche un avertissement avec du texte statique. La valeur "password" n'est valide que pour les champs de saisie.

### Exemple

L'exemple suivant donne à la propriété `lineType` la valeur "multiline no wrap" :

```
fl.getDocumentDOM().selection[0].lineType = "multiline no wrap";
```

## text.maxCharacters

### Disponibilité

Flash MX 2004.

### Utilisation

`text.maxCharacters`

### Description

Propriété ; entier spécifiant le nombre maximal de caractères que l'utilisateur peut indiquer dans cet objet Text.

Elle est disponible uniquement avec les champs de saisie, et déclenche un avertissement avec un autre type de champ texte.

### Exemple

L'exemple suivant donne la valeur 30 à la propriété `maxCharacters` :

```
fl.getDocumentDOM().selection[0].maxCharacters = 30;
```

## text.orientation

### Disponibilité

Flash MX 2004.

### Utilisation

`text.orientation`

### Description

Propriété ; chaîne spécifiant l'orientation du champ texte. Les valeurs possibles sont "horizontal", "vertical left to right" (vertical de gauche à droite) et "vertical right to left" (vertical de droite à gauche).

Cette propriété est disponible uniquement avec les champs texte statique, et déclenche un avertissement avec un autre type de champ texte.

### Exemple

L'exemple suivant donne à l'orientation la valeur "vertical right to left" :

```
fl.getDocumentDOM().getTimeline().layers[0].frames[0].elements[0].orientation = "vertical right to left";
```

## text.renderAsHTML

### Disponibilité

Flash MX 2004.

### Utilisation

`text.renderAsHTML`

### Description

Propriété ; valeur booléenne. Si elle a la valeur `true`, Flash affiche le texte en HTML, en interprétant les balises HTML incorporées.

Elle est disponible uniquement avec les champs texte dynamique ou de saisie, et déclenche un avertissement avec un autre type de champ texte.

### Exemple

L'exemple suivant donne à la propriété `renderAsHTML` la valeur `true` :

```
fl.getDocumentDOM().selection[0].renderAsHTML = true;
```

## text.scrollable

### Disponibilité

Flash MX 2004.

### Utilisation

`text.scrollable`

### Description

Propriété ; valeur booléenne. Si elle a la valeur `true`, il est possible de faire défiler le texte.

Cette propriété est uniquement disponible avec du texte saisi ou dynamique ; elle déclenche un avertissement avec du texte statique.

### Exemple

L'exemple suivant donne à la propriété `scrollable` la valeur `false` :

```
fl.getDocumentDOM().selection[0].scrollable = false;
```

## text.selectable

### Disponibilité

Flash MX 2004.

### Utilisation

`text.selectable`

### Description

Propriété ; valeur booléenne. Si elle a la valeur `true`, il est possible de sélectionner le texte.

Le texte de saisie peut toujours être sélectionné. Flash déclenche un avertissement si cette propriété reçoit la valeur `false` avec un champ texte de saisie.

### Exemple

L'exemple suivant donne à la propriété `selectable` la valeur `true` :

```
fl.getDocumentDOM().getTimeline().layers[0].frames[0].elements[0].selectable = true;
```

# text.selectionEnd

## Disponibilité

Flash MX 2004.

## Utilisation

`text.selectionEnd`

## Description

Propriété ; entier en base zéro indiquant la fin d'une sous-sélection de texte. Pour plus d'informations, consultez la section [text.selectionStart](#).

# text.selectionStart

## Disponibilité

Flash MX 2004.

## Utilisation

`text.selectionStart`

## Description

Propriété ; entier en base zéro indiquant le début d'une sous-sélection de texte.

Cette propriété peut être utilisée conjointement avec `text.selectionEnd` pour sélectionner une plage de caractères. Les caractères sont sélectionnés jusqu'à `text.selectionEnd` non compris. Voir [text.selectionEnd](#).

- S'il n'y a pas de sélection, ou si le point d'insertion est actif, `text.selectionEnd` possède la même valeur que `text.selectionStart`.
- Si `text.selectionStart` reçoit une valeur supérieure à celle de `text.selectionEnd`, `text.selectionEnd` prendra la même valeur que `text.selectionStart`, et aucun texte ne sera sélectionné.

## Exemple

L'exemple suivant définit le début de la sous-sélection de texte sur le sixième caractère :

```
fl.getDocumentDOM().getTimeline().layers[0].frames[0].elements[0].selectionStart = 5;
```

L'exemple suivant sélectionne les caractères « Barbara » dans un champ texte contenant la phrase « My name is Barbara », puis met ces caractères en vert gras :

```
fl.getDocumentDOM().selection[0].selectionStart = 11;
fl.getDocumentDOM().selection[0].selectionEnd = 18;
var s = fl.getDocumentDOM().selection[0].selectionStart;
var e = fl.getDocumentDOM().selection[0].selectionEnd;
fl.getDocumentDOM().setElementTextAttr('bold', true, s, e);
fl.getDocumentDOM().setElementTextAttr("fillColor", "#00ff00", s, e);
```

## text.setTextAttr()

### Disponibilité

Flash MX 2004.

### Utilisation

```
text.setTextAttr(attrName, attrValue [, startIndex [, endIndex]])
```

### Paramètres

*attrName* Chaîne spécifiant le nom de la propriété TextAttrs à modifier.

*attrValue* Valeur de la propriété TextAttrs.

REMARQUE

La liste des valeurs possibles de *attrName* et *attrValue* figure à la section « [Résumé des propriétés de l'objet TextAttrs](#) », à la page 532.

*startIndex* Nombre entier qui spécifie l'index (en base zéro) du premier caractère du tableau. Ce paramètre est facultatif.

*endIndex* Nombre entier spécifiant l'index du point d'extrémité de la chaîne de texte sélectionnée, de *startIndex* compris à *endIndex* non compris. Ce paramètre est facultatif.

### Valeur renvoyée

Aucune.

## Description

Méthode ; donne la valeur spécifiée par le paramètre *attrValue* à l'attribut spécifié par le paramètre *attrName* associé au texte identifié par les paramètres facultatifs *startIndex* et *endIndex*. Cette méthode permet de changer des attributs de texte chevauchant des éléments TextRun (voir [Objet TextRun](#)), ou représentant des parties d'éléments TextRun existants. Son utilisation est susceptible de modifier la position et le nombre d'éléments TextRun dans le tableau `text.textRuns` de cet objet (consultez la section [text.textRuns](#)).

Si vous omettez les paramètres facultatifs, la méthode utilise la plage de caractères complète de l'objet Text. Si vous précisez uniquement *startIndex*, la plage devient le caractère qui occupe cette position. Si *startIndex* et *endIndex* sont tous deux spécifiés, la plage de caractères commence à *startIndex* et se termine au dernier caractère avant *endIndex*.

## Exemple

L'exemple suivant met en italique le champ texte sélectionné :

```
fl.getDocumentDOM().selection[0].setTextAttr("italic", true);
```

L'exemple suivant donne une taille de 10 points au troisième caractère :

```
fl.getDocumentDOM().selection[0].setTextAttr("size", 10, 2);
```

L'exemple suivant donne la couleur rouge au texte compris entre le troisième et le huitième caractères :

```
fl.getDocumentDOM().selection[0].setTextAttr("fillColor", 0xff0000, 2, 8);
```

## text.setTextString()

### Disponibilité

Flash MX 2004.

### Utilisation

```
text.setTextString(text [, startIndex [, endIndex]])
```

### Paramètres

*text* Chaîne de caractères composée des caractères à insérer dans l'objet Text.

*startIndex* Nombre entier qui spécifie l'index (en base zéro) du premier caractère de la chaîne dans laquelle le texte sera inséré. Ce paramètre est facultatif.

*endIndex* Nombre entier qui spécifie l'index du dernier caractère de la chaîne sélectionnée. La nouvelle chaîne remplace le texte compris entre *startIndex* et *endIndex* non compris. Ce paramètre est facultatif.

## Valeur renvoyée

Aucune.

## Description

Propriété ; modifie la chaîne de texte de cet objet Text. Si vous omettez les paramètres facultatifs, l'objet Text est remplacé dans son intégralité. Si vous précisez uniquement *startIndex*, la chaîne spécifiée est insérée à la position *startIndex*. Si *startIndex* et *endIndex* sont transmis, la chaîne transmise remplace le segment de texte compris entre *startIndex* et le dernier caractère avant *endIndex*.

## Exemple

L'exemple suivant affecte la chaîne "this is a string" au champ texte sélectionné :

```
fl.getDocumentDOM().selection[0].setTextString("this is a string");
```

L'exemple suivant insère la chaîne "abc" à partir du cinquième caractère du champ texte sélectionné :

```
fl.getDocumentDOM().selection[0].setTextString("01234567890");  
fl.getDocumentDOM().selection[0].setTextString("abc", 4);  
// le champ texte est maintenant "0123abc4567890"
```

L'exemple suivant remplace le texte compris entre le troisième et le huitième caractères du texte sélectionné par la chaîne "abcdefghij". Les caractères compris entre *startIndex* et *endIndex* sont remplacés. La chaîne de caractères qui débute à *endIndex* est décalée après la chaîne insérée.

```
fl.getDocumentDOM().selection[0].setTextString("01234567890");  
fl.getDocumentDOM().selection[0].setTextString("abcdefghij", 2, 8);  
// le champ texte est maintenant "01abcdefghij890"
```

# text.shortcut

## Disponibilité

Flash MX 2004.

## Utilisation

text.shortcut

## Description

Propriété ; chaîne équivalente au champ Raccourci du panneau Accessibilité. Ce raccourci est lu par les logiciels de lecture vocale. Cette propriété ne peut pas être utilisée avec du texte dynamique.



## Exemple

L'exemple suivant lit et affiche la touche de raccourci de l'objet sélectionné :

```
var theShortcut = fl.getDocumentDOM().selection[0].shortcut;  
fl.trace(theShortcut);
```

L'exemple suivant définit la touche de raccourci de l'objet sélectionné :

```
fl.getDocumentDOM().selection[0].shortcut = "Ctrl+i";
```

# text.silent

## Disponibilité

Flash MX 2004.

## Utilisation

`text.silent`

## Description

Propriété ; valeur booléenne indiquant si l'objet est accessible. Cette propriété est équivalente à l'inverse du paramètre Rendre l'objet accessible du panneau Accessibilité. Ainsi, si `silent` est `true`, l'option Rendre l'objet accessible est désactivée. Si cette valeur est `false`, l'option Rendre l'objet accessible est activée.

## Exemple

L'exemple suivant vérifie que l'objet est accessible ; si la valeur renvoyée est `false`, l'objet est accessible :

```
var isSilent =  
    fl.getDocumentDOM().getTimeline().layers[0].frames[0].elements[0].silent;
```

L'exemple suivant rend l'objet accessible :

```
fl.getDocumentDOM().getTimeline().layers[0].frames[0].elements[0].silent =  
    false;
```

# text.tabIndex

## Disponibilité

Flash MX 2004.

## Utilisation

`text.tabIndex`

## Description

Propriété ; entier équivalent au champ Index de tabulation du panneau Accessibilité. Cette valeur permet de déterminer l'ordre dans lequel les objets seront sélectionnés lorsque l'utilisateur appuiera sur la touche de tabulation.

## Exemple

L'exemple suivant lit la valeur de la propriété `tabIndex` de l'objet actuellement sélectionné :

```
var theTabIndex = fl.getDocumentDOM().selection[0].tabIndex;
```

L'exemple suivant définit la propriété `tabIndex` de l'objet actuellement sélectionné :

```
fl.getDocumentDOM().selection[0].tabIndex = 1;
```

# text.textRuns

## Disponibilité

Flash MX 2004.

## Utilisation

`text.textRuns`

## Description

Propriété en lecture seule ; tableau d'objets `TextRun` (consultez la section [Objet TextRun](#)).

## Exemple

L'exemple suivant enregistre dans la variable `myTextRuns` la valeur de la propriété `textRuns` :

```
var myTextRuns = fl.getDocumentDOM().selection[0].textRuns;
```

# text.textType

## Disponibilité

Flash MX 2004.

## Utilisation

`text.textType`

## Description

Propriété ; chaîne qui spécifie le type de champ texte. Les valeurs possibles sont les suivantes : "static" (statique), "dynamic" (dynamique) et "input" (saisie).

### Exemple

L'exemple suivant donne à la propriété `textType` la valeur "input" :

```
fl.getDocumentDOM().selection[0].textType = "input";
```

## text.useDeviceFonts

### Disponibilité

Flash MX 2004.

### Utilisation

```
text.useDeviceFonts
```

### Description

Propriété ; valeur booléenne. Si elle a la valeur `true`, Flash trace le texte à l'aide des polices de périphérique.

### Exemple

L'exemple suivant force Flash à utiliser les polices de périphérique lors du tracé de texte.

```
fl.getDocumentDOM().selection[0].useDeviceFonts = true;
```

## text.variableName

### Disponibilité

Flash MX 2004.

### Utilisation

```
text.variableName
```

### Description

Propriété ; chaîne contenant le nom de la variable associée à l'objet Text. Elle est disponible uniquement avec les champs texte dynamique ou de saisie, et déclenche un avertissement avec un autre type de champ texte.

# Objet TextAttrs

## Disponibilité

Flash MX 2004.

## Description

L'objet TextAttrs contient toutes les propriétés de texte pouvant être appliquées à une sous-sélection. Cet objet est une propriété de l'objet TextRun ([textRun.textAttrs](#)).

## Résumé des propriétés de l'objet TextAttrs

Les propriétés suivantes sont disponibles avec l'objet TextAttrs :

Propriété	Description
<a href="#">textAttrs.aliasText</a>	Valeur booléenne spécifiant que Flash doit tracer le texte à l'aide d'une méthode optimisée pour améliorer la lisibilité du texte en petits caractères.
<a href="#">textAttrs.alignment</a>	Chaîne qui spécifie la justification des paragraphes. Les valeurs possibles sont les suivantes : "left" (aligné à gauche), "center" (centré), "right" (aligné à droite) et "justify" (justifié).
<a href="#">textAttrs.autoKern</a>	Valeur booléenne qui détermine si Flash utilise ( <code>true</code> ) ou ignore ( <code>false</code> ) les informations de crénage (distance entre deux lettres) des polices de caractères.
<a href="#">textAttrs.bold</a>	Une valeur booléenne. Si elle a la valeur <code>true</code> , Flash trace le texte à l'aide de la version « gras » de la police.
<a href="#">textAttrs.characterPosition</a>	Chaîne qui détermine la ligne de base du texte.
<a href="#">textAttrs.characterSpacing</a>	Déconseillé en faveur de <a href="#">textAttrs.letterSpacing</a> . Nombre entier représentant l'espacement entre les caractères.
<a href="#">textAttrs.face</a>	Chaîne qui représente le nom de la police, par exemple "Arial".
<a href="#">textAttrs.fillColor</a>	Chaîne, valeur hexadécimale ou entier représentant la couleur de remplissage.
<a href="#">textAttrs.indent</a>	Entier spécifiant l'indentation des paragraphes.
<a href="#">textAttrs.italic</a>	Une valeur booléenne. Si elle a la valeur <code>true</code> , Flash trace le texte à l'aide de la version « italique » de la police.
<a href="#">textAttrs.leftMargin</a>	Entier spécifiant la marge gauche des paragraphes.

Propriété	Description
<code>textAttrs.letterSpacing</code>	Nombre entier représentant l'espacement entre les caractères.
<code>textAttrs.lineSpacing</code>	Entier spécifiant l'interlignage des paragraphes.
<code>textAttrs.rightMargin</code>	Entier spécifiant la marge droite des paragraphes.
<code>textAttrs.rotation</code>	Une valeur booléenne. Si elle a la valeur <code>true</code> , Flash fait pivoter les caractères du texte à 90 °. La valeur par défaut est <code>false</code> .
<code>textAttrs.size</code>	Entier définissant la taille de la police.
<code>textAttrs.target</code>	Chaîne représentant la propriété <code>target</code> du champ texte.
<code>textAttrs.url</code>	Chaîne représentant la propriété <code>URL</code> du champ texte.

## textAttrs.aliasText

### Disponibilité

Flash MX 2004.

### Utilisation

`textAttrs.aliasText`

### Description

Propriété ; valeur booléenne spécifiant que Flash doit tracer le texte à l'aide d'une méthode optimisée pour améliorer la lisibilité du texte en petits caractères.

### Exemple

L'exemple suivant donne la valeur `true` à la propriété `aliasText` de tout le texte du champ texte sélectionné :

```
fl.getDocumentDOM().setElementTextAttr('aliasText', true);
```

## textAttrs.alignment

### Disponibilité

Flash MX 2004.

### Utilisation

`textAttrs.alignment`

## Description

Propriété ; chaîne qui spécifie la justification des paragraphes. Les valeurs possibles sont les suivantes : "left" (aligné à gauche), "center" (centré), "right" (aligné à droite) et "justify" (justifié).

## Exemple

L'exemple suivant justifie le paragraphe qui contient le texte compris entre le caractère d'index 0 et le caractère d'index 3 non compris : Cette opération peut affecter des caractères situés à l'extérieur de la plage spécifiée s'ils se trouvent dans le même paragraphe.

```
fl.getDocumentDOM().setTextSelection(0, 3);  
fl.getDocumentDOM().setElementTextAttr("alignment", "justify");
```

# textAttrs.autoKern

## Disponibilité

Flash MX 2004.

## Utilisation

textAttrs.autoKern

## Description

Propriété ; valeur booléenne qui détermine si Flash utilise (`true`) ou ignore (`false`) les informations de crénage (distance entre deux lettres) des polices de caractères.

## Exemple

L'exemple suivant sélectionne le texte compris entre le caractère d'index 2 et le caractère d'index 6 non compris, puis donne la valeur `true` à la propriété `autoKern` :

```
fl.getDocumentDOM().setTextSelection(3, 6);  
fl.getDocumentDOM().setElementTextAttr('autoKern', true);
```

# textAttrs.bold

## Disponibilité

Flash MX 2004.

## Utilisation

textAttrs.bold

## Description

Propriété ; valeur booléenne. Si elle a la valeur `true`, Flash trace le texte à l'aide de la version « gras » de la police.

## Exemple

L'exemple suivant sélectionne le premier caractère de l'objet Text sélectionné, puis donne la valeur `true` à la propriété `bold` :

```
fl.getDocumentDOM().setTextSelection(0, 1);  
fl.getDocumentDOM().setElementTextAttr('bold', true);
```

## textAttrs.characterPosition

### Disponibilité

Flash MX 2004.

### Utilisation

`textAttrs.characterPosition`

### Description

Propriété ; chaîne qui détermine la ligne de base du texte. Les valeurs possibles sont les suivantes : "normal", "subscript" (index) et "superscript" (exposant). Cette propriété ne s'applique qu'au texte statique.

### Exemple

L'exemple suivant sélectionne le texte compris entre le caractère d'index 2 et le caractère d'index 6 non compris, puis donne la valeur "subscript" à la propriété

`characterPosition` :

```
fl.getDocumentDOM().setTextSelection(2, 6);  
fl.getDocumentDOM().setElementTextAttr("characterPosition", "subscript");
```

## textAttrs.characterSpacing

### Disponibilité

Flash MX 2004. Délaissée dans Flash 8 au profit de [textAttrs.letterSpacing](#).

### Utilisation

`textAttrs.characterSpacing`

### Description

Propriété ; nombre entier représentant l'espacement entre les caractères. Les valeurs gérées sont comprises entre -60 et 60.

Cette propriété est disponible uniquement avec les champs de texte statique, et déclenche un avertissement avec un autre type de champ texte.

## Exemple

L'exemple suivant donne la valeur 10 à l'espacement des caractères du champ texte sélectionné :

```
fl.getDocumentDOM().setElementTextAttr("characterSpacing", 10);
```

## textAttrs.face

### Disponibilité

Flash MX 2004.

### Utilisation

textAttrs.face

### Description

Propriété ; chaîne qui représente le nom de la police, par exemple "Arial".

## Exemple

L'exemple suivant donne la valeur "Arial" à la police de caractères du texte sélectionné, à partir du caractère d'index 2 et jusqu'au caractère d'index 8 (non inclus) :

```
fl.getDocumentDOM().selection[0].setTextAttr("face", "Arial", 2, 8);
```

## textAttrs.fillColor

### Disponibilité

Flash MX 2004.

### Utilisation

textAttrs.fillColor

### Description

Propriété ; couleur du remplissage, exprimée dans l'un des formats suivants :

- Chaîne au format "#RRVVBB" ou "#RRVVBBA"
- Nombre hexadécimal au format 0xRRVVBB
- Entier représentant l'équivalent décimal d'un nombre hexadécimal

## Exemple

L'exemple suivant donne la valeur « rouge » à la couleur du texte sélectionné, à partir du caractère d'index 2 et jusqu'au caractère d'index 8 (non inclus) :

```
fl.getDocumentDOM().selection[0].setTextAttr("fillColor", 0xff0000, 2, 8);
```



## textAttrs.indent

### Disponibilité

Flash MX 2004.

### Utilisation

`textAttrs.indent`

### Description

Propriété ; entier spécifiant l'indentation des paragraphes. Les valeurs gérées sont comprises entre -720 et 720.

### Exemple

L'exemple suivant donne la valeur 100 à l'indentation du champ texte sélectionné, à partir du caractère d'index 2 et jusqu'au caractère d'index 8 (non inclus). Cette opération peut affecter des caractères situés à l'extérieur de la plage spécifiée s'ils se trouvent dans le même paragraphe.

```
fl.getDocumentDOM().selection[0].setTextAttr("indent", 100, 2, 8);
```

## textAttrs.italic

### Disponibilité

Flash MX 2004.

### Utilisation

`textAttrs.italic`

### Description

Propriété ; valeur booléenne. Si elle a la valeur `true`, Flash trace le texte au moyen de la version « italique » de la police.

### Exemple

L'exemple suivant met en italique le champ texte sélectionné :

```
fl.getDocumentDOM().selection[0].setTextAttr("italic", true);
```

# textAttrs.leftMargin

## Disponibilité

Flash MX 2004.

## Utilisation

`textAttrs.leftMargin`

## Description

Propriété ; entier spécifiant la marge gauche des paragraphes. Les valeurs gérées sont comprises entre 0 et 720.

## Exemple

L'exemple suivant donne la valeur 100 à la propriété `leftMargin` du champ texte sélectionné, à partir du caractère d'index 2 et jusqu'au caractère d'index 8 (non inclus). Cette opération peut affecter des caractères situés à l'extérieur de la plage spécifiée s'ils se trouvent dans le même paragraphe.

```
fl.getDocumentDOM().selection[0].setTextAttr("leftMargin", 100, 2, 8);
```

# textAttrs.letterSpacing

## Disponibilité

Flash 8.

## Utilisation

`textAttrs.letterSpacing`

## Description

Propriété ; nombre entier représentant l'espacement entre les caractères. Les valeurs gérées sont comprises entre -60 et 60.

Cette propriété est disponible uniquement avec les champs de texte statique, elle déclenche un avertissement avec un autre type de champ texte.

## Exemple

Le code suivant sélectionne les caractères de l'index 0 jusqu'à l'index 10 (sans inclure celui-ci) et fixe l'espacement des caractères à 60 :

```
fl.getDocumentDOM().setTextSelection(0, 10);  
fl.getDocumentDOM().setElementTextAttr("letterSpacing", 60);
```

## textAttrs.lineSpacing

### Disponibilité

Flash MX 2004.

### Utilisation

`textAttrs.lineSpacing`

### Description

Propriété ; entier spécifiant l'interlignage du paragraphe. Les valeurs gérées sont comprises entre -360 et 720.

### Exemple

L'exemple suivant donne la valeur 100 à la propriété `lineSpacing` du champ texte sélectionné :

```
fl.getDocumentDOM().selection[0].setTextAttr("lineSpacing", 100);
```

## textAttrs.rightMargin

### Disponibilité

Flash MX 2004.

### Utilisation

`textAttrs.rightMargin`

### Description

Propriété ; entier spécifiant la marge droite des paragraphes. Les valeurs gérées sont comprises entre 0 et 720.

### Exemple

L'exemple suivant donne la valeur 100 à la propriété `rightMargin` du champ texte sélectionné, à partir du caractère d'index 2 et jusqu'au caractère d'index 8 (non inclus). Cette opération peut affecter des caractères situés à l'extérieur de la plage spécifiée s'ils se trouvent dans le même paragraphe.

```
fl.getDocumentDOM().selection[0].setTextAttr("rightMargin", 100, 2, 8);
```

## textAttrs.rotation

### Disponibilité

Flash MX 2004.

### Utilisation

`textAttrs.rotation`

### Description

Propriété ; valeur booléenne. Si elle a la valeur `true`, Flash fait pivoter les caractères du texte à 90 °. La valeur par défaut est `false`. Cette propriété est disponible uniquement avec les champs de texte statique ayant une orientation verticale, et déclenche un avertissement avec un autre type de champ texte.

### Exemple

L'exemple suivant donne la valeur `true` à la rotation du champ texte sélectionné :

```
fl.getDocumentDOM().setElementTextAttr("rotation", true);
```

## textAttrs.size

### Disponibilité

Flash MX 2004.

### Utilisation

`textAttrs.size`

### Description

Propriété ; entier définissant la taille de la police.

### Exemple

L'exemple suivant lit la taille du caractère ayant l'index 2 et affiche le résultat dans le panneau de sortie :

```
fl.outputPanel.trace(fl.getDocumentDOM().selection[0].getTextAttr("size",  
2));
```

# textAttrs.target

## Disponibilité

Flash MX 2004.

## Utilisation

`textAttrs.target`

## Description

Propriété ; chaîne représentant la propriété `target` du champ texte. Cette propriété ne s'applique qu'au texte statique.

## Exemple

L'exemple suivant lit la propriété `target` du champ texte de la première image du calque supérieur de la séquence active, puis l'affiche dans le panneau de sortie :

```
fl.outputPanel.trace(fl.getDocumentDOM().getTimeline().layers[0].frames[0].  
elements[0].getTextAttr("target"));
```

# textAttrs.url

## Disponibilité

Flash MX 2004.

## Utilisation

`textAttrs.url`

## Description

Propriété ; chaîne représentant la propriété `URL` du champ texte. Cette propriété ne s'applique qu'au texte statique.

## Exemple

L'exemple suivant règle l'URL du champ texte sélectionné sur <http://www.adobe.com> :

```
fl.getDocumentDOM().setElementTextAttr("url", "http://www.adobe.com");
```

# Objet TextRun

## Disponibilité

Flash MX 2004.

## Description

L'objet TextRun représente un flux de caractères dont les attributs correspondent à toutes les propriétés de l'Objet TextAttrs. Cet objet est une propriété de l'objet Text ([text.textRuns](#)).

## Résumé des propriétés de l'objet TextRun

Outre les propriétés de l'Objet Text, l'objet TextRun possède les propriétés suivantes :

Propriété	Description
<a href="#">textRun.characters</a>	Une chaîne représentant le texte contenu dans l'objet TextRun.
<a href="#">textRun.textAttrs</a>	L'objet <a href="#">TextAttrs</a> contenant les attributs du segment de texte.

## textRun.characters

### Disponibilité

Flash MX 2004.

### Utilisation

`textRun.characters`

### Description

Propriété ; texte contenu dans l'objet TextRun.

### Exemple

L'exemple suivant affiche dans le panneau de sortie les caractères qui forment la première suite de caractères du champ texte sélectionné :

```
fl.trace(fl.getDocumentDOM().selection[0].textRuns[0].characters);
```

# textRun.textAttrs

## Disponibilité

Flash MX 2004.

## Utilisation

textRun.textAttrs

## Description

Propriété ; l'Objet [TextAttrs](#) contenant les attributs du segment de texte.

## Exemple

L'exemple suivant affiche dans le panneau de sortie les propriétés de la première suite de caractères du champ texte sélectionné.

```
var curTextAttrs = fl.getDocumentDOM().selection[0].textRuns[0].textAttrs;
for (var prop in curTextAttrs) {
    fl.trace(prop + " = " + curTextAttrs[prop]);
}
```

# Objet Timeline

## Disponibilité

Flash MX 2004.

## Description

L'objet Timeline représente le scénario de Flash qui est accessible, pour le document actif, par le biais de `fl.getDocumentDOM().getTimeline()`. Cette méthode renvoie le scénario de la séquence ou du symbole qui est en cours de modification.

Lorsque l'utilisateur travaille sur les séquences, le scénario de chaque séquence possède une valeur d'index qui permet d'y accéder (pour le document actif) via `fl.getDocumentDOM().timelines[i]`. (Dans cet exemple, *i* est l'index de la valeur du scénario.)

Si vous travaillez sur les images en utilisant les méthodes et les propriétés de l'objet Timeline, n'oubliez pas que l'index des images est en base zéro et n'est donc pas égal au numéro de l'image dans la séquence d'images du scénario. Autrement dit, l'image n° 1 possède l'index 0.

## Résumé des méthodes de l'objet Timeline

Les méthodes suivantes sont disponibles pour l'objet Timeline :

Méthode	Description
<code>timeline.addMotionGuide()</code>	Ajoute un calque de guide de mouvement au-dessus du calque actif et rattache ce calque actif au nouveau calque de guide de mouvement.
<code>timeline.addNewLayer()</code>	Ajoute un nouveau calque au document et l'active.
<code>timeline.clearFrames()</code>	Supprime tout le contenu d'une image ou d'une plage d'images dans le calque actif.
<code>timeline.clearKeyframes()</code>	Convertit une image-clé en image standard et supprime son contenu du calque actif.
<code>timeline.convertToBlankKeyframes()</code>	Convertit des images en images-clés vierges dans le calque actif.
<code>timeline.convertToKeyframes()</code>	Convertit une plage d'images-clés en images standard (ou convertit la sélection si aucune image n'est spécifiée) dans le calque actif.
<code>timeline.copyFrames()</code>	Copie une plage d'images du calque actif dans le Presse-papiers.



Méthode	Description
<code>timeline.copyMotion()</code>	A partir d'une interpolation de mouvement ou d'une animation image par image, copie un mouvement sur les images sélectionnées pour pouvoir l'appliquer à d'autres images.
<code>timeline.copyMotionAsAS3()</code>	A partir d'une interpolation de mouvement ou d'une animation image par image, copie via le Presse-papiers un mouvement dans les images sélectionnées sous la forme de code ActionScript 3.0.
<code>timeline.createMotionTween()</code>	Donne la valeur <code>motion</code> à la propriété <code>frame.tweenType</code> pour toutes les images-clés sélectionnées dans le calque actif, et convertit le contenu de chaque image en une occurrence de symbole unique si nécessaire.
<code>timeline.cutFrames()</code>	A partir du scénario, coupe une plage d'images du calque actif et la place dans le Presse-papiers.
<code>timeline.deleteLayer()</code>	Supprime un calque
<code>timeline.expandFolder()</code>	Développe ou réduit le ou les dossiers spécifiés.
<code>timeline.findLayerIndex()</code>	Recherche le nom de calque spécifié dans un tableau d'index.
<code>timeline.getFrameProperty()</code>	Lit la valeur de la propriété spécifiée pour les images sélectionnées.
<code>timeline.getLayerProperty()</code>	Lit la valeur de la propriété spécifiée pour les calques sélectionnés.
<code>timeline.getSelectedFrames()</code>	Recherche dans un tableau les images actuellement sélectionnées.
<code>timeline.getSelectedLayers()</code>	Recherche les valeurs d'index en base zéro des calques sélectionnés.
<code>timeline.insertBlankKeyframe()</code>	Insère une image-clé vierge à l'index spécifié dans le tableau des images ; si l'index n'est pas spécifié, une image-clé vierge est insérée au niveau de la tête de lecture ou de la sélection.
<code>timeline.insertFrames()</code>	Insère le nombre d'images spécifié au numéro d'image indiqué.
<code>timeline.insertKeyframe()</code>	Insère une image-clé au numéro d'image indiqué.
<code>timeline.pasteFrames()</code>	Colle la plage d'images du Presse-papiers dans les images spécifiées.
<code>timeline.pasteMotion()</code>	Copie la plage d'images de mouvements récupérée par <code>timeline.copyMotion()</code> dans le scénario.

Méthode	Description
<code>timeline.removeFrames()</code>	Supprime l'image.
<code>timeline.reorderLayer()</code>	Déplace le premier calque spécifié avant ou après le second calque spécifié.
<code>timeline.reverseFrames()</code>	Inverse une plage d'images.
<code>timeline.selectAllFrames()</code>	Sélectionne toutes les images du scénario actif.
<code>timeline setFrameProperty()</code>	Définit la propriété de l'objet Frame pour les images sélectionnées.
<code>timeline.setLayerProperty()</code>	Donne une valeur spécifiée à la propriété spécifiée pour tous les calques sélectionnés.
<code>timeline.setSelectedFrames()</code>	Sélectionne une plage d'images dans le calque actif, ou applique les images sélectionnées au tableau de sélection passé à cette méthode.
<code>timeline.setSelectedLayers()</code>	Sélectionne le calque spécifié et l'active.
<code>timeline.showLayerMasking()</code>	Affiche les masques de calque pendant la création, en verrouillant le masque et les calques masqués.

## Résumé des propriétés de l'objet Timeline

Les méthodes suivantes sont disponibles pour l'objet Timeline :

Propriété	Description
<code>timeline.currentFrame</code>	Index (en base zéro) de l'image qui se trouve sous la tête de lecture.
<code>timeline.currentLayer</code>	Index (en base zéro) du calque actif.
<code>timeline.frameCount</code>	En lecture seule ; entier indiquant le nombre d'images du calque le plus long du scénario actif.
<code>timeline.layerCount</code>	En lecture seule ; entier indiquant le nombre de calques du scénario spécifié.
<code>timeline.layers</code>	Propriété en lecture seule ; tableau des objets layer.
<code>timeline.name</code>	Chaîne indiquant le nom du scénario actif.

# timeline.addMotionGuide()

## Disponibilité

Flash MX 2004.

## Utilisation

```
timeline.addMotionGuide()
```

## Paramètres

Aucun.

## Valeur renvoyée

Un entier représentant l'index (en base zéro) du nouveau calque de guidage. Si le calque actif n'est pas de type "Normal", Flash renvoie -1.

## Description

Méthode ; ajoute un calque de guide de mouvement au-dessus du calque actif et rattache ce calque actif au nouveau calque de guide de mouvement grâce à la conversion du calque actif en calque de type "Guided".

Cette méthode ne peut être utilisée qu'avec un calque de type "Normal". Elle n'a aucun d'effet sur un calque de type "Folder", "Mask", "Masked", "Guide" ou "Guided".

## Exemple

L'exemple suivant ajoute un calque de guide de mouvement au-dessus du calque actif et convertit ce calque actif en type "Guided" (guidé) :

```
fl.getDocumentDOM().getTimeline().addMotionGuide();
```

# timeline.addNewLayer()

## Disponibilité

Flash MX 2004.

## Utilisation

```
timeline.addNewLayer([name] [, layerType [, bAddAbove]])
```

## Paramètres

*name* Chaîne spécifiant le nom du nouveau calque. Si ce paramètre est omis, un nouveau nom de calque par défaut est affecté au nouveau calque (« Layer n », où *n* correspond au nombre total de calques). Ce paramètre est facultatif.

*layerType* Chaîne spécifiant le type de calque à ajouter. Si ce paramètre est omis, un calque de type « Normal » est créé. Ce paramètre est facultatif. Les valeurs gérées sont les suivantes : "normal", "guide", "guided" (guidé), "mask" (masque), "masked" (masqué) et "folder" (dossier).

*bAddAbove* Valeur booléenne. Si elle vaut `true` (valeur par défaut), Flash ajoute le nouveau calque au-dessus du calque actif ; si elle vaut `false`, Flash ajoute le nouveau calque en dessous du calque actif. Ce paramètre est facultatif.

### Valeur renvoyée

Un entier représentant l'index (en base zéro) du nouveau calque.

### Description

Méthode ; ajoute un nouveau calque au document et l'active.

### Exemple

L'exemple suivant ajoute un nouveau calque au scénario, avec un nom par défaut créé par Flash :

```
fl.getDocumentDOM().getTimeline().addNewLayer();
```

L'exemple suivant ajoute un nouveau calque de dossier au-dessus du calque actif et le nomme "Folder1" :

```
fl.getDocumentDOM().getTimeline().addNewLayer("Folder1", "folder", true);
```

## timeline.clearFrames()

### Disponibilité

Flash MX 2004.

### Utilisation

```
timeline.clearFrames([startFrameIndex [, endFrameIndex]])
```

### Paramètres

*startFrameIndex* Index en base zéro qui définit le début de la plage d'images à vider. Si le paramètre *startFrameIndex* est omis, la méthode utilise la sélection actuelle. Ce paramètre est facultatif.

*endFrameIndex* Index en base zéro qui définit la fin de la plage d'images à vider. Cette plage s'étend jusqu'à *endFrameIndex* non compris. Si seul le paramètre *startFrameIndex* est passé, la valeur de *endFrameIndex* prend par défaut la valeur de *startFrameIndex*. Ce paramètre est facultatif.

### Valeur renvoyée

Aucune.

### Description

Méthode ; supprime tout le contenu d'une image ou d'une plage d'images dans le calque actif.

### Exemple

L'exemple suivant vide les images à partir de l'image 6 et jusqu'à l'image 11 non comprise (n'oubliez pas que les valeurs d'index diffèrent des valeurs de numéro d'image) :

```
fl.getDocumentDOM().getTimeline().clearFrames(5, 10);
```

L'exemple suivant vide l'image 15 :

```
fl.getDocumentDOM().getTimeline().clearFrames(14);
```

## timeline.clearKeyframes()

### Disponibilité

Flash MX 2004.

### Utilisation

```
timeline.clearKeyframes([startFrameIndex [, endFrameIndex]])
```

### Paramètres

*startFrameIndex* Index en base zéro qui définit le début de la plage d'images à vider. Si le paramètre *startFrameIndex* est omis, la méthode utilise la sélection actuelle. Ce paramètre est facultatif.

*endFrameIndex* Index en base zéro qui définit la fin de la plage d'images à vider. Cette plage s'étend jusqu'à *endFrameIndex* non compris. Si seul le paramètre *startFrameIndex* est passé, la valeur de *endFrameIndex* prend par défaut la valeur de *startFrameIndex*. Ce paramètre est facultatif.

### Valeur renvoyée

Aucune.

### Description

Méthode ; convertit une image-clé en image standard et supprime son contenu du calque actif.

## Exemple

L'exemple suivant vide les images-clés à partir de l'image 5 et jusqu'à l'image 10 non comprise (n'oubliez pas que les valeurs d'index diffèrent des valeurs de numéro d'image) :

```
fl.getDocumentDOM().getTimeline().clearKeyframes(4, 9);
```

L'exemple suivant vide l'image-clé 15 et la convertit en image normale :

```
fl.getDocumentDOM().getTimeline().clearKeyframes(14);
```

# timeline.convertToBlankKeyframes()

## Disponibilité

Flash MX 2004.

## Utilisation

```
timeline.convertToBlankKeyframes([startFrameIndex [, endFrameIndex]])
```

## Paramètres

*startFrameIndex* Index en base zéro qui définit la première image à convertir en image-clé. Si le paramètre *startFrameIndex* est omis, la méthode convertit la sélection actuelle. Ce paramètre est facultatif.

*endFrameIndex* Index en base zéro qui définit la dernière image à convertir en image-clé. La plage à convertir s'étend jusqu'à *endFrameIndex* non compris. Si seul le paramètre *startFrameIndex* est passé, la valeur de *endFrameIndex* prend par défaut la valeur de *startFrameIndex*. Ce paramètre est facultatif.

## Valeur renvoyée

Aucune.

## Description

Méthode ; convertit des images en images-clés vides dans le calque actif.

## Exemple

L'exemple suivant convertit en images vides les images-clés à partir de l'image 2 et jusqu'à l'image 10 non comprise (n'oubliez pas que les valeurs d'index diffèrent des valeurs de numéro d'image) :

```
fl.getDocumentDOM().getTimeline().convertToBlankKeyframes(1, 9);
```

L'exemple suivant convertit l'image 5 en image-clé vide :

```
fl.getDocumentDOM().getTimeline().convertToBlankKeyframes(4);
```

# timeline.convertToKeyframes()

## Disponibilité

Flash MX 2004.

## Utilisation

```
timeline.convertToKeyframes([startFrameIndex [, endFrameIndex]])
```

## Paramètres

*startFrameIndex* Index en base zéro qui définit la première image à convertir en image-clé. Si le paramètre *startFrameIndex* est omis, la méthode convertit la sélection actuelle. Ce paramètre est facultatif.

*endFrameIndex* Index en base zéro qui définit la dernière image à convertir en image-clé. La plage à convertir s'étend jusqu'à *endFrameIndex* non compris. Si seul le paramètre *startFrameIndex* est passé, la valeur de *endFrameIndex* prend par défaut la valeur de *startFrameIndex*. Ce paramètre est facultatif.

## Valeur renvoyée

Aucune.

## Description

Méthode ; convertit une plage d'images-clés en images standard (ou convertit la sélection si aucune image n'est spécifiée) dans le calque actif.

## Exemple

L'exemple suivant convertit les images sélectionnées en images-clés :

```
fl.getDocumentDOM().getTimeline().convertToKeyframes();
```

L'exemple suivant convertit en images-clés la plage d'images allant de l'image 2 à l'image 10 non comprise (n'oubliez pas que les valeurs d'index diffèrent des valeurs de numéro d'image) :

```
fl.getDocumentDOM().getTimeline().convertToKeyframes(1, 9);
```

L'exemple suivant convertit l'image 5 en image-clé :

```
fl.getDocumentDOM().getTimeline().convertToKeyframes(4);
```

# timeline.copyFrames()

## Disponibilité

Flash MX 2004.

## Utilisation

```
timeline.copyFrames([startFrameIndex [, endFrameIndex]])
```

## Paramètres

*startFrameIndex* Index en base zéro qui définit le début de la plage d'images à copier.

Si le paramètre *startFrameIndex* est omis, la méthode utilise la sélection actuelle.

Ce paramètre est facultatif.

*endFrameIndex* Index en base zéro qui définit la dernière image à copier. La plage d'images à copier s'étend jusqu'à *endFrameIndex* non compris. Si seul le paramètre *startFrameIndex* est passé, la valeur de *endFrameIndex* prend par défaut la valeur de *startFrameIndex*.

Ce paramètre est facultatif.

## Valeur renvoyée

Aucune.

## Description

Méthode ; copie une plage d'images du calque actif dans le Presse-papiers.

## Exemple

L'exemple suivant copie les images sélectionnées dans le Presse-papiers :

```
fl.getDocumentDOM().getTimeline().copyFrames();
```

L'exemple suivant copie dans le Presse-papiers la plage d'images allant de l'image 2 à l'image 10 non comprise (n'oubliez pas que les valeurs d'index diffèrent des valeurs de numéro d'image) :

```
fl.getDocumentDOM().getTimeline().copyFrames(1, 9);
```

L'exemple suivant copie l'image 5 dans le Presse-papiers :

```
fl.getDocumentDOM().getTimeline().copyFrames(4);
```



# timeline.copyMotion()

## Disponibilité

Flash CS3 Professional.

## Utilisation

`timeline.copyMotion()`

## Paramètres

Aucun.

## Valeur renvoyée

Aucune.

## Description

Méthode ; copie un mouvement dans les images sélectionnées, soit d'une interpolation de mouvement, soit d'une animation image par image. Vous pouvez ensuite utiliser `timeline.pasteMotion()` pour appliquer le mouvement à d'autres images.

Pour copier un mouvement sous forme de texte (code) à coller ensuite dans un script, voir `timeline.copyMotionAsAS3()`.

## Exemple

L'exemple suivant copie le mouvement à partir de l'image ou des images sélectionnées :

```
fl.getDocumentDOM().getTimeline().copyMotion();
```

## Voir aussi

`timeline.copyMotionAsAS3()`, `timeline.pasteMotion()`

# timeline.copyMotionAsAS3()

## Disponibilité

Flash CS3 Professional.

## Utilisation

`timeline.copyMotionAsAS3()`

## Paramètres

Aucun.

## Valeur renvoyée

Aucune.

## Description

Méthode ; à partir d'une interpolation de mouvement ou d'une animation image par image, copie sous forme de code ActionScript 3.0. via le Presse-papiers un mouvement dans les images sélectionnées. Vous pouvez ensuite coller ce code dans un script.

Pour copier un mouvement dans un format applicable à d'autres images, voir [timeline.copyMotion\(\)](#).

## Exemple

L'exemple suivant copie dans le Presse-papiers, sous forme de code ActionScript 3.0, le mouvement à partir de l'image ou des images sélectionnées :

```
fl.getDocumentDOM().getTimeline().copyMotionAsAS3();
```

## Voir aussi

[timeline.copyMotion\(\)](#)

# timeline.createMotionTween()

## Disponibilité

Flash MX 2004.

## Utilisation

```
timeline.createMotionTween([startFrameIndex [, endFrameIndex]])
```

## Paramètres

*startFrameIndex* Index en base zéro qui définit l'image de début de l'interpolation. Si le paramètre *startFrameIndex* est omis, la méthode utilise la sélection actuelle. Ce paramètre est facultatif.

*endFrameIndex* Index en base zéro qui définit la dernière image de l'interpolation. La plage s'étend jusqu'à *endFrameIndex* non compris. Si seul le paramètre *startFrameIndex* est passé, la valeur de *endFrameIndex* prend par défaut la valeur de *startFrameIndex*. Ce paramètre est facultatif.

## Valeur renvoyée

Aucune.

## Description

Méthode ; affecte la valeur `motion` à la propriété `frame.tweenType` pour toutes les images-clés sélectionnées dans le calque actif, et convertit le contenu de chaque image en une instance de symbole unique si nécessaire. Cette propriété est l'équivalent de l'option de menu « Créer une interpolation de mouvement » dans l'interface de Flash.

## Exemple

L'exemple suivant convertit en instance de symbole graphique la forme qui se trouve dans la première image jusqu'à l'image 10 non comprise, et donne la valeur `motion` à la propriété `frame.tweenType` (n'oubliez pas que les valeurs d'index diffèrent des valeurs de numéro d'image) :

```
fl.getDocumentDOM().getTimeline().createMotionTween(0, 9);
```

# timeline.currentFrame

## Disponibilité

Flash MX 2004.

## Utilisation

`timeline.currentFrame`

## Description

Propriété ; index (en base zéro) de l'image qui se trouve sous la tête de lecture.

## Exemple

L'exemple suivant place la tête de lecture face à l'image 10 du scénario actif (n'oubliez pas que les valeurs d'index diffèrent des valeurs de numéro d'image) :

```
fl.getDocumentDOM().getTimeline().currentFrame = 9;
```

L'exemple suivant enregistre dans la variable `curFrame` l'emplacement actuel de la tête de lecture :

```
var curFrame = fl.getDocumentDOM().getTimeline().currentFrame;
```

# timeline.currentLayer

## Disponibilité

Flash MX 2004.

## Utilisation

`timeline.currentLayer`

## Description

Propriété ; index (en base zéro) du calque actif. La valeur 0 spécifie le calque supérieur, la valeur 1 indique le calque placé juste en dessous, etc.

## Exemple

L'exemple suivant active le calque supérieur :

```
fl.getDocumentDOM().getTimeline().currentLayer = 0;
```

L'exemple suivant enregistre dans la variable `curLayer` l'index du calque actif :

```
var curLayer = fl.getDocumentDOM().getTimeline().currentLayer;
```

# timeline.cutFrames()

## Disponibilité

Flash MX 2004.

## Utilisation

`timeline.cutFrames([startFrameIndex [, endFrameIndex]])`

## Paramètres

*startFrameIndex* Index en base zéro qui définit le début de la plage d'images à couper.

Si le paramètre *startFrameIndex* est omis, la méthode utilise la sélection actuelle.

Ce paramètre est facultatif.

*endFrameIndex* Index en base zéro qui définit la dernière image à couper. La plage s'étend jusqu'à *endFrameIndex* non compris. Si seul le paramètre *startFrameIndex* est passé, la valeur de *endFrameIndex* prend par défaut la valeur de *startFrameIndex*. Ce paramètre est facultatif.

## Valeur renvoyée

Aucune.

## Description

Méthode ; à partir du scénario, coupe une plage d'images du calque actif et la place dans le Presse-papiers.

## Exemple

L'exemple suivant coupe les images sélectionnées dans le scénario et les place dans le Presse-papiers :

```
fl.getDocumentDOM().getTimeline().cutFrames();
```

L'exemple suivant coupe et place dans le Presse-papiers la plage d'images, de l'image 2 à l'image 10 non comprise (n'oubliez pas que les valeurs d'index diffèrent des valeurs de numéro d'image) :

```
fl.getDocumentDOM().getTimeline().cutFrames(1, 9);
```

L'exemple suivant coupe l'image 5 dans le scénario et la place dans le Presse-papiers :

```
fl.getDocumentDOM().getTimeline().cutFrames(4);
```

## timeline.deleteLayer()

### Disponibilité

Flash MX 2004.

### Utilisation

```
timeline.deleteLayer([ index ])
```

### Paramètres

*index* Nombre entier en base zéro indiquant le calque à supprimer. S'il n'existe qu'un seul calque dans le scénario, cette méthode n'a aucun effet. Ce paramètre est facultatif.

### Valeur renvoyée

Aucune.

### Description

Méthode ; supprime un calque S'il s'agit d'un calque de type Dossier, tous les calques de ce dossier sont supprimés. Si l'index du calque n'est pas spécifié, Flash supprime les calques actuellement sélectionnés.

## Exemple

L'exemple suivant supprime le second calque à partir du haut :

```
fl.getDocumentDOM().getTimeline().deleteLayer(1);
```

L'exemple suivant supprime les calques sélectionnés :

```
fl.getDocumentDOM().getTimeline().deleteLayer();
```

# timeline.expandFolder()

## Disponibilité

Flash MX 2004.

## Utilisation

```
timeline.expandFolder(bExpand [, bRecurseNestedParents [, index]])
```

## Paramètres

*bExpand* Valeur booléenne. Si elle est *true*, la méthode développe le dossier ; si elle est *false*, la méthode réduit le dossier.

*bRecurseNestedParents* Valeur booléenne. Si elle est *true*, tous les calques du dossier spécifié sont ouverts ou fermés ; suivant la valeur du paramètre *bExpand*. Ce paramètre est facultatif.

*index* Index en base zéro du dossier à développer ou réduire. Utilisez la valeur -1 pour l'appliquer à tous les calques (le paramètre *bRecurseNestedParents* doit également avoir la valeur *true*). Cette propriété est l'équivalent de l'option de menu « Développer/réduire tous les dossiers » dans l'interface de Flash. Ce paramètre est facultatif.

## Valeur renvoyée

Aucune.

## Description

Méthode ; développe ou réduit le ou les dossiers spécifiés. Si vous ne spécifiez pas de calque, cette méthode opère sur le calque actif.

## Exemple

Les exemples suivants utilisent cette structure de dossiers :

```
Folder 1 ***  
--layer 7  
--Folder 2 ****  
----Layer 5
```

L'exemple suivant développe le dossier Folder 1 uniquement :

```
fl.getDocumentDOM().getTimeline().currentLayer = 1;  
fl.getDocumentDOM().getTimeline().expandFolder(true);
```

L'exemple suivant développe le dossier Folder 1 uniquement (en supposant que Folder 2 était réduit lorsque Folder 1 a été réduit précédemment, sinon Folder 2 est également développé) :

```
fl.getDocumentDOM().getTimeline().expandFolder(true, false, 0);
```

L'exemple suivant réduit tous les dossiers du scénario actuel :

```
fl.getDocumentDOM().getTimeline().expandFolder(false, true, -1);
```

# timeline.findLayerIndex()

## Disponibilité

Flash MX 2004.

## Utilisation

```
timeline.findLayerIndex(name)
```

## Paramètres

*name* Chaîne spécifiant le nom du calque à trouver.

## Valeur renvoyée

Un tableau de valeurs d'index pour le calque spécifié. Si le calque spécifié est introuvable, Flash renvoie `undefined`.

## Description

Méthode ; recherche le nom de calque spécifié dans un tableau d'index. Le tableau des calques possède une seule dimension, si bien que les dossiers font partie de l'indexation.

## Exemple

L'exemple suivant affiche la valeur d'index de tous les calques nommés Layer 7 dans le panneau de sortie :

```
var layerIndex = fl.getDocumentDOM().getTimeline().findLayerIndex("Layer  
7");  
fl.trace(layerIndex);
```

L'exemple suivant montre comment passer les valeurs renvoyées par cette méthode à `timeline.setSelectedLayers()` :

```
var layerIndex = fl.getDocumentDOM().getTimeline().findLayerIndex("Layer  
1");  
fl.getDocumentDOM().getTimeline().setSelectedLayers(layerIndex[0], true);
```

# timeline.frameCount

## Disponibilité

Flash MX 2004.

## Utilisation

`timeline.frameCount`

## Description

Propriété en lecture seule ; entier indiquant le nombre d'images du calque le plus long du scénario actif.

## Exemple

L'exemple suivant enregistre dans la variable `countNum` le nombre d'images du calque le plus long du document actif :

```
var countNum = fl.getDocumentDOM().getTimeline().frameCount;
```

# timeline.getFrameProperty()

## Disponibilité

Flash MX 2004.

## Utilisation

`timeline.getFrameProperty(property [, startFrameIndex [, endFrameIndex]])`

## Paramètres

*property* Chaîne spécifiant le nom de la propriété dont la valeur est recherchée. La liste complète des propriétés figure dans la section « [Résumé des propriétés de l'objet Frame](#) », à la page 336.

*startFrameIndex* Index en base zéro qui définit l'image de début dont la valeur est recherchée. Si le paramètre *startFrameIndex* est omis, la méthode utilise la sélection actuelle. Ce paramètre est facultatif.

*endFrameIndex* Index en base zéro qui définit la fin de la plage d'images à sélectionner. Cette plage s'étend jusqu'à *endFrameIndex* non compris. Si seul le paramètre *startFrameIndex* est passé, la valeur de *endFrameIndex* prend par défaut la valeur de *startFrameIndex*. Ce paramètre est facultatif.

## Valeur renvoyée

La valeur de la propriété spécifiée, ou `undefined` si les images sélectionnées n'ont pas toutes la même valeur pour cette propriété.



## Description

Méthode ; lit la valeur de la propriété spécifiée pour les images sélectionnées.

## Exemple

L'exemple suivant lit le nom de la première image du calque supérieur du document actif, et affiche ce nom dans le panneau de sortie :

```
fl.getDocumentDOM().getTimeline().currentLayer = 0;  
fl.getDocumentDOM().getTimeline().setSelectedFrames(0, 0, true);  
var frameName = fl.getDocumentDOM().getTimeline().getFrameProperty("name");  
fl.trace(frameName);
```

# timeline.getLayerProperty()

## Disponibilité

Flash MX 2004.

## Utilisation

`timeline.getLayerProperty(property)`

## Paramètres

*property* Chaîne spécifiant le nom de la propriété dont la valeur est recherchée. La liste des propriétés est indiquée dans la section « [Résumé des propriétés de l'objet Layer](#) », à la page 370.

## Valeur renvoyée

La valeur de la propriété spécifiée. Flash examine les propriétés du calque pour en déterminer le type. Si les calques spécifiés n'ont pas tous la même valeur pour cette propriété, Flash renvoie `undefined`.

## Description

Méthode ; lit la valeur de la propriété spécifiée pour les calques sélectionnés.

## Exemple

L'exemple suivant lit le nom du calque supérieur du document actif, et affiche ce nom dans le panneau de sortie :

```
fl.getDocumentDOM().getTimeline().currentLayer = 0;  
var layerName = fl.getDocumentDOM().getTimeline().getLayerProperty("name");  
fl.trace(layerName);
```

# timeline.getSelectedFrames()

## Disponibilité

Flash MX 2004.

## Paramètres

Aucun.

## Valeur renvoyée

Un tableau contenant  $3n$  entiers, où  $n$  est le nombre de zones sélectionnées. Le premier entier de chaque groupe est l'index du calque, le second entier est l'image de départ de la sélection, et le troisième entier indique l'image de fin de la sélection. L'image de fin n'est pas incluse dans la sélection.

## Description

Méthode ; recherche dans un tableau les images actuellement sélectionnées.

## Exemple

Si le calque supérieur est le calque actif, l'exemple suivant affiche 0, 5, 10, 0, 20, 25 dans le panneau de sortie :

```
var timeline = fl.getDocumentDOM().getTimeline();
timeline.setSelectedFrames(5,10);
timeline.setSelectedFrames(20,25,false);
var theSelectedFrames = timeline.getSelectedFrames();
fl.trace(theSelectedFrames);
```

## Voir aussi

[timeline.setSelectedFrames\(\)](#)

# timeline.getSelectedLayers()

## Disponibilité

Flash MX 2004.

## Paramètres

Aucun.

## Valeur renvoyée

Un tableau des valeurs d'index en base zéro des calques sélectionnés.

## Description

Méthode ; recherche les valeurs d'index en base zéro des calques sélectionnés.

## Exemple

L'exemple suivant affiche 1,0 dans le panneau de sortie.

```
fl.getDocumentDOM().getTimeline().setSelectedLayers(0);
fl.getDocumentDOM().getTimeline().setSelectedLayers(1, false);
var layerArray = fl.getDocumentDOM().getTimeline().getSelectedLayers();
fl.trace(layerArray);
```

## Voir aussi

[timeline.setSelectedLayers\(\)](#)

# timeline.insertBlankKeyframe()

## Disponibilité

Flash MX 2004.

## Utilisation

```
timeline.insertBlankKeyframe([ frameNumIndex])
```

## Paramètres

*frameNumIndex* Index en base zéro qui définit l'image à laquelle l'image-clé doit être insérée. Si le paramètre *frameNumIndex* est omis, la méthode utilise l'image qui se trouve en face de la tête de lecture. Ce paramètre est facultatif.

Si l'image spécifiée ou l'image sélectionnée est une image normale, l'image-clé est insérée au niveau de l'image. Par exemple, pour une plage de 10 images numérotées de 1 à 10, si vous sélectionnez l'image 5, cette méthode transforme l'image 5 en image-clé vide, et la longueur de la plage reste de 10 images. Si l'image 5 est sélectionnée et est une image-clé suivie d'une image normale, cette méthode insère une image-clé vide au niveau de l'image 6. Si l'image 5 est une image-clé et que l'image suivante est déjà une image-clé, aucune image-clé n'est insérée, mais la tête de lecture est placée en face de l'image 6.

## Valeur renvoyée

Aucune.

## Description

Méthode ; insère une image-clé vierge à l'index spécifié dans le tableau des images ; si l'index n'est pas spécifié, une image-clé vierge est insérée au niveau de la tête de lecture ou de la sélection. Voir aussi [timeline.insertKeyframe\(\)](#).

## Exemple

L'exemple suivant insère une image-clé vide au niveau de l'image 20 (n'oubliez pas que les valeurs d'index diffèrent des valeurs de numéro d'image) :

```
fl.getDocumentDOM().getTimeline().insertBlankKeyframe(19);
```

L'exemple suivant insère une image-clé vide au niveau de l'image sélectionnée (ou au niveau de la tête de lecture si aucune image n'est sélectionnée) :

```
fl.getDocumentDOM().getTimeline().insertBlankKeyframe();
```

## timeline.insertFrames()

### Disponibilité

Flash MX 2004.

### Utilisation

```
timeline.insertFrames([numFrames [, bAllLayers [, frameNumIndex]])
```

### Paramètres

*numFrames* Nombre entier qui indique le nombre d'images à insérer. Si ce paramètre est omis, la méthode insère les images au niveau de la sélection actuelle dans le calque actif.

Ce paramètre est facultatif.

*bAllLayers* Valeur booléenne. Si elle a la valeur `true` (valeur par défaut), la méthode insère dans tous les calques le nombre d'images spécifié dans le paramètre *numFrames* ; si elle a la valeur `false`, la méthode insère ces images dans le calque actif. Ce paramètre est facultatif.

*frameNumIndex* Index en base zéro qui définit l'image au niveau de laquelle une nouvelle image doit être insérée. Ce paramètre est facultatif.

### Valeur renvoyée

Aucune.

### Description

Méthode ; insère le nombre d'images spécifié au numéro d'index indiqué.

Si aucun paramètre n'est spécifié, cette méthode opère comme suit :

- Si une ou plusieurs images sont sélectionnées, la méthode insère le nombre d'images sélectionnées à l'emplacement de la première image sélectionnée dans le calque actif. Autrement dit, si les images 6 à 10 sont sélectionnées (soit un total de cinq images), la méthode ajoute cinq images à l'image 6 dans le calque contenant les images sélectionnées.
- Si aucune image n'est sélectionnée, la méthode insère une image dans tous les calques, au niveau de l'image active.

Si des paramètres sont spécifiés, cette méthode opère comme suit :

- Si seul le paramètre *numFrames* est spécifié, la méthode insère le nombre d'images indiqué au niveau de l'image active dans le calque actif.
- Si le paramètre *numFrames* est spécifié et si le paramètre *allLayers* a la valeur *true*, la méthode insère le nombre d'images indiqué au niveau de l'image active dans tous les calques.
- Si les trois paramètres sont spécifiés, la méthode insère le nombre d'images indiqué au niveau de l'index spécifié (*frameIndex*) ; la valeur de *allLayers* détermine si les images sont ajoutées au calque actif seulement ou à tous les calques.

Si l'image spécifiée ou l'image sélectionnée est une image normale, l'image est insérée au niveau de cette image. Par exemple, pour une plage de 10 images numérotées de 1 à 10, si vous sélectionnez l'image 5 (ou si vous passez la valeur 4 pour le paramètre *frameIndex*), cette méthode ajoute une image à l'image 5, et la longueur de la plage devient 11 images. Si l'image 5 est sélectionnée et s'il s'agit d'une image-clé, cette méthode insère une image au niveau de l'image 6, que l'image suivante soit également une image-clé ou non.

## Exemple

L'exemple suivant insère une ou plusieurs images (selon l'étendue de la sélection) au niveau de la sélection actuelle dans le calque actif :

```
fl.getDocumentDOM().getTimeline().insertFrames();
```

L'exemple suivant insère cinq images au niveau de l'image active dans tous les calques :

```
fl.getDocumentDOM().getTimeline().insertFrames(5);
```

### REMARQUE

Si le document comporte plusieurs calques dotés d'images, et si vous avez sélectionné une image dans un calque lors de l'utilisation de la commande précédente, Flash insère les images dans le calque sélectionné uniquement. Si le document comporte plusieurs calques dotés d'images mais qu'aucune image n'est sélectionnée, Flash insère les images dans tous les calques.

L'exemple suivant insère trois images dans le calque actif uniquement :

```
fl.getDocumentDOM().getTimeline().insertFrames(3, false);
```

L'exemple suivant insère quatre images dans tous les calques, en commençant à la première image :

```
fl.getDocumentDOM().getTimeline().insertFrames(4, true, 0);
```

# timeline.insertKeyframe()

## Disponibilité

Flash MX 2004.

## Utilisation

```
timeline.insertKeyframe([ frameNumIndex])
```

## Paramètres

*frameNumIndex* Index en base zéro qui définit l'index de l'image au niveau de laquelle l'image-clé doit être insérée dans le calque actif. Si le paramètre *frameNumIndex* est omis, la méthode utilise l'index de l'image qui se trouve en face de la tête de lecture ou de l'image sélectionnée. Ce paramètre est facultatif.

## Valeur renvoyée

Aucune.

## Description

Méthode ; insère une image-clé au numéro d'image indiqué. Si ce paramètre est omis, la méthode insère une image-clé au niveau de la sélection actuelle ou de la tête de lecture.

Cette méthode donne un résultat identique à celui de `timeline.insertBlankKeyframe()`, si ce n'est que l'image-clé insérée contient le contenu de l'image convertie et n'est donc pas vide.

## Exemple

L'exemple suivant insère une image-clé au niveau de la tête de lecture ou de l'image sélectionnée :

```
fl.getDocumentDOM().getTimeline().insertKeyframe();
```

L'exemple suivant insère une image-clé au niveau de l'image 10 du second calque (n'oubliez pas que les valeurs d'index diffèrent des valeurs de numéro d'image) :

```
fl.getDocumentDOM().getTimeline().currentLayer = 1;  
fl.getDocumentDOM().getTimeline().insertKeyframe(9);
```

## timeline.layerCount

### Disponibilité

Flash MX 2004.

### Utilisation

`timeline.layerCount`

### Description

Propriété en lecture seule ; entier indiquant le nombre de calques du scénario spécifié.

### Exemple

L'exemple suivant enregistre dans la variable `NumLayer` le nombre de calques de la séquence active :

```
var NumLayer = fl.getDocumentDOM().getTimeline().layerCount;
```

## timeline.layers

### Disponibilité

Flash MX 2004.

### Utilisation

`timeline.layers`

### Description

Propriété en lecture seule ; tableau des objets `layer`.

### Exemple

L'exemple suivant enregistre dans la variable `currentLayers` le tableau des objets `layer` du document actif :

```
var currentLayers = fl.getDocumentDOM().getTimeline().layers;
```

# timeline.name

## Disponibilité

Flash MX 2004.

## Utilisation

`timeline.name`

## Description

Propriété ; chaîne qui spécifie le nom du scénario actif. Ce nom est celui de la séquence, de l'écran (diapositive ou formulaire) ou du symbole qui est en cours de modification.

## Exemple

L'exemple suivant lit le nom de la séquence active :

```
var sceneName = fl.getDocumentDOM().timelines[0].name;
```

L'exemple suivant donne la valeur `FirstScene` au nom de la première séquence :

```
fl.getDocumentDOM().timelines[0].name = "FirstScene";
```

# timeline.pasteFrames()

## Disponibilité

Flash MX 2004.

## Utilisation

`timeline.pasteFrames([startFrameIndex [, endFrameIndex]])`

## Paramètres

*startFrameIndex* Index en base zéro qui définit le début de la plage d'images à coller. Si le paramètre *startFrameIndex* est omis, la méthode utilise la sélection actuelle. Ce paramètre est facultatif.

*endFrameIndex* Index en base zéro qui définit la dernière image au niveau de laquelle les images doivent être collées. Cette plage s'étend jusqu'à *endFrameIndex* non compris. Si seul le paramètre *startFrameIndex* est passé, la valeur de *endFrameIndex* prend par défaut la valeur de *startFrameIndex*. Ce paramètre est facultatif.

## Valeur renvoyée

Aucune.

## Description

Méthode ; colle la plage d'images du Presse-papiers dans les images spécifiées.



## Exemple

L'exemple suivant colle la plage d'images du Presse-papiers dans l'image sélectionnée ou, à défaut, à l'emplacement de la tête de lecture :

```
fl.getDocumentDOM().getTimeline().pasteFrames();
```

L'exemple suivant colle les images dans le Presse-papiers à partir de l'image 2 et jusqu'à l'image 10 non comprise (n'oubliez pas que les valeurs d'index diffèrent des valeurs de numéro d'image) :

```
fl.getDocumentDOM().getTimeline().pasteFrames(1, 9);
```

L'exemple suivant colle les images dans le Presse-papiers à partir de l'image 5 :

```
fl.getDocumentDOM().getTimeline().pasteFrames(4);
```

## timeline.pasteMotion()

### Disponibilité

Flash CS3 Professional.

### Utilisation

```
timeline.pasteMotion()
```

### Paramètres

Aucun.

### Valeur renvoyée

Aucune.

### Description

Méthode ; copie dans le scénario la plage d'images de mouvement récupérée par [timeline.copyMotion\(\)](#). Au besoin, les images existantes sont déplacées (repositionnées à droite) pour faire de la place aux images qui sont collées.

## Exemple

L'exemple suivant colle le mouvement du Presse-papiers dans l'image sélectionnée ou à l'emplacement de la tête de lecture en déplaçant cette image à droite des images collées :

```
fl.getDocumentDOM().getTimeline().pasteMotion();
```

### Voir aussi

[timeline.copyMotion\(\)](#)

# timeline.removeFrames()

## Disponibilité

Flash MX 2004.

## Utilisation

```
timeline.removeFrames([startFrameIndex [, endFrameIndex]])
```

## Paramètres

*startFrameIndex* Index en base zéro qui définit la première image à partir de laquelle les images doivent être supprimées. Si le paramètre *startFrameIndex* est omis, la méthode utilise la sélection actuelle ; s'il n'existe pas de sélection, toutes les images (de tous les calques) qui se trouvent au niveau de la tête de lecture sont supprimées. Ce paramètre est facultatif.

*endFrameIndex* Index en base zéro qui définit la dernière image au niveau de laquelle les images doivent être supprimées ; la plage s'étend jusqu'à *endFrameIndex* non compris. Si seul le paramètre *startFrameIndex* est passé, la valeur de *endFrameIndex* prend par défaut la valeur de *startFrameIndex*. Ce paramètre est facultatif.

## Valeur renvoyée

Aucune.

## Description

Méthode ; supprime l'image.

## Exemple

L'exemple suivant supprime les images du calque supérieur de la séquence active à partir de l'image 5 et jusqu'à l'image 10 non comprise (n'oubliez pas que les valeurs d'index diffèrent des valeurs de numéro d'image) :

```
fl.getDocumentDOM().getTimeline().currentLayer = 0;  
fl.getDocumentDOM().getTimeline().removeFrames(4, 9);
```

L'exemple suivant supprime l'image 8 du calque supérieur de la séquence active :

```
fl.getDocumentDOM().getTimeline().currentLayer = 0;  
fl.getDocumentDOM().getTimeline().removeFrames(7);
```

# timeline.reorderLayer()

## Disponibilité

Flash MX 2004.

## Utilisation

```
timeline.reorderLayer(layerToMove, layerToPutItBy [, bAddBefore])
```

## Paramètres

*layerToMove* Index en base zéro indiquant le calque à déplacer.

*layerToPutItBy* Index en base zéro indiquant le calque à côté duquel le calque doit être placé. Par exemple, si vous spécifiez 1 pour *layerToMove* et 0 pour *layerToPutItBy*, le second calque est placé à côté du premier.

*bAddBefore* Indique si le calque doit être placé avant ou après *layerToPutItBy*. Si sa valeur est *false*, le calque est placé après *layerToPutItBy*. La valeur par défaut est *true*. Ce paramètre est facultatif.

## Valeur renvoyée

Aucune.

## Description

Méthode ; déplace le premier calque spécifié avant ou après le second calque spécifié.

## Exemple

L'exemple suivant place le calque ayant l'index 2 au niveau supérieur (au-dessus du calque ayant l'index 0) :

```
fl.getDocumentDOM().getTimeline().reorderLayer(2, 0);
```

L'exemple suivant place le calque ayant l'index 3 après le calque doté de l'index 5 :

```
fl.getDocumentDOM().getTimeline().reorderLayer(3, 5, false);
```

# timeline.reverseFrames()

## Disponibilité

Flash MX 2004.

## Utilisation

```
timeline.reverseFrames([startFrameIndex [, endFrameIndex]])
```

## Paramètres

*startFrameIndex* Index en base zéro qui définit la première image à partir de laquelle les images doivent être inversées. Si le paramètre *startFrameIndex* est omis, la méthode utilise la sélection actuelle. Ce paramètre est facultatif.

*endFrameIndex* Index en base zéro qui définit la dernière image au niveau de laquelle les images doivent être inversées ; la plage s'étend jusqu'à *endFrameIndex* non compris. Si seul le paramètre *startFrameIndex* est passé, la valeur de *endFrameIndex* prend par défaut la valeur de *startFrameIndex*. Ce paramètre est facultatif.

## Valeur renvoyée

Aucune.

## Description

Méthode ; inverse une plage d'images.

## Exemple

L'exemple suivant inverse les positions des calques actuellement sélectionnés :

```
fl.getDocumentDOM().getTimeline().reverseFrames();
```

L'exemple suivant inverse les images à partir de l'image 10 et jusqu'à l'image 15 non comprise (n'oubliez pas que les valeurs d'index diffèrent des valeurs de numéro d'image) :

```
fl.getDocumentDOM().getTimeline().reverseFrames(9, 14);
```

# timeline.selectAllFrames()

## Disponibilité

Flash MX 2004.

## Utilisation

```
timeline.selectAllFrames()
```

## Paramètres

Aucun.

## Valeur renvoyée

Aucune.

## Description

Méthode ; sélectionne toutes les images du scénario actif.

## Exemple

L'exemple suivant sélectionne toutes les images du scénario actif.

```
fl.getDocumentDOM().getTimeline().selectAllFrames();
```

# timeline setFrameProperty()

## Disponibilité

Flash MX 2004.

## Utilisation

```
timeline.setFrameProperty(property, value [, startFrameIndex [,  
                                endFrameIndex]])
```

## Paramètres

*property* Chaîne spécifiant le nom de la propriété à modifier. La liste complète des propriétés et de leurs valeurs figure dans la section « [Résumé des propriétés de l'objet Frame](#) », à la page 336.

REMARQUE

Cette méthode ne permet pas de définir les propriétés en lecture seule, telles que `frame.duration` et `frame.elements`.

*value* Valeur à affecter à la propriété. Pour déterminer les valeurs et les types appropriés, consultez la section « [Résumé des propriétés de l'objet Frame](#) », à la page 336.

*startFrameIndex* Index en base zéro qui définit la première image à modifier. Si le paramètre *startFrameIndex* est omis, la méthode utilise la sélection actuelle. Ce paramètre est facultatif.

*endFrameIndex* Index en base zéro qui définit la dernière image à modifier. La plage s'étend jusqu'à *endFrameIndex* non compris. Si seul le paramètre *startFrameIndex* est passé et que *endFrameIndex* est omis, la valeur de *endFrameIndex* prend par défaut la valeur de *startFrameIndex*. Ce paramètre est facultatif.

## Valeur renvoyée

Aucune.

## Description

Méthode ; définit la propriété de l'objet `Frame` pour les images sélectionnées.

## Exemple

L'exemple suivant affecte la commande `ActionScript stop()` à la première image du calque supérieur du document actif :

```
fl.getDocumentDOM().getTimeline().currentLayer = 0;  
fl.getDocumentDOM().getTimeline().setSelectedFrames(0,0,true);  
fl.getDocumentDOM().getTimeline().setFrameProperty("actionScript",  
    "stop();");
```

L'exemple suivant définit une interpolation de mouvement des images comprises entre l'image 2 et l'image 5 non comprise (n'oubliez pas que les valeurs d'index diffèrent des valeurs de numéro d'image) :

```
var doc = fl.getDocumentDOM();  
doc.getTimeline().setFrameProperty("tweenType","motion",1,4);
```

# timeline.setLayerProperty()

## Disponibilité

Flash MX 2004.

## Utilisation

`timeline.setLayerProperty(property, value [, layersToChange])`

## Paramètres

*property* Chaîne définissant la propriété à modifier. La liste des propriétés est indiquée dans la section « [Objet Layer](#) », à la page 370.

*value* Valeur à affecter à la propriété. Faites appel au type de valeur utilisé pour définir la propriété pour l'objet `Layer`.

*layersToChange* Chaîne identifiant les calques à modifier. Les valeurs acceptables sont "selected" (sélectionné), "all" (tous) et "others" (autres) Si vous omettez ce paramètre, la valeur par défaut est "selected". Ce paramètre est facultatif.

## Valeur renvoyée

Aucune.

## Description

Méthode ; donne une valeur spécifiée à la propriété spécifiée pour tous les calques sélectionnés.

## Exemple

L'exemple suivant rend invisibles le ou les calques sélectionnés :

```
fl.getDocumentDOM().getTimeline().setLayerProperty("visible", false);
```

L'exemple suivant définit comme "selLayer" le nom du ou des calques sélectionnés :

```
fl.getDocumentDOM().getTimeline().setLayerProperty("name", "selLayer");
```

# timeline.setSelectedFrames()

## Disponibilité

Flash MX 2004.

## Utilisation

```
timeline.setSelectedFrames(startFrameIndex, endFrameIndex [,  
                           bReplaceCurrentSelection])
```

```
timeline.setSelectedFrames(selectionList [, bReplaceCurrentSelection])
```

## Paramètres

*startFrameIndex* Index en base zéro qui définit la première image à modifier.

*endFrameIndex* Index en base zéro qui définit la fin de sélection ; *endFrameIndex* est l'image qui suit la dernière image de la plage à sélectionner.

*bReplaceCurrentSelection* Valeur booléenne ; si elle vaut `true`, les images actuellement sélectionnées sont désélectionnées, puis les images spécifiées sont sélectionnées. La valeur par défaut est `true`.

*selectionList* Tableau de trois entiers renvoyés par `timeline.getSelectedFrames()`.

## Valeur renvoyée

Aucune.

## Description

Méthode ; sélectionne une plage d'images dans le calque actif, ou applique les images sélectionnées au tableau de sélection passé à cette méthode.

## Exemple

Les exemples suivants montrent deux sélections possibles, d'abord dans le calque supérieur la plage d'images allant de l'image 1 à l'image 10 (non comprise) ; puis la plage de l'image 12 à l'image 15 (non comprise) du même calque pour l'ajouter à la sélection en cours (n'oubliez pas que les valeurs d'index diffèrent des valeurs de numéro d'image) :

```
fl.getDocumentDOM().getTimeline().setSelectedFrames(0, 9);
fl.getDocumentDOM().getTimeline().setSelectedFrames(11, 14, false);

fl.getDocumentDOM().getTimeline().setSelectedFrames([0, 0, 9]);
fl.getDocumentDOM().getTimeline().setSelectedFrames([0, 11, 14], false);
```

L'exemple suivant enregistre dans la variable `savedSelectionList` le tableau des images sélectionnées, puis utilise ce tableau dans le code pour sélectionner à nouveau ces images après modification de la sélection par le biais d'une commande ou d'une interaction de l'utilisateur :

```
var savedSelectionList =
    fl.getDocumentDOM().getTimeline().getSelectedFrames();
// Exécuter une action qui change la sélection.
fl.getDocumentDOM().getTimeline().setSelectedFrames(savedSelectionList);
```

## Voir aussi

[timeline.getSelectedFrames\(\)](#)

# timeline.setSelectedLayers()

## Disponibilité

Flash MX 2004.

## Utilisation

`timeline.setSelectedLayers(index [, bReplaceCurrentSelection])`

## Paramètres

*index* Nombre entier en base zéro indiquant le calque à sélectionner.

*bReplaceCurrentSelection* Valeur booléenne. Si elle est `true`, la méthode remplace la sélection en cours ; si elle est `false`, la méthode étend la sélection. La valeur par défaut est `true`. Ce paramètre est facultatif.

## Valeur renvoyée

Aucune.

## Description

Méthode ; sélectionne le calque spécifié et l'active. Lorsqu'un calque est sélectionné, toutes ses images sont sélectionnées.



## Exemple

L'exemple suivant sélectionne le calque supérieur :

```
fl.getDocumentDOM().getTimeline().setSelectedLayers(0);
```

L'exemple suivant ajoute le calque suivant à la sélection :

```
fl.getDocumentDOM().getTimeline().setSelectedLayers(1, false);
```

## Voir aussi

[timeline.getSelectedLayers\(\)](#)

# timeline.showLayerMasking()

## Disponibilité

Flash MX 2004.

## Utilisation

```
timeline.showLayerMasking([layer])
```

## Paramètres

*layer* Index en base zéro d'un calque de masque ou d'un calque masqué à afficher durant la création. Ce paramètre est facultatif.

## Valeur renvoyée

Aucune.

## Description

Méthode ; affiche les masques de calque pendant la création, en verrouillant le masque et les calques masqués. Si aucun calque n'est spécifié, la méthode s'applique au calque actif. Si cette méthode est utilisée sur un calque qui n'est pas de type Masque ou Masqué, Flash affiche une erreur dans le panneau de sortie.

## Exemple

L'exemple suivant spécifie que le masquage du premier calque doit apparaître pendant la création.

```
fl.getDocumentDOM().getTimeline().showLayerMasking(0);
```

# Objet ToolObj

## Disponibilité

Flash MX 2004.

## Description

Un objet toolObj représente un outil dans le panneau Outils. Pour accéder à un objet ToolObj, utilisez les propriétés de l'Objet Tools : soit le tableau `tools.toolObjs`, soit `tools.activeTool`.

## Résumé des méthodes de l'objet ToolObj

Les méthodes suivantes sont disponibles avec l'objet ToolObj :

REMARQUE

Ces méthodes ne sont utilisées que lors de la création d'outils extensibles.

Méthode	Description
<code>toolObj.enablePIControl()</code>	Active ou désactive le contrôle spécifié dans un inspecteur des propriétés. Cette méthode n'est utilisée que lors de la création d'outils extensibles.
<code>toolObj.setIcon()</code>	Identifie un fichier PNG à utiliser comme icône d'outil dans le panneau Outils de Flash.
<code>toolObj.setMenuString()</code>	Définit la chaîne qui doit apparaître dans le menu contextuel pour identifier l'outil.
<code>toolObj.setOptionsFile()</code>	Associe un fichier XML à l'outil.
<code>toolObj.setPI()</code>	Définit un inspecteur des propriétés spécifique à utiliser lorsque l'outil est activé.
<code>toolObj.setToolName()</code>	Affecte un nom à l'outil pour la configuration du panneau Outils.
<code>toolObj.setToolTip()</code>	Définit l'info-bulle qui doit apparaître lorsque la souris est amenée sur l'icône de l'outil.

Méthode	Description
<code>toolObj.showPControl()</code>	Affiche ou masque un contrôle dans l'inspecteur des propriétés.
<code>toolObj.showTransformHandles()</code>	Cette méthode est appelée par la méthode <code>configureTool()</code> du fichier JavaScript d'un outil extensible pour indiquer que les poignées de transformation libre doivent apparaître lorsque l'outil est actif.

## Résumé des propriétés de l'objet ToolObj

Les propriétés suivantes sont disponibles pour l'objet ToolObj :

Propriété	Description
<code>toolObj.depth</code>	Entier qui définit la profondeur de l'outil dans le menu contextuel du panneau Outils.
<code>toolObj.iconID</code>	Entier qui définit l'ID de ressource de l'outil.
<code>toolObj.position</code>	En lecture seule ; entier indiquant la position de l'outil dans le panneau Outils.

## toolObj.depth

### Disponibilité

Flash MX 2004.

### Utilisation

`toolObj.depth`

### Description

Propriété en lecture seule ; entier qui définit la profondeur de l'outil dans le menu contextuel du panneau Outils. Cette méthode n'est utilisée que lors de la création d'outils extensibles.

### Exemple

L'exemple suivant spécifie que l'outil possède une profondeur de 1, ce qui signifie qu'il se trouve un niveau en dessous d'un outil dans le panneau Outils :

```
fl.tools.activeTool.depth = 1;
```

# toolObj.enablePIControl()

## Disponibilité

Flash MX 2004.

## Utilisation

`toolObj.enablePIControl(control, bEnable)`

## Paramètres

*control* Chaîne spécifiant le nom du contrôle à activer ou désactiver. Les valeurs autorisées dépendent de l'inspecteur des propriétés appelé par cet outil (voir [toolObj.setPI\(\)](#)).

L'inspecteur des propriétés d'une forme possède les contrôles suivants :

stroke	fill
--------	------

L'inspecteur des propriétés d'un texte possède les contrôles suivants :

type	font	pointsize
color	bold	italic
direction	alignLeft	alignCenter
alignRight	alignJustify	spacing
position	autoKern	small
rotation	format	lineType
selectable	html	border
deviceFonts	varEdit	options
link	maxChars	target

L'inspecteur des propriétés d'un film possède les contrôles suivants :

size	publish	background
framerate	player	profile

*bEnable* Valeur booléenne indiquant si le contrôle doit être activé (`true`) ou non (`false`).

## Valeur renvoyée

Aucune.

## Description

Méthode ; active ou désactive le contrôle spécifié dans un inspecteur des propriétés.  
Cette méthode n'est utilisée que lors de la création d'outils extensibles.

## Exemple

Si la commande suivante est insérée dans le fichier JavaScript d'un outil extensible, Flash n'affiche pas les options de trait dans l'inspecteur des propriétés de cet outil :

```
theTool.enablePIControl("stroke", false);
```

# toolObj.iconID

## Disponibilité

Flash MX 2004.

## Utilisation

`toolObj.iconID`

## Description

Propriété en lecture seule ; entier de valeur-1. Cette propriété s'utilise uniquement lors de la création d'outils extensibles. Une valeur `iconID` de 1 signifie que Flash ne tentera pas de trouver une icône pour l'outil. Au contraire, le script de l'outil doit définir l'icône à afficher dans le panneau Outils (voir [toolObj.setIcon\(\)](#)).

## Exemple

L'exemple suivant attribue une valeur de 1 (l'ID d'icône de l'outil actuel) à la variable

`toolIconID` :

```
var toolIconID = fl.tools.activeTool.iconID
```

# toolObj.position

## Disponibilité

Flash MX 2004.

## Utilisation

`toolObj.position`

## Description

Propriété en lecture seule ; entier indiquant la position de l'outil dans le panneau Outils. Cette propriété n'est utilisée que lors de la création d'outils extensibles.

## Exemple

Si les commandes suivantes sont insérées dans la méthode `mouseDown()` du fichier JavaScript d'un outil extensible, la position de cet outil dans le panneau Outils sera affichée sur un entier dans le panneau de sortie :

```
myToolPos = fl.tools.activeTool.position;  
fl.trace(myToolPos);
```

## toolObj.setIcon()

### Disponibilité

Flash MX 2004.

### Utilisation

```
toolObj.setIcon(file)
```

### Paramètres

*file* Chaîne spécifiant le nom du fichier PNG à utiliser comme icône. Le fichier PNG doit être placé dans le même dossier que le fichier JSFL.

### Valeur renvoyée

Aucune.

### Description

Méthode ; identifie un fichier PNG à utiliser comme icône d'outil dans le panneau Outils. Cette méthode n'est utilisée que lors de la création d'outils extensibles.

## Exemple

L'exemple suivant indique que l'image enregistrée dans le fichier « PolyStar.png » doit être utilisée comme icône de l'outil PolyStar. Ce code est extrait du fichier d'exemple « PolyStar.jsfl » (consultez la section « [Exemple d'outil PolyStar](#) », à la page 20) :

```
theTool = fl.tools.activeTool;  
theTool.setIcon("PolyStar.png");
```

## toolObj.setMenuString()

### Disponibilité

Flash MX 2004.

### Utilisation

```
toolObj.setMenuString(menuStr)
```

### Paramètres

*menuStr* Chaîne spécifiant le nom qui doit apparaître dans le menu contextuel pour identifier l'outil.

### Valeur renvoyée

Aucune.

### Description

Méthode ; définit la chaîne qui doit apparaître dans le menu contextuel pour identifier l'outil. Cette méthode n'est utilisée que lors de la création d'outils extensibles.

### Exemple

L'exemple suivant indique que l'outil `theTool` doit afficher le nom « PolyStar Tool » dans son menu contextuel. Ce code est extrait du fichier d'exemple « PolyStar.jsfl » (consultez la section « [Exemple d'outil PolyStar](#) », à la page 20) :

```
theTool = fl.tools.activeTool;  
theTool.setMenuString("PolyStar Tool");
```

## toolObj.setOptionsFile()

### Disponibilité

Flash MX 2004.

### Utilisation

```
toolObj.setOptionsFile(xmlFile)
```

### Paramètres

*xmlFile* Chaîne spécifiant le nom du fichier XML qui contient la description des options de l'outil. Le fichier XML doit être placé dans le même dossier que le fichier JSFL.

### Valeur renvoyée

Aucune.

## Description

Méthode ; associe un fichier XML à l'outil. Le fichier spécifie les options qui s'afficheront dans un panneau modal appelé par un bouton Options dans l'inspecteur des propriétés. Vous utiliserez en règle générale cette méthode dans la fonction `configureTool()` intégrée à votre fichier JSFL. Voir [configureTool\(\)](#).

Le fichier « PolyStar.xml » spécifie trois options associées à l'outil Polygon :

```
<properties>
  <property name="Style"
    variable="style"
    list="polygon,star"
    defaultValue="0"
    type="Strings"  />

  <property name="Number of Sides"
    variable="nsides"
    min="3"
    max="32"
    defaultValue="5"
    type="Number"  />

  <property name="Star point size"
    variable="pointParam"
    min="0"
    max="1"
    defaultValue=".5"
    type="Double"  />

</properties>
```

## Exemple

L'exemple suivant spécifie que le fichier « PolyStar.xml » est associé à l'outil actif. Ce code est extrait du fichier d'exemple « PolyStar.jsfl » (consultez la section « [Exemple d'outil PolyStar](#) », à la page 20) :

```
theTool = fl.tools.activeTool;
theTool.setOptionsFile("PolyStar.xml");
```



# toolObj.setPI()

## Disponibilité

Flash MX 2004.

## Utilisation

```
toolObj.setPI(pi)
```

## Paramètres

*pi* Chaîne spécifiant l'inspecteur des propriétés à utiliser pour cet outil.

## Valeur renvoyée

Aucune.

## Description

Méthode ; définit un inspecteur des propriétés spécifique à utiliser lorsque l'outil est activé. Cette méthode n'est utilisée que lors de la création d'outils extensibles. Les valeurs gérées sont "shape" (forme, valeur par défaut), "text" et "movie" (film).

## Exemple

L'exemple suivant spécifie que l'inspecteur des propriétés des formes doit être utilisé lorsque l'outil est activé. Ce code est extrait du fichier d'exemple « PolyStar.jsfl » (consultez la section « [Exemple d'outil PolyStar](#) », à la page 20) :

```
theTool = fl.tools.activeTool;  
theTool.setPI("shape");
```

# toolObj.setToolName()

## Disponibilité

Flash MX 2004.

## Utilisation

```
toolObj.setToolName(name)
```

## Paramètres

*name* Chaîne spécifiant le nom de l'outil.

## Valeur renvoyée

Aucune.

## Description

Méthode ; affecte un nom à l'outil pour la configuration du panneau Outils. Cette méthode n'est utilisée que lors de la création d'outils extensibles. Ce nom est uniquement utilisé par le fichier XML de disposition que Flash lit pour construire le panneau Outils. Il n'apparaît donc pas dans l'interface utilisateur de Flash.

## Exemple

L'exemple suivant affecte le nom « polystar » à l'outil nommé `theTool`. Ce code est extrait du fichier d'exemple « PolyStar.jsfl » (consultez la section « [Exemple d'outil PolyStar](#) », à la page 20) :

```
theTool = fl.tools.activeTool;  
theTool.setToolName("polystar");
```

## toolObj.setToolTip()

### Disponibilité

Flash MX 2004.

### Utilisation

```
toolObj.setToolTip(toolTip)
```

### Paramètres

*toolTip* Chaîne spécifiant l'info-bulle à utiliser pour cet outil.

### Valeur renvoyée

Aucune.

## Description

Méthode ; définit l'info-bulle qui doit apparaître lorsque la souris est amenée sur l'icône de l'outil. Cette méthode n'est utilisée que lors de la création d'outils extensibles.

## Exemple

L'exemple suivant spécifie que l'info-bulle de l'outil doit être « PolyStar Tool ». Ce code est extrait du fichier d'exemple « PolyStar.jsfl » (consultez la section « [Exemple d'outil PolyStar](#) », à la page 20) :

```
theTool = fl.tools.activeTool;  
theTool.setToolTip("PolyStar Tool");
```

# toolObj.showPIControl()

## Disponibilité

Flash MX 2004.

## Utilisation

```
toolObj.showPIControl(control, bShow)
```

## Paramètres

*control* Chaîne spécifiant le nom du contrôle à afficher ou masquer. Cette méthode n'est utilisée que lors de la création d'outils extensibles. Les valeurs autorisées dépendent de l'inspecteur des propriétés appelé par cet outil (voir [toolObj.setPI\(\)](#)).

L'inspecteur des propriétés d'une forme possède les contrôles suivants :

stroke	fill
--------	------

L'inspecteur des propriétés d'un texte possède les contrôles suivants :

type	font	pointsize
color	bold	italic
direction	alignLeft	alignCenter
alignRight	alignJustify	spacing
position	autoKern	small
rotation	format	lineType
selectable	html	border
deviceFonts	varEdit	options
link	maxChars	target

L'inspecteur des propriétés d'un film possède les contrôles suivants :

size	publish	background
framerate	player	profile

*bShow* Valeur booléenne indiquant si le contrôle spécifié doit être affiché (*true*) ou masqué (*false*).

## Valeur renvoyée

Aucune.

## Description

Méthode ; affiche ou masque un contrôle dans l'inspecteur des propriétés. Cette méthode n'est utilisée que lors de la création d'outils extensibles.

## Exemple

Si la commande suivante est insérée dans le fichier JavaScript d'un outil extensible, Flash n'affiche pas les options de remplissage dans l'inspecteur des propriétés de cet outil :

```
fl.tools.activeTool.showPIControl("fill", false);
```

# toolObj.showTransformHandles()

## Disponibilité

Flash MX 2004.

## Utilisation

```
toolObj.showTransformHandles(bShow)
```

## Paramètres

*bShow* Valeur booléenne indiquant si les poignées de transformation libre de l'outil actif doivent être affichées (*true*) ou masquées (*false*).

## Valeur renvoyée

Aucune.

## Description

Méthode ; elle est appelée par la méthode `configureTool()` du fichier JavaScript d'un outil extensible pour indiquer que les poignées de transformation libre doivent apparaître lorsque l'outil est actif. Cette méthode n'est utilisée que lors de la création d'outils extensibles.

## Exemple

Voir `configureTool()`.

# Objet Tools

## Disponibilité

Flash MX 2004.

## Description

L'objet Tools est accessible à partir de l'objet flash ([fl.tools](#)). La propriété [tools.toolObj](#) contient un tableau des objets ToolObj, et la propriété [tools.activeTool](#) renvoie l'objet ToolObj de l'outil actuellement actif. (Consultez également les sections « [Objet ToolObj](#) », à la page 578 et « [Outils extensibles](#) », à la page 22.)

### REMARQUE

Les méthodes et propriétés suivantes sont uniquement utilisées lors de la création d'outils extensibles.

## Résumé des méthodes de l'objet Tools

Les méthodes suivantes sont disponibles avec l'objet Tools :

Méthode	Description
<a href="#">tools.constrainPoint()</a>	Lit deux points et renvoie un nouveau point ajusté ou <i>imposé</i> .
<a href="#">tools.getKeyDown()</a>	Renvoie la touche la plus récemment appuyée.
<a href="#">tools.setCursor()</a>	Donne l'apparence spécifiée au pointeur.
<a href="#">tools.snapPoint()</a>	Reçoit un point et renvoie un nouveau point qui peut être ajusté ou <i>accroché</i> à l'objet géométrique le plus proche.

## Résumé des propriétés de l'objet Tools

Les propriétés suivantes sont disponibles avec l'objet Tools :

Propriété	Description
<a href="#">tools.activeTool</a>	Propriété en lecture seule ; <a href="#">Objet ToolObj</a> de l'outil actif.
<a href="#">tools.altIsDown</a>	Propriété en lecture seule ; valeur booléenne qui indique si la touche Alt a été pressée.
<a href="#">tools.ctrlIsDown</a>	Propriété en lecture seule ; valeur booléenne qui indique si la touche Contrôle a été pressée.

Propriété	Description
<code>tools.mouseIsDown</code>	Propriété en lecture seule ; valeur booléenne qui indique si un bouton gauche de la souris est enfoncé.
<code>tools.penDownLoc</code>	Propriété en lecture seule ; point qui représente la position du dernier événement de bouton de souris enfoncé sur le scénario.
<code>tools.penLoc</code>	Propriété en lecture seule ; point représentant l'emplacement actuel de la souris.
<code>tools.shiftIsDown</code>	Propriété en lecture seule ; valeur booléenne qui indique si la touche Maj est actuellement enfoncée.
<code>tools.toolObjs</code>	Propriété en lecture seule ; tableau des objets Tools.

## tools.activeTool

### Disponibilité

Flash MX 2004.

### Utilisation

`tools.activeTool`

### Description

Propriété en lecture seule ; renvoie l'Objet `ToolObj` de l'outil actif.

### Exemple

L'exemple suivant enregistre un objet qui représente l'outil actif dans la variable `theTool` :

```
var theTool = fl.tools.activeTool;
```

## tools.altIsDown

### Disponibilité

Flash MX 2004.

### Utilisation

`tools.altIsDown`

## Description

Propriété en lecture seule ; valeur booléenne qui indique si la touche Alt est actuellement enfoncée. Sa valeur est `true` si la touche Alt est enfoncée, ou `false` sinon.

## Exemple

L'exemple suivant détermine si la touche Alt est enfoncée :

```
var isAltDown = fl.tools.altIsDown;
```

# tools.constrainPoint()

## Disponibilité

Flash MX 2004.

## Utilisation

```
tools.constrainPoint(pt1, pt2)
```

## Paramètres

*pt1* et *pt2* Points indiquant respectivement le point auquel le clic a eu lieu et le point jusqu'auquel le bouton de la souris est resté enfoncé.

## Valeur renvoyée

Nouveau point ajusté ou imposé.

## Description

Méthode ; reçoit deux points et renvoie un nouveau point ajusté ou imposé. Si la touche Maj est enfoncée lors de l'exécution de la commande, le point renvoyé est imposé soit pour contraindre un angle à 45° (par exemple pour une ligne terminée par une flèche), soit pour obliger un objet à conserver ses proportions (par exemple pour tracer un carré parfait à l'aide de l'outil Rectangle).

## Exemple

L'exemple suivant renvoie un point imposé :

```
pt2 = fl.tools.constrainPoint(pt1, tempPt);
```

## tools.ctrlIsDown

### Disponibilité

Flash MX 2004.

### Utilisation

`tools.ctrlIsDown`

### Description

Propriété en lecture seule ; valeur booléenne réglée sur `true` si la touche Contrôle est enfoncée, `false` dans le cas contraire.

### Exemple

L'exemple suivant détermine si la touche Contrôle est enfoncée :

```
var isCtrlDown = fl.tools.ctrlIsDown;
```

## tools.getKeyDown()

### Disponibilité

Flash MX 2004.

### Utilisation

`tools.getKeyDown()`

### Paramètres

Aucun.

### Valeur renvoyée

La valeur de la touche sur un entier.

### Description

Méthode ; renvoie la touche la plus récemment appuyée.

### Exemple

L'exemple suivant affiche la valeur d'entier de la touche la plus récemment enfoncée :

```
var theKey = fl.tools.getKeyDown();  
fl.trace(theKey);
```



# tools.mouseIsDown

## Disponibilité

Flash MX 2004.

## Utilisation

`tools.mouseIsDown`

## Description

Propriété en lecture seule ; valeur booléenne réglée sur `true` si le bouton gauche de la souris est enfoncé, `false` dans le cas contraire.

## Exemple

L'exemple suivant détermine si le bouton gauche de la souris est enfoncé.

```
var isMouseDown = fl.tools.mouseIsDown;
```

# tools.penDownLoc

## Disponibilité

Flash MX 2004.

## Utilisation

`tools.penDownLoc`

## Description

Propriété en lecture seule ; point qui représente la position du dernier événement de bouton de souris enfoncé sur la scène. La propriété `tools.penDownLoc` comprend deux propriétés, `x` et `y`, qui correspondent à la position `x,y` du pointeur de la souris.

## Exemple

L'exemple suivant détermine la position du dernier événement de bouton de souris enfoncé sur la scène et affiche les valeurs `x` et `y` dans le panneau Sortie :

```
var pt1 = fl.tools.penDownLoc;  
fl.trace("x,y location of last mouseDown event was " + pt1.x + ", " + pt1.y)
```

## Voir aussi

[tools.penLoc](#)

# tools.penLoc

## Disponibilité

Flash MX 2004.

## Utilisation

`tools.penLoc`

## Description

Propriété en lecture seule ; point représentant l'emplacement actuel de la souris. La propriété `tools.penLoc` comprend deux propriétés, *x* et *y*, qui correspondent à la position *x,y* du pointeur de la souris.

## Exemple

L'exemple suivant détermine l'emplacement actuel de la souris :

```
var tempPt = fl.tools.penLoc;
```

## Voir aussi

[tools.penDownLoc](#)

# tools.setCursor()

## Disponibilité

Flash MX 2004.

## Utilisation

`tools.setCursor(cursor)`

## Paramètres

*cursor* Entier définissant l'apparence du pointeur, décrite dans la liste suivante :

- 0 curseur Plus (+)
- 1 flèche noire
- 2 flèche blanche
- 3 quadruple flèche
- 4 flèche double horizontale
- 5 flèche double verticale
- 6 X
- 7 main

**Valeur renvoyée**

Aucune.

**Description**

Méthode ; donne l'apparence spécifiée au pointeur.

**Exemple**

L'exemple suivant donne l'aspect d'une flèche noire au pointeur.

```
fl.tools.setCursor(1);
```

## tools.shiftIsDown

**Disponibilité**

Flash MX 2004.

**Utilisation**

```
tools.shiftIsDown
```

**Description**

Propriété en lecture seule ; valeur booléenne réglée sur `true` si la touche Maj est enfoncée, `false` dans le cas contraire.

**Exemple**

L'exemple suivant détermine si la touche Maj est enfoncée.

```
var isShiftDown = fl.tools.shiftIsDown;
```

## tools.snapPoint()

**Disponibilité**

Flash MX 2004.

**Utilisation**

```
tools.snapPoint(pt)
```

**Paramètres**

*pt* Spécifie l'emplacement du point pour lequel un point d'accrochage doit être renvoyé.

### Valeur renvoyée

Nouveau point qui peut être ajusté ou accroché à l'objet géométrique le plus proche.

### Description

Méthode ; reçoit un point et renvoie un nouveau point qui peut être ajusté ou *accroché* à l'objet géométrique le plus proche. Si l'accrochage a été désactivé dans le menu Affichage de l'interface utilisateur de Flash, le point renvoyé est le point original.

### Exemple

L'exemple suivant renvoie un nouveau point qui peut être accroché à l'objet géométrique le plus proche.

```
var theSnapPoint = fl.tools.snapPoint(pt1);
```

## tools.toolObjs

### Disponibilité

Flash MX 2004.

### Utilisation

```
tools.toolObjs
```

### Description

Propriété en lecture seule ; tableau des objets Tools (consultez la section [Objet ToolObj](#)).

# Objet Vertex

## Disponibilité

Flash MX 2004.

## Description

L'objet Vertex est la partie de la structure de données d'une forme qui contient les coordonnées (sommets) de cette forme.

## Résumé des méthodes de l'objet Vertex

Vous pouvez utiliser les méthodes suivantes avec l'objet Vertex :

Méthode	Description
<code>vertex.getHalfEdge()</code>	Reçoit un <a href="#">Objet halfEdge</a> qui partage ce sommet.
<code>vertex.setLocation()</code>	Définit l'emplacement du sommet.

## Résumé des propriétés de l'objet Vertex

Les propriétés suivantes sont disponibles pour l'objet Vertex :

Propriété	Description
<code>vertex.x</code>	Propriété en lecture seule ; emplacement de la coordonnée x du sommet, en pixels.
<code>vertex.y</code>	Propriété en lecture seule ; emplacement de la coordonnée y du sommet, en pixels.

## vertex.getHalfEdge()

### Disponibilité

Flash MX 2004.

### Utilisation

```
vertex.getHalfEdge()
```

### Paramètres

Aucun.

### Valeur renvoyée

Un [Objet halfEdge](#)

## Description

Méthode ; reçoit un [Objet halfEdge](#) qui partage ce sommet.

## Exemple

L'exemple suivant montre comment obtenir les autres demi-contours qui partagent le même sommet :

```
var shape = fl.getDocumentDOM().selection[0];
var hEdge = shape.edges[0].getHalfEdge(0);
var theVertex = hEdge.getVertex();
var someHEdge = theVertex.getHalfEdge(); // Pas nécessairement le
// même demi-contour
var theSameVertex = someHEdge.getVertex();
fl.trace('the same vertex: ' + theSameVertex);
```

## vertex.setLocation()

### Disponibilité

Flash MX 2004.

### Utilisation

```
vertex.setLocation(x, y)
```

### Paramètres

*x* Valeur en virgule flottante indiquant la coordonnée *x* de la position souhaitée pour le sommet, en pixels.

*y* Valeur en virgule flottante indiquant la coordonnée *y* de la position souhaitée pour le sommet, en pixels.

### Valeur renvoyée

Aucune.

### Description

Méthode ; définit l'emplacement du sommet. Il est nécessaire d'appeler [shape.beginEdit\(\)](#) avant d'utiliser cette méthode.

## Exemple

L'exemple suivant amène le sommet sélectionné au point d'origine :

```
var shape = fl.getDocumentDOM().selection[0];
shape.beginEdit();
var hEdge = shape.edges[0].getHalfEdge(0);
var vertex = hEdge.getVertex();
var someHEdge = vertex.getHalfEdge();
```

```
var vertex = someHEdge.getVertex();  
// Placer le sommet à son point d'origine.  
vertex.setLocation(0.0, 0.0);  
shape.endEdit();
```

## vertex.x

### Disponibilité

Flash MX 2004.

### Utilisation

vertex.x

### Description

Propriété en lecture seule ; l'emplacement de la coordonnée  $x$  du sommet, en pixels.

### Exemple

L'exemple suivant affiche dans le panneau de sortie l'emplacement des coordonnées  $x$  et  $y$  du sommet :

```
var shape = fl.getDocumentDOM().selection[0];  
var hEdge = shape.edges[0].getHalfEdge(0);  
var vertex = hEdge.getVertex();  
  
fl.trace('x location of vertex is: ' + vertex.x);  
fl.trace('y location of vertex is: ' + vertex.y);
```

## vertex.y

### Disponibilité

Flash MX 2004.

### Utilisation

vertex.y

### Description

Propriété en lecture seule ; emplacement de la coordonnée  $y$  du sommet, en pixels.

### Exemple

Voir [vertex.x](#).

# Objet XMLUI

## Disponibilité

Flash MX 2004.

## Description

Flash 8 prend en charge des boîtes de dialogue personnalisées écrites à l'aide d'un sous-ensemble du langage XUL (XML User Interface Language). Une boîte de dialogue XMLUI (XML User Interface) peut être utilisée par plusieurs fonctionnalités de Flash, telles que les commandes et les comportements, pour offrir une interface utilisateur à de nouvelles fonctionnalités créées avec l'API d'extensibilité. L'objet XMLUI permet de lire et définir les propriétés d'une boîte de dialogue XMLUI, et d'en accepter ou annuler une. Les méthodes XMLUI peuvent être utilisées dans des fonctions de rappel, telles que les gestionnaires `oncommand` des boutons.

Il est possible de créer un fichier `dialog.xml` et de l'appeler à partir de l'API JavaScript avec la méthode `document.xmlPanel()`. Pour rechercher un objet représentant la boîte de dialogue XMLUI en cours, utilisez `fl.xmlui`.

## Résumé des méthodes de l'objet XMLUI

Les méthodes suivantes sont disponibles avec l'objet XMLUI :

Méthode	Description
<code>xmlui.accept()</code>	Ferme la boîte de dialogue XMLUI actuelle avec l'état d'acceptation.
<code>xmlui.cancel()</code>	Ferme la boîte de dialogue XMLUI active avec l'état d'annulation.
<code>xmlui.get()</code>	Extrait la valeur de la propriété spécifiée de la boîte de dialogue XMLUI actuelle.
<code>xmlui.getControlItemElement()</code>	Renvoie l'élément de contrôle actif pour le contrôle spécifié.
<code>xmlui.getEnabled()</code>	Renvoie une valeur booléenne qui spécifie si le contrôle est activé ou désactivé (affiché en grisé).
<code>xmlui.getVisible()</code>	Renvoie une valeur booléenne qui spécifie si le contrôle est visible ou masqué.
<code>xmlui.set()</code>	Modifie la valeur de la propriété spécifiée de la boîte de dialogue XMLUI actuelle.



Méthode	Description
<code>xmlui.setControlItemElement()</code>	Définit le libellé et la valeur de l'élément actif.
<code>xmlui.setControlItemElements()</code>	Définit les paires libellé et valeur de l'élément actif.
<code>xmlui.setEnabled()</code>	Active ou désactive (affiche en grisé) un contrôle.
<code>xmlui.setVisible()</code>	Affiche ou masque un contrôle.

## xmlui.accept()

### Disponibilité

Flash MX 2004.

### Utilisation

`xmlui.accept()`

### Paramètres

Aucun.

### Valeur renvoyée

Aucune.

### Description

Méthode ; ferme la boîte de dialogue XMLUI avec l'état d'acceptation, ce qui est équivalent à un clic sur le bouton OK.

### Voir aussi

`fl.xmlui`, `document.xmlPanel()`, `xmlui.cancel()`

## xmlui.cancel()

### Disponibilité

Flash MX 2004.

### Utilisation

`xmlui.cancel()`

### Paramètres

Aucun.

## Valeur renvoyée

Aucune.

## Description

Méthode ; ferme la boîte de dialogue XMLUI avec l'état d'annulation, ce qui est équivalent à un clic sur le bouton Annuler.

## Voir aussi

`fl.xmlui`, `document.xmlPanel()`, `xmlui.accept()`

# xmlui.get()

## Disponibilité

Flash MX 2004.

## Utilisation

`xmlui.get(controlPropertyName)`

## Paramètres

*controlPropertyName* Chaîne spécifiant le nom de la propriété XMLUI dont vous recherchez la valeur.

## Valeur renvoyée

Chaîne représentant la valeur de la propriété spécifiée. Dans les cas où une valeur booléenne `true` ou `false` est attendue, cette méthode renvoie la chaîne `"true"` ou `"false"`.

## Description

Méthode ; lit la valeur de la propriété spécifiée de la boîte de dialogue XMLUI active.

## Exemple

L'exemple suivant renvoie la valeur de la propriété « URL » :

```
fl.xmlui.get("URL");
```

## Voir aussi

`fl.xmlui`, `document.xmlPanel()`, `xmlui.getControlItemElement()`, `xmlui.set()`

# xmlui.getControlItemElement()

## Disponibilité

Flash 8.

## Utilisation

```
xmlui.getControlItemElement(controlPropertyName)
```

## Paramètres

*controlPropertyName* Chaîne spécifiant la propriété dont vous souhaitez récupérer l'élément de contrôle.

## Valeur renvoyée

Objet représentant l'élément de contrôle actif pour le contrôle spécifié par *controlPropertyName*.

## Description

Méthode ; renvoie le libellé et la valeur de la ligne sélectionnée dans un contrôle de type zone de liste (ListBox) ou zone déroulante (ComboBox) pour le contrôle spécifié par le paramètre *controlPropertyName*.

## Exemple

L'exemple suivant renvoie le libellé et la valeur de la ligne actuellement sélectionnée pour le contrôle myListBox :

```
var elem = new Object();  
elem = fl.xmlui.getControlItemElement("myListBox");  
fl.trace("label = " + elem.label + " value = " + elem.value);
```

## Voir aussi

```
fl.xmlui, document.xmlPanel(), xmlui.get(), xmlui.setControlItemElement(),  
xmlui.setControlItemElements()
```

# xmlui.setEnabled()

## Disponibilité

Flash 8.

## Utilisation

```
xmlui.setEnabled(controlID)
```

## Paramètres

*controlID* Chaîne spécifiant l'attribut ID du contrôle dont vous recherchez l'état.

## Valeur renvoyée

Valeur booléenne `true` si le contrôle est activé, `false` dans le cas contraire.

## Description

Méthode ; renvoie une valeur booléenne qui spécifie si le contrôle est activé ou désactivé (affiché en grisé).

## Exemple

L'exemple suivant renvoie une valeur indiquant si le contrôle dont l'attribut ID correspond à `myListBox` est activé :

```
var isEnabled = fl.xmlui.setEnabled("myListBox");  
fl.trace(isEnabled);
```

## Voir aussi

[fl.xmlui](#), [document.xmlPanel\(\)](#), [xmlui.setEnabled\(\)](#)

# xmlui.getVisible()

## Disponibilité

Flash 8.

## Utilisation

```
xmlui.getVisible(controlID)
```

## Paramètres

*controlID* Chaîne spécifiant l'attribut ID du contrôle dont vous recherchez l'état de visibilité.

## Valeur renvoyée

Valeur booléenne `true` si le contrôle est visible, ou `false` s'il est invisible (masqué).

## Description

Méthode ; renvoie une valeur booléenne qui spécifie si le contrôle est visible ou masqué.

## Exemple

L'exemple suivant renvoie une valeur indiquant si le contrôle dont l'attribut ID correspond à `myListBox` est visible :

```
var isVisible = fl.xmlui.getVisible("myListBox");  
fl.trace(isVisible);
```

## Voir aussi

[xmlui.setVisible\(\)](#)

# xmlui.set()

## Disponibilité

Flash MX 2004.

## Utilisation

```
xmlui.set(controlPropertyName, value)
```

## Paramètres

*controlPropertyName* Chaîne spécifiant le nom de la propriété XMLUI à modifier.

*value* Chaîne spécifiant la valeur à donner à la propriété XMLUI.

## Valeur renvoyée

Aucune.

## Description

Méthode ; modifie la valeur de la propriété spécifiée de la boîte de dialogue XMLUI active.

## Exemple

L'exemple suivant définit la valeur d'une propriété nommée URL sur « `www.adobe.com` » :

```
fl.xmlui.set("URL", "www.adobe.com");
```

## Voir aussi

[fl.xmlui](#), [document.xmlPanel\(\)](#), [xmlui.get\(\)](#), [xmlui.setControlItemElement\(\)](#),  
[xmlui.setControlItemElements\(\)](#)

# xmlui.setControlItemElement()

## Disponibilité

Flash 8.

## Utilisation

```
xmlui.setControlItemElement(controlPropertyName, elementItem)
```

## Paramètres

*controlPropertyName* Chaîne spécifiant l'élément de contrôle à définir.

*elementItem* Objet JavaScript doté d'une propriété de chaîne nommée `label` et d'une propriété de chaîne facultative nommée `value`. Si la propriété `value` n'existe pas, elle est créée et reçoit la même valeur que `label`.

## Valeur renvoyée

Aucune.

## Description

Méthode ; définit le libellé et la valeur de la ligne actuellement sélectionnée dans le contrôle de type zone de liste (ListBox) ou zone déroulante (ComboBox) spécifié par le paramètre *controlPropertyName*.

## Exemple

L'exemple suivant définit le libellé et la valeur de l'élément actif de la propriété de contrôle

"PhoneNumber" :

```
var elem = new Object();  
elem.label = "Fax";  
elem.value = "707-555-5555";  
fl.xmlui.setControlItemElement("PhoneNumber",elem);
```

## Voir aussi

[fl.xmlui, document.xmlPanel\(\), xmlui.getControlItemElement\(\), xmlui.set\(\), xmlui.setControlItemElements\(\)](#)

# xmlui.setControlItemElements()

## Disponibilité

Flash 8.

## Utilisation

```
xmlui.setControlItemElements(controlID, elementItemArray)
```

## Paramètres

*controlID* Chaîne spécifiant l'attribut ID du contrôle à définir.

*elementItemArray* Tableau constitué d'objets JavaScript dotés chacun d'une propriété de chaîne nommée *label* et d'une propriété de chaîne facultative nommée *value*. Si la propriété *value* n'existe pas, elle est créée et reçoit la même valeur que *label*.

## Valeur renvoyée

Aucune.

## Description

Méthode ; vide les valeurs du contrôle de type zone de liste ou zone déroulante spécifié par *controlID* et remplace les éléments de liste ou les commandes de menu par les paires *label*, *value* spécifiées par *elementItemArray*.

## Exemple

L'exemple suivant règle le libellé et la valeur des éléments associés au contrôle possédant l'attribut d'ID *myControlID* sur les paires *label*, *value* spécifiées :

```
var nameArray = new Array("January", "February", "March");
var monthArray = new Array();
for (i=0;i<nameArray.length;i++){
    elem = new Object();
    elem.label = nameArray[i];
    elem.value = i;
    monthArray[i] = elem;
}
fl.xmlui.setControlItemElements("myControlID", monthArray);
```

## Voir aussi

[xmlui.getControlItemElement\(\)](#), [xmlui.set\(\)](#), [xmlui.setControlItemElement\(\)](#)

# xmlui.setEnabled()

## Disponibilité

Flash 8.

## Utilisation

`xmlui.setEnabled(controlID, enable)`

## Paramètres

*controlID* Chaîne spécifiant l'attribut ID du contrôle à activer ou désactiver.

*enable* Valeur booléenne *true* si vous souhaitez que le contrôle soit activé, *false* si vous voulez le désactiver (l'afficher en grisé).

### Valeur renvoyée

Aucune.

### Description

Méthode ; active ou désactive (affiche en grisé) un contrôle.

### Exemple

L'exemple suivant affiche en grisé le contrôle dont l'attribut ID est réglé sur `myControl` :

```
fl.xmlui.setEnabled("myControl", false);
```

### Voir aussi

[xmlui.setEnabled\(\)](#)

## xmlui.setVisible()

### Disponibilité

Flash 8.

### Utilisation

```
xmlui.setVisible(controlID, visible)
```

### Paramètres

*controlID* Chaîne spécifiant l'attribut ID du contrôle à afficher ou masquer.

*visible* Valeur booléenne *true* si vous voulez afficher le contrôle, *false* si vous voulez le masquer.

### Valeur renvoyée

Aucune.

### Description

Méthode ; affiche ou masque un contrôle.

### Exemple

L'exemple suivant masque le contrôle dont l'attribut ID est réglé sur `myControl` :

```
fl.xmlui.setVisible("myControl", false);
```

### Voir aussi

[xmlui.setVisible\(\)](#)



# Objet VideoItem

**Héritage** [Objet Item](#) > objet VideoItem

## Disponibilité

Flash MX 2004.

## Description

L'objet VideoItem est une sous-classe de l'[Objet Item](#).

## Résumé des propriétés de l'objet VideoItem

Outre les propriétés de l'[Objet Item](#), l'objet VideoItem gère les propriétés suivantes :

Propriété	Description
<a href="#">videoItem.sourceFilePath</a>	Propriété en lecture seule ; chaîne représentant le chemin d'accès de l'élément vidéo.
<a href="#">videoItem.videoType</a>	Propriété en lecture seule ; chaîne spécifiant le type de vidéo représenté par l'élément.

## videoItem.sourceFilePath

### Disponibilité

Flash 8.

### Utilisation

`videoItem.sourceFilePath`

### Description

Propriété en lecture seule ; chaîne exprimée sous la forme « fichier:/// URI » et représentant le chemin d'accès de l'élément vidéo.

### Exemple

L'exemple suivant affiche le nom et le chemin d'accès source de tout élément de la bibliothèque de type "video" :

```
for (idx in fl.getDocumentDOM().library.items) {  
    if (fl.getDocumentDOM().library.items[idx].itemType == "video") {  
        var myItem = fl.getDocumentDOM().library.items[idx];  
        fl.trace(myItem.name + " source is " + myItem.sourceFilePath);  
    }  
}
```

## Voir aussi

[library.items](#)

# videoItem.videoType

## Disponibilité

Flash 8.

## Utilisation

`videoItem.videoType`

## Description

Propriété en lecture seule ; chaîne spécifiant le type de vidéo représenté par l'élément.

Les valeurs gérées sont "embedded video", "linked video" et "video".

## Exemple

L'exemple suivant affiche le nom et le type de tout élément de la bibliothèque de type "video" :

```
for (idx in fl.getDocumentDOM().library.items) {  
    if (fl.getDocumentDOM().library.items[idx].itemType == "video") {  
        var myItem = fl.getDocumentDOM().library.items[idx];  
        fl.trace(myItem.name + " is " + myItem.videoType);  
    }  
}
```

## Voir aussi

[library.items](#)

Le mécanisme d'extensibilité de niveau C permet d'implémenter des fichiers d'extensions Adobe Flash CS3 Professional à l'aide d'une combinaison de codes JavaScript et C personnalisé. Les fonctions sont définies en C, puis regroupées dans une DLL ou une bibliothèque partagée qui est ensuite enregistrée dans le répertoire approprié ; ces fonctions peuvent alors être appelées en JavaScript à l'aide de l'API JavaScript d'Adobe Flash.

Le mécanisme d'extensibilité de niveau C a été introduit dans Macromedia Flash MX 2004 et n'a pas été modifié dans Flash 8. La seule révision effectuée dans Flash CS3 concerne un changement opéré dans le fichier `mm_jsapi.h` qui est contenu dans le fichier d'exemple ZIP ou SIT (voir « [Exemple d'une mise en œuvre de DLL](#) », à la page 618).

Par exemple, il est possible de définir en C une fonction effectuant des calculs intenses beaucoup plus rapidement qu'en JavaScript, ou de créer des outils ou des effets très sophistiqués.

Ce mécanisme d'extensibilité est un sous-ensemble de l'API d'Adobe Dreamweaver CS3. Si vous connaissez déjà cette API, les fonctions de l'API du mécanisme d'extensibilité de niveau C ne vous sembleront pas vraiment étrangères. Toutefois, cette API diffère de l'API de Dreamweaver sur certains points :

- Cette API ne contient pas toutes les commandes de l'API de Dreamweaver.
- Toutes les déclarations qui sont de type `wchar_t` et `char` dans l'API de Dreamweaver sont implémentées comme déclarations `unsigned short` dans celle-ci, afin d'assurer la prise en charge d'Unicode pour le passage de chaînes.
- La fonction `JSVal JS_BytesToValue()` de cette API ne figure pas dans l'API de Dreamweaver.
- L'emplacement où les DLL et les fichiers de bibliothèques partagées doivent être stockés est différent (voir « [Intégration de fonctions en C](#) », à la page 612).

# Intégration de fonctions en C

Le mécanisme d'extensibilité de niveau C permet d'implémenter des fichiers d'extensions Flash à l'aide d'une combinaison de codes JavaScript et C. Le processus d'implémentation de cette possibilité se résume ainsi :

1. Définir des fonctions en langage C ou C++.
2. Regrouper ces fonctions dans une DLL (Windows) ou une bibliothèque partagée (Macintosh).
3. Enregistrer le fichier de la DLL ou de la bibliothèque à l'emplacement approprié :
  - Windows 2000 ou Windows XP :  
`lecteur d'amorçage\Documents and Settings\utilisateur\Local Settings\Application Data\Adobe\Flash CS3\langue\Configuration\External Libraries`
  - Mac OS X :  
`Disque dur Macintosh/Utilisateurs/utilisateur/Library/Application Support/Adobe/Flash CS3/langue/Configuration/External Libraries`
4. Créer un fichier JSFL qui appelle ces fonctions.
5. Exécuter le fichier JSFL à partir du menu Commandes dans l'environnement de programmation Flash.

Pour plus d'informations, voir la section « [Exemple d'une mise en œuvre de DLL](#) », à la page 618.

## Extensibilité de niveau C et interpréteur JavaScript

L'interaction entre l'API JavaScript de Flash et le code C contenu dans la DLL ou la bibliothèque partagée comporte trois étapes :

- Au démarrage, pour enregistrer les fonctions de la bibliothèque
- Lors de l'appel de la fonction en C, pour transmettre les arguments passés de JavaScript en C
- Avant le retour de la fonction C, pour transmettre la valeur renvoyée

Pour accomplir ces tâches, l'interpréteur définit plusieurs types de données et expose une API. Les définitions des types de données et des fonctions présentées dans cette section apparaissent dans le fichier `mm_jsapi.h`. Pour permettre à votre bibliothèque de fonctionner correctement, vous devez indiquer en « `include` » le fichier `mm_jsapi.h` au début de chaque fichier de votre bibliothèque, à l'aide de la ligne suivante :

```
#include "mm_jsapi.h"
```

Cette inclusion du fichier mm\_jsapi.h comprend également le fichier mm\_jsapi\_environment.h qui définit la structure MM\_Environment.

Pour obtenir le fichier mm\_jsapi.h, vous pouvez l'extraire du fichier d'exemple ZIP ou SIT (voir « [Exemple d'une mise en œuvre de DLL](#) », à la page 618) ou copier le code ci-dessous dans un fichier nommé mm\_jsapi.h :

```
#ifndef _MM_JSAPI_H_
#define _MM_JSAPI_H_

/
*****
* Public data types
*****

typedef struct JSContext JSContext;
typedef struct JSObject JSObject;
typedef long jsval;
#ifndef JSBool
typedef long JSBool;
#endif

typedef JSBool (*JSNative)(JSContext *cx, JSObject *obj, unsigned int argc,
                           jsval *argv, jsval *rval);

/* Valeurs acceptées pour JSBool */
#define JS_TRUE 1
#define JS_FALSE 0

/
*****
* Public functions
*****

/* JSBool JS_DefineFunction(unsigned short *name, JSNative call, unsigned
   int nargs) */
#define JS_DefineFunction(n, c, a) \
    (mmEnv.defineFunction ? (*(mmEnv.defineFunction))(mmEnv.libObj, n, c, \
    a) \
    : JS_FALSE)
```

```

/* unsigned short *JS_ValueToString(JSContext *cx, jsval v, unsigned int
 *pLength) */
#define JS_ValueToString(c, v, l) \
    (mmEnv.valueToString ? (*(mmEnv.valueToString))(c, v, l) : (char *)0)

/* unsigned char *JS_ValueToBytes(JSContext *cx, jsval v, unsigned int
 *pLength) */
#define JS_ValueToBytes(c, v, l) \
    (mmEnv.valueToBytes ? (*(mmEnv.valueToBytes))(c, v, l) : (unsigned char
 * *)0)

/* JSBool JS_ValueToInteger(JSContext *cx, jsval v, long *lp); */
#define JS_ValueToInteger(c, v, l) \
    (mmEnv.valueToInteger ? (*(mmEnv.valueToInteger))(c, v, l) : JS_FALSE)

/* JSBool JS_ValueToDouble(JSContext *cx, jsval v, double *dp); */
#define JS_ValueToDouble(c, v, d) \
    (mmEnv.valueToDouble ? (*(mmEnv.valueToDouble))(c, v, d) : JS_FALSE)

/* JSBool JS_ValueToBoolean(JSContext *cx, jsval v, JSBool *bp); */
#define JS_ValueToBoolean(c, v, b) \
    (mmEnv.valueToBoolean ? (*(mmEnv.valueToBoolean))(c, v, b) : JS_FALSE)

/* JSBool JS_ValueToObject(JSContext *cx, jsval v, JSObject **op); */
#define JS_ValueToObject(c, v, o) \
    (mmEnv.valueToObject ? (*(mmEnv.valueToObject))(c, v, o) : JS_FALSE)

/* JSBool JS_StringToValue(JSContext *cx, unsigned short *bytes, uint sz,
 jsval *vp); */
#define JS_StringToValue(c, b, s, v) \
    (mmEnv.stringToValue ? (*(mmEnv.stringToValue))(c, b, s, v) : JS_FALSE)

/* JSBool JS_BytesToValue(JSContext *cx, unsigned char *bytes, uint sz,
 jsval *vp); */
#define JS_BytesToValue(c, b, s, v) \
    (mmEnv.bytesToValue ? (*(mmEnv.bytesToValue))(c, b, s, v) : JS_FALSE)

/* JSBool JS_DoubleToValue(JSContext *cx, double dv, jsval *vp); */
#define JS_DoubleToValue(c, d, v) \
    (mmEnv.doubleToValue ? (*(mmEnv.doubleToValue))(c, d, v) : JS_FALSE)

/* jsval JS_IntegerToValue(long lv); */
#define JS_IntegerToValue(lv) \
    (((jsval)(lv) << 1) | 0x1)

/* jsval JS_BooleanToValue(JSBool bv); */
#define JS_BooleanToValue(bv) \
    (((jsval)(bv) << 3) | 0x6)

/* jsval JS_ObjectToValue(JSObject *obj); */
#define JS_ObjectToValue(ov) \
    ((jsval)(ov))

```

```

/* unsigned short *JS_ObjectType(JSObject *obj); */
#define JS_ObjectType(o) \
    (mmEnv.objectType ? (*(mmEnv.objectType))(o) : (char *)0)

/* JSObject *JS_NewArrayObject(JSContext *cx, unsigned int length, jsval
    *v) */
#define JS_NewArrayObject(c, l, v) \
    (mmEnv.newArrayObject ? (*(mmEnv.newArrayObject))(c, l, v) : (JSObject
    *)0)

/* long JS_GetArrayLength(JSContext *cx, JSObject *obj) */
#define JS_GetArrayLength(c, o) \
    (mmEnv.getArrayLength ? (*(mmEnv.getArrayLength))(c, o) : -1)

/* JSBool JS_GetElement(JSContext *cx, JSObject *obj, jsint idx, jsval *vp)
    */
#define JS_GetElement(c, o, i, v) \
    (mmEnv.getElement ? (*(mmEnv.getElement))(c, o, i, v) : JS_FALSE)

/* JSBool JS_SetElement(JSContext *cx, JSObject *obj, jsint idx, jsval *vp)
    */
#define JS_SetElement(c, o, i, v) \
    (mmEnv.setElement ? (*(mmEnv.setElement))(c, o, i, v) : JS_FALSE)

/* JSBool JS_ExecuteScript(JSContext *cx, JSObject *obj, unsigned short
    *script,
    * unsigned int sz, jsval *rval) */
#define JS_ExecuteScript(c, o, s, z, r) \
    (mmEnv.executeScript ? (*(mmEnv.executeScript))(c, o, s, z,
    (LPCTSTR)___FILE__, \
    ___LINE__, r) : JS_FALSE)

/* JSBool JS_ReportError(JSContext *cx, unsigned short *error, unsigned int
    sz) */
#define JS_ReportError(c, e, s) \
    (mmEnv.reportError ? (*(mmEnv.reportError))(c, e, s) : JS_FALSE)

/
*****
*****
* Private data types, macros, and globals

*****
*****/

typedef struct {
    JSObject *libObj;

```

```

    JSBool (*defineFunction)(JSObject *libObj, unsigned short *name,
JSNative call,
        unsigned int nargs);
    unsigned short *(*valueToString)(JSContext *cx, jsval v, unsigned int
*pLength);
    unsigned char *(*valueToBytes)(JSContext *cx, jsval v, unsigned int
*pLength);
    JSBool (*valueToInteger)(JSContext *cx, jsval v, long *lp);
    JSBool (*valueToDouble)(JSContext *cx, jsval v, double *dp);
    JSBool (*valueToBoolean)(JSContext *cx, jsval v, JSBool *bp);
    JSBool (*valueToObject)(JSContext *cx, jsval v, JSObject **op);
    JSBool (*stringToValue)(JSContext *cx, unsigned short *b, unsigned int
sz, jsval *vp);
    JSBool (*bytesToValue)(JSContext *cx, unsigned char *b, unsigned int sz,
jsval *vp);
    JSBool (*doubleToValue)(JSContext *cx, double dv, jsval *vp);
    unsigned short *(*objectType)(JSObject *obj);
    JSObject *(*newArrayObject)(JSContext *cx, unsigned int length, jsval
*vp);
    long (*getArrayLength)(JSContext *cx, JSObject *obj);
    JSBool (*getElement)(JSContext *cx, JSObject *obj, unsigned int idx,
jsval *vp);
    JSBool (*setElement)(JSContext *cx, JSObject *obj, unsigned int idx,
jsval *vp);
    JSBool (*executeScript)(JSContext *cx, JSObject *obj, unsigned short
*script,
        unsigned int sz, unsigned short *file, unsigned int lineNum, jsval
*rval);
    JSBool (*reportError)(JSContext *cx, unsigned short *error, unsigned int
sz);
} MM_Environment;

extern MM_Environment mmEnv;

// Déclaration du point d'entrée externe et de la liaison
#ifdef _WIN32
#   ifndef _MAC
        // Windows
        __declspec( dllexport ) void MM_InitWrapper( MM_Environment *env,
            unsigned int envSize );
#   endif
#else
extern void MM_InitWrapper( MM_Environment *env, unsigned int envSize );
#endif

#define MM_STATE
/* Définitions des variables globales */
\
MM_Environment mmEnv;
\

```



```

void                                                                    \
MM_InitWrapper(MM_Environment *env, unsigned int envSize)              \
\
{                                                                        \
    extern void MM_Init();                                              \
    \
    char **envPtr = (char **)env;                                       \
    \
    char **mmPtr = (char **>(&mmEnv);                                   \
    \
    char **envEnd = (char **)((char *)envPtr + envSize);               \
    \
    char **mmEnd = (char **)((char *)mmPtr + sizeof(MM_Environment));  \
    \
    /* Copier les champs de env dans mmEnv, un pointeur à la fois */    \
    \
    while (mmPtr < mmEnd && envPtr < envEnd)                             \
    \
        *mmPtr++ = *envPtr++;                                           \
    \
    /* Si env ne définit pas tous les champs de mmEnv, définir les champs \
    supplémentaires sur NULL */                                          \
    while (mmPtr < mmEnd)                                                \
        *mmPtr++ = (char *)0;                                           \
    \
    /* Appeler la fonction MM_Init de l'utilisateur */                 \
    \
    MM_Init();                                                           \
    \
}                                                                        \
\
#endif /* _MM_JSAPI_H_ */

```

## Exemple d'une mise en œuvre de DLL

Vous trouverez un exemple de mise en œuvre d'une DLL dans les fichiers ZIP et SIT du dossier ExtendingFlash/dllSampleComputeSum (voir « [Exemples d'implémentation](#) », à la page 19). Pour comprendre le fonctionnement de ce processus sans compiler la DLL, procédez comme suit :

- Enregistrer le fichier Sample.jsfl dans le répertoire Configuration/Commands (voir « [Enregistrement de fichiers JSFL](#) », à la page 7).
- Enregistrer le fichier Sample.dll dans le répertoire Configuration/External Libraries (voir « [Intégration de fonctions en C](#) », à la page 612).
- Dans l'environnement de programmation de Flash, choisissez Commandes > Sample. Les instructions de trace du fichier JSFL affichent dans le panneau de sortie les résultats de la fonction définie dans Sample.dll.

Cette section explique le développement de l'exemple. Dans ce cas précis, la DLL ne contient qu'une fonction, qui additionne deux chiffres. Le code C est présenté dans l'exemple suivant :

```
// Code source en C
// Enregistrer la DLL ou la bibliothèque partagée sous le nom "Sample".
#include <windows.h>
#include <stdlib.h>

#include "mm_jsapi.h"

// Exemple de fonction
// Chaque implémentation d'une fonction JavaScript doit comporter
// cette signature.
JSBool computeSum(JSContext *cx, JSObject *obj, unsigned int argc, jsval
*argv, jsval *rval)
{
    long a, b, sum;

    // Vérifier le nombre d'arguments passés.
    if (argc != 2)
        return JS_FALSE;

    // Convertir les deux arguments de jsvals en longs.
    if (JS_ValueToInteger(cx, argv[0], &a) == JS_FALSE ||
        JS_ValueToInteger(cx, argv[1], &b) == JS_FALSE)
        return JS_FALSE;

    /* Effectuer les tâches à proprement parler. */
    sum = a + b;

    /* Conditionner la valeur renvoyée sous forme de jsval. */
    *rval = JS_IntegerToValue(sum);

    /* Indiquer le succès de l'opération. */
    return JS_TRUE;
}
```

Après l'écriture de ce code, constituez le fichier DLL ou la bibliothèque partagée pour l'enregistrer ensuite dans le répertoire approprié, Configuration/External Libraries (voir « [Intégration de fonctions en C](#) », à la page 612). Créez ensuite un fichier JSFL contenant le code suivant et enregistrez-le dans le répertoire Configuration/Commands (voir « [Enregistrement de fichiers JSFL](#) », à la page 7).

```
// Fichier JSFL exécutant la fonction C définie ci-dessus.  
var a = 5;  
var b = 10;  
var sum = Sample.computeSum(a, b);  
fl.trace("The sum of " + a + " and " + b + " is " + sum );
```

Pour exécuter la fonction définie dans la DLL, choisissez Commandes > Sample dans l'environnement de programmation Flash.

## Types de données

L'interpréteur JavaScript définit les types de données suivants :

- JSContext
- JSObject
- jsval
- JSBool

### typedef struct JSContext JSContext

Un pointeur sur ce type de données opaque est passé à la fonction en C. Certaines fonctions de l'API acceptent ce pointeur parmi leurs arguments.

### typedef struct JSObject JSObject

Un pointeur sur ce type de données opaque est passé à la fonction en C. Ce type de données représente un objet, qui peut être de type tableau ou autre.

### typedef struct jsval jsval

Structure de données opaque pouvant contenir un entier ou un pointeur vers un nombre en virgule flottante, une chaîne ou un objet. Certaines fonctions de l'API peuvent lire les valeurs des arguments des fonctions via le contenu d'une structure `jsval`, et certaines permettent d'écrire la valeur renvoyée par la fonction dans une structure `jsval`.

```
typedef enum { JS_FALSE = 0, JS_TRUE = 1 }
JSBool
```

Type de données simple contenant une valeur booléenne.

## API de niveau C

L'API d'extensibilité de niveau C se compose de la signature de fonction JSBool (\*JSNative) et des fonctions suivantes :

- JSBool JS\_DefineFunction()
- unsigned short \*JS\_ValueToString()
- JSBool JS\_ValueToInteger()
- JSBool JS\_ValueToInteger()
- JSBool JS\_ValueToBoolean()
- JSBool JS\_ValueToObject()
- JSBool JS\_StringToValue()
- JSBool JS\_DoubleToValue()
- JSVal JS\_BooleanToValue()
- JSVal JS\_BytesToValue()
- JSVal JS\_IntegerToValue()
- JSVal JS\_ObjectToValue()
- unsigned short \*JS\_ObjectType()
- JSObject \*JS\_NewArrayObject()
- long JS\_GetArrayLength()
- JSBool JS\_GetElement()
- JSBool JS\_SetElement()
- JSBool JS\_ExecuteScript()

```
typedef JSBool (*JSNative)(JSContext *cx,  
JSObject *obj, unsigned int argc, jsval *argv,  
jsval *rval)
```

### Description

Méthode qui décrit les implémentations de niveau C des fonctions JavaScript dans les situations suivantes :

- Le pointeur *cx* pointe sur une structure opaque, *JSContext*, qui doit être passée à certaines fonctions de l'API JavaScript. Cette variable contient le contexte d'exécution de l'interpréteur.
- Le pointeur *obj* pointe sur l'objet dans le contexte duquel le script s'exécute. Pendant l'exécution du script, le mot-clé *this* est égal à cet objet.
- L'entier *argc* est le nombre d'arguments passés à la fonction.
- Le pointeur *argv* pointe sur un tableau de structures *jsval*. La longueur du tableau correspond au nombre d'éléments défini par *argc*.
- Le pointeur *rval* pointe sur une structure *jsval*. La valeur renvoyée par la fonction doit être transmise à *\*rval*.

La fonction renvoie la valeur *JS\_TRUE* si elle s'exécute correctement, et *JS\_FALSE* dans le cas contraire. Si la fonction renvoie *JS\_FALSE*, l'exécution du script est interrompue et un message d'erreur s'affiche.

## JSBool JS\_DefineFunction()

### Utilisation

```
JSBool JS_DefineFunction(unsigned short *name, JSNative call, unsigned int  
nargs)
```

### Description

Cette méthode enregistre une fonction de niveau C auprès de l'interpréteur JavaScript de Flash. Lorsque la fonction *JS\_DefineFunction()* a enregistré la fonction de niveau C spécifiée dans l'argument *call*, vous pouvez appeler cette dernière dans un script en JavaScript à l'aide du nom indiqué dans l'argument *name*. L'argument *name* est sensible à la casse.

En règle générale, cette fonction est appelée par la fonction *MM\_Init()*, que Flash appelle au démarrage.

## Arguments

`unsigned short *name, JSNative call, unsigned int nargs`

- L'argument *name* est le nom de la fonction tel qu'il est exposé à JavaScript.
- L'argument *call* est un pointeur sur une fonction de niveau C. La fonction doit renvoyer une valeur `JSBool` qui indique son succès ou son échec.
- L'argument *nargs* indique le nombre d'arguments attendus par la fonction.

## Valeur renvoyée

Valeur booléenne : `JS_TRUE` indique le succès de l'opération, `JS_FALSE` indique un échec.

# unsigned short \*JS\_ValueToString()

## Utilisation

```
unsigned short *JS_ValueToString(JSContext *cx, jsval v,  
                                unsigned int *pLength)
```

## Description

Cette méthode extrait un argument de fonction d'une structure `jsval`, le convertit en chaîne si possible, et renvoie la valeur ainsi convertie à la routine appelante.

### REMARQUE

Ne modifiez pas le pointeur sur le tampon renvoyé, vous risqueriez d'altérer les structures de données de l'interpréteur JavaScript. Pour modifier la chaîne, copiez ses caractères dans un autre tampon et créez une nouvelle chaîne en JavaScript.

## Arguments

`JSContext *cx, jsval v, unsigned int *pLength`

- L'argument *cx* est le pointeur opaque sur `JSContext` qui est passé à la fonction JavaScript.
- L'argument *v* est la structure `jsval` dont la chaîne doit être extraite.
- L'argument *pLength* est un pointeur sur un entier non signé. Cette fonction met la longueur de la chaîne (en octets) dans *\*pLength*.

## Valeur renvoyée

Un pointeur sur une chaîne terminée par un octet nul en cas de succès, ou la valeur `null` en cas d'échec. La routine appelante ne doit pas libérer cette chaîne à la fin de son exécution.

## JSBool JS\_ValueToInteger()

### Utilisation

```
JSBool JS_ValueToInteger(JSContext *cx, jsval v, long *lp);
```

### Description

Cette méthode extrait un argument de fonction d'une structure `jsval`, le convertit en entier si possible, et renvoie la valeur ainsi convertie à la routine appelante.

### Arguments

`JSContext *cx`, `jsval v`, `long *lp`

- L'argument `cx` est le pointeur opaque sur `JSContext` qui est passé à la fonction JavaScript.
- L'argument `v` est la structure `jsval` dont l'entier doit être extrait.
- L'argument `lp` est un pointeur sur un entier de 4 octets. Cette fonction enregistre la valeur convertie dans `*lp`.

### Valeur renvoyée

Valeur booléenne : `JS_TRUE` indique le succès de l'opération, `JS_FALSE` indique un échec.

## JSBool JS\_ValueToInteger()

### Utilisation

```
JSBool JS_ValueToDouble(JSContext *cx, jsval v, double *dp);
```

### Description

Cette méthode extrait un argument de fonction d'une structure `jsval`, le convertit en double si possible, et renvoie la valeur ainsi convertie à la routine appelante.

### Arguments

`JSContext *cx`, `jsval v`, `double *dp`

- L'argument `cx` est le pointeur opaque sur `JSContext` qui est passé à la fonction JavaScript.
- L'argument `v` est la structure `jsval` dont le double doit être extrait.
- L'argument `dp` est un pointeur sur un nombre double de 8 octets. Cette fonction enregistre la valeur convertie dans `*dp`.

### Valeur renvoyée

Valeur booléenne : `JS_TRUE` indique le succès de l'opération, `JS_FALSE` indique un échec.

## JSBool JS\_ValueToBoolean()

### Utilisation

```
JSBool JS_ValueToBoolean(JSContext *cx, jsval v, JSBool *bp);
```

### Description

Cette méthode extrait un argument de fonction d'une structure `jsval`, le convertit en booléen si possible, et renvoie la valeur ainsi convertie à la routine appelante.

### Arguments

`JSContext *cx`, `jsval v`, `JSBool *bp`

- L'argument `cx` est le pointeur opaque sur `JSContext` qui est passé à la fonction JavaScript.
- L'argument `v` est la structure `jsval` dont le booléen doit être extrait.
- L'argument `bp` est un pointeur sur une valeur booléenne `JSBool`. Cette fonction enregistre la valeur convertie dans `*bp`.

### Valeur renvoyée

Valeur booléenne : `JS_TRUE` indique le succès de l'opération, `JS_FALSE` indique un échec.

## JSBool JS\_ValueToObject()

### Utilisation

```
JSBool JS_ValueToObject(JSContext *cx, jsval v, JSObject **op);
```

### Description

Cette méthode extrait un argument de fonction d'une structure `jsval`, le convertit en objet si possible, et renvoie la valeur ainsi convertie à la routine appelante. Si l'objet est un tableau, le contenu de celui-ci peut être lu à l'aide de `JS_GetArrayLength()` et `JS_GetElement()`.

### Arguments

`JSContext *cx`, `jsval v`, `JSObject **op`

- L'argument `cx` est le pointeur opaque sur `JSContext` qui est passé à la fonction JavaScript.
- L'argument `v` est la structure `jsval` dont l'objet doit être extrait.
- L'argument `op` est un pointeur sur un pointeur `JSObject`. Cette fonction enregistre la valeur convertie dans `*op`.

### Valeur renvoyée

Valeur booléenne : `JS_TRUE` indique le succès de l'opération, `JS_FALSE` indique un échec.



## JSBool JS\_StringToValue()

### Utilisation

```
JSBool JS_StringToValue(JSContext *cx, unsigned short *bytes, uint sz,
                        jsval *vp);
```

### Description

Cette méthode enregistre la valeur de retour d'une chaîne dans une structure `jsval`. Elle alloue un nouvel objet chaîne JavaScript.

### Arguments

`JSContext *cx`, `unsigned short *bytes`, `size_t sz`, `jsval *vp`

- L'argument `cx` est le pointeur opaque sur `JSContext` qui est passé à la fonction JavaScript.
- L'argument `bytes` est la chaîne à enregistrer dans la structure `jsval`. Les données de la chaîne sont copiées. Il est donc conseillé de libérer la chaîne lorsqu'elle n'est plus nécessaire. Si la taille de la chaîne n'est pas spécifiée (voir l'argument `sz`), la chaîne doit être terminée par un octet de valeur nulle.
- L'argument `sz` est la taille de la chaîne, en octets. Si `sz` contient 0, la longueur de la chaîne terminée par un octet de valeur nulle est automatiquement calculée.
- L'argument `vp` est un pointeur sur la structure `jsval` dans laquelle le contenu de la chaîne doit être copié.

### Valeur renvoyée

Valeur booléenne : `JS_TRUE` indique le succès de l'opération, `JS_FALSE` indique un échec.

## JSBool JS\_DoubleToValue()

### Utilisation

```
JSBool JS_DoubleToValue(JSContext *cx, double dv, jsval *vp);
```

### Description

Cette méthode enregistre la valeur de retour d'un nombre en virgule flottante dans une structure `jsval`.

## Arguments

`JSContext *cx, double dv, jsval *vp`

- L'argument `cx` est le pointeur opaque sur `JSContext` qui est passé à la fonction JavaScript.
- L'argument `dv` est un nombre en virgule flottante.
- L'argument `vp` est un pointeur sur la structure `jsval` dans laquelle le contenu du double doit être copié.

## Valeur renvoyée

Valeur booléenne : `JS_TRUE` indique le succès de l'opération, `JS_FALSE` indique un échec.

# JSVal JS\_BooleanToValue()

## Utilisation

```
jsval JS_BooleanToValue(JSBool bv);
```

## Description

Cette méthode enregistre la valeur de retour d'un booléen dans une structure `jsval`.

## Arguments

`JSBool bv`

- L'argument `bv` est une valeur booléenne : `JS_TRUE` indique le succès de l'opération, `JS_FALSE` indique un échec.

## Valeur renvoyée

Une structure `JSVal` qui contient la valeur booléenne passée en argument à la fonction.

# JSVal JS\_BytesToValue()

## Utilisation

```
JSBool JS_BytesToValue(JSContext *cx, unsigned short *bytes, uint sz, jsval *vp);
```

## Description

Cette méthode convertit des octets en valeur JavaScript.

## Arguments

`JSContext *cx`, unsigned short *bytes*, uint *sz*, `jsval *vp`

- L'argument *cx* est le contexte JavaScript.
- L'argument *bytes* est la chaîne d'octets à convertir en objet JavaScript.
- L'argument *sz* est le nombre d'octets à convertir.
- L'argument *vp* est la valeur JavaScript.

## Valeur renvoyée

Valeur booléenne : `JS_TRUE` indique le succès de l'opération, `JS_FALSE` indique un échec.

# JSVal JS\_IntegerToValue()

## Utilisation

```
jsval JS_IntegerToValue(long lv);
```

## Description

Cette méthode convertit un entier long en structure `JSVal`.

## Arguments

*lv*

- L'argument *lv* est la valeur de l'entier long à convertir en structure `jsval`.

## Valeur renvoyée

Une structure `JSVal` qui contient l'entier passé en argument à la fonction.

# JSVal JS\_ObjectToValue()

## Utilisation

```
jsval JS_ObjectToValue(JSObject *obj);
```

## Description

Cette méthode enregistre la valeur de retour d'un objet dans une structure `JSVal`. Utilisez `JS_NewArrayObject()` pour créer un objet tableau, et `JS_SetElement()` pour définir son contenu.

## Arguments

`JSObject *obj`

- L'argument *obj* est un pointeur sur l'objet `JSObject` à convertir en structure `JSVal`.

## Valeur renvoyée

Une structure `JSVa1` qui contient l'objet passé en argument à la fonction.

# unsigned short \*JS\_ObjectType()

## Utilisation

```
unsigned short *JS_ObjectType(JSObject *obj);
```

## Description

Cette méthode reçoit une référence à un objet et renvoie le nom de la classe de cet objet. Par exemple, si l'objet est un objet DOM, la fonction renvoie `Document`. Si l'objet est un nœud du document, la fonction renvoie `Element`. Pour un objet tableau, la fonction renvoie `"Array"`.

### REMARQUE

Ne modifiez pas le pointeur sur le tampon renvoyé, vous risqueriez d'altérer les structures de données de l'interpréteur JavaScript.

## Arguments

`JSObject *obj`

- En général, cet argument est transmis et converti à l'aide de la fonction `JS_ValueToObject()`.

## Valeur renvoyée

Un pointeur sur une chaîne terminée par un octet nul. La routine appelante ne doit pas libérer cette chaîne à la fin de son exécution.

# JSObject \*JS\_NewArrayObject()

## Utilisation

```
JSObject *JS_NewArrayObject(JSContext *cx, unsigned int length [, jsval  
*v])
```

## Description

Cette méthode crée un nouvel objet qui contient un tableau de `JSVa1`.

## Arguments

`JSText *cx, unsigned int length, jsval *v`

- L'argument `cx` est le pointeur opaque sur `JSText` qui est passé à la fonction JavaScript.
- L'argument `length` est le nombre maximal d'éléments du tableau.
- L'argument `v` est un pointeur facultatif sur les structures `jsval` à placer dans le tableau.  
Si la valeur renvoyée n'est pas `null`, `v` est un tableau qui contient des éléments `length`.  
Si la valeur renvoyée est `null`, le contenu initial de l'objet tableau est non défini (`undefined`). Il peut être défini à l'aide de la fonction `JS_SetElement()`.

## Valeur renvoyée

Un pointeur sur un nouvel objet tableau, ou la valeur `null` en cas d'échec.

# long JS\_GetArrayLength()

## Utilisation

`long JS_GetArrayLength(JSText *cx, JSObject *obj)`

## Description

Cette méthode reçoit un pointeur sur un objet tableau et extrait le nombre d'éléments de ce tableau.

## Arguments

`JSText *cx, JSObject *obj`

- L'argument `cx` est le pointeur opaque sur `JSText` qui est passé à la fonction JavaScript.
- L'argument `obj` est un pointeur sur un objet tableau.

## Valeur renvoyée

Le nombre d'éléments du tableau, ou -1 en cas d'échec.

# JSBool JS\_GetElement()

## Utilisation

`JSBool JS_GetElement(JSText *cx, JSObject *obj, jsint idx, jsval *vp)`

## Description

Cette méthode lit un élément dans un objet tableau.

## Arguments

`JSContext *cx, JSObject *obj, jsint idx, jsval *vp`

- L'argument `cx` est le pointeur opaque sur `JSContext` qui est passé à la fonction JavaScript.
- L'argument `obj` est un pointeur sur un objet tableau.
- L'argument `idx` est un index d'entier dans le tableau : le premier élément est l'index 0 et le dernier, l'index (`length - 1`).
- L'argument `vp` est un pointeur sur une structure `jsval` dans laquelle le contenu de la structure `jsval` du tableau doit être copié.

## Valeur renvoyée

Valeur booléenne : `JS_TRUE` indique le succès de l'opération, `JS_FALSE` indique un échec.

# JSBool JS\_SetElement()

## Utilisation

`JSBool JS_SetElement(JSContext *cx, JSObject *obj, jsint idx, jsval *vp)`

## Description

Cette méthode écrit un élément dans un objet tableau.

## Arguments

`JSContext *cx, JSObject *obj, jsint idx, jsval *vp`

- L'argument `cx` est le pointeur opaque sur `JSContext` qui est passé à la fonction JavaScript.
- L'argument `obj` est un pointeur sur un objet tableau.
- L'argument `idx` est un index d'entier dans le tableau : le premier élément est l'index 0 et le dernier, l'index (`length - 1`).
- L'argument `vp` est un pointeur sur une structure `jsval` dont le contenu doit être copié dans la structure `jsval` du tableau.

## Valeur renvoyée

Valeur booléenne : `JS_TRUE` indique le succès de l'opération, `JS_FALSE` indique un échec.

# JSBool JS\_ExecuteScript()

## Utilisation

```
JS_ExecuteScript (JSContext *cx, JSObject *obj, unsigned short *script,  
                 unsigned int sz, jsval *rval)
```

## Description

Cette méthode compile et exécute une chaîne en JavaScript. Si le script génère une valeur de retour, celle-ci est renvoyée dans *\*rval*.

## Arguments

```
JSContext *cx, JSObject *obj, unsigned short *script, unsigned int sz, jsval  
*rval
```

- L'argument *cx* est le pointeur opaque sur *JSContext* qui est passé à la fonction JavaScript.
- L'argument *obj* pointe sur l'objet dans le contexte duquel le script s'exécute. Pendant l'exécution du script, le mot-clé *this* est égal à cet objet. En général, ce pointeur est le pointeur *JSObject* passé à la fonction JavaScript.
- L'argument *script* est une chaîne qui contient du code JavaScript. Si la taille de la chaîne n'est pas spécifiée (voir l'argument *sz*), la chaîne doit être terminée par un octet de valeur nulle.
- L'argument *sz* est la taille de la chaîne, en octets. Si la valeur de *sz* est 0, la longueur de la chaîne terminée par un octet de valeur nulle est automatiquement calculée.
- L'argument *rval* pointe sur une structure *jsval*. La valeur renvoyée par la fonction est stockée dans *\*rval*.

## Valeur renvoyée

Valeur booléenne : *JS\_TRUE* indique le succès de l'opération, *JS\_FALSE* indique un échec.

