macromedia®

FLASH® 8

Guide de référence du langage ActionScript Flash Lite 1.x

Marques de commerce

1 Step RoboPDF, ActiveEdit, ActiveTest, Authorware, Blue Sky Software, Blue Sky, Breeze, Breezo, Captivate, Central, ColdFusion, Contribute, Database Explorer, Director, Dreamweaver, Fireworks, Flash, FlashCast, FlashHelp, Flash Lite, FlashPaper, Flash Video Endocer, Flex, Flex Builder, Fontographer, FreeHand, Generator, HomeSite, JRun, MacRecorder, Macromedia, MXML, RoboEngine, RoboHelp, RoboInfo, RoboPDF, Roundtrip, Roundtrip HTML, Shockwave, SoundEdit, Studio MX, UltraDev et WebHelp sont des marques de commerce ou des marques déposées de Macromedia, Inc. qui peuvent être déposées aux Etats-Unis et/ou dans d'autres juridictions ou pays. Les autres noms de produits, logos, graphiques, mises en page, titres, mots ou expressions mentionnés dans cette publication peuvent être des marques de commerce, des marques de service ou des noms de marque appartenant à Macromedia, Inc. ou à d'autres entités et peuvent être déposés dans certaines juridictions ou pays.

Informations au sujet des parties tierces

Ce guide contient des liens vers des sites Web qui ne sont pas sous le contrôle de Macromedia, qui n'est donc aucunement responsable de leur contenu. Si vous accédez à un site Web tiers mentionné dans ce guide, vous le faites à vos propres risques. Macromedia fournit ces liens à des fins pratiques et l'inclusion de ces liens n'implique pas que Macromedia parraine ou accepte la moindre responsabilité pour le contenu de ces sites Web tiers.

Technologie de compression et décompression audio discours utilisée sous licence de Nellymoser, Inc. (www.nellymoser.com).

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

Navigateur Opera ® Copyright © 1995-2002 Opera Software ASA et ses fournisseurs. Tous droits réservés.

La vidéo de Macromedia Flash 8 est optimisée par la technologie vidéo On2 TrueMotion. © 1992-2005 On2 Technologies, Inc. Tous droits réservés. http://www.on2.com.

Visual SourceSafe est une marque commerciale ou une marque déposée de Microsoft Corporation aux Etats-Unis et/ou dans d'autres pays.

Copyright © 2005 Macromedia, Inc. Tous droits réservés. La copie, photocopie, reproduction, traduction ou conversion de ce manuel, sous quelque forme que ce soit, mécanique ou électronique, est interdite sans une autorisation préalable écrite de Macromedia, Inc. Le propriétaire ou l'utilisateur autorisé d'une copie valide du logiciel qu'accompagne ce manuel peut néanmoins imprimer une copie de ce dernier à partir d'une version électronique afin d'apprendre à utiliser le logiciel, à condition qu'aucune partie de ce manuel ne soit imprimée, reproduite, distribuée, revendue ou cédée dans un autre but, y compris, mais sans limitation aucune, dans un but commercial, tel que la revente de copies de cette documentation ou l'offre de services de maintenance rémunérés.

Remerciements

Gestion du projet : Mary Leigh Burke

Rédaction : Guy Haas, Denise Green, Mike Krisher

Rédactrice en chef : Rosana Francescato

Révision : Linda Adler, Geta Carlson, Evelyn Eldridge

Gestion de la production : Patrice O'Neill, Kristin Conradi, Yuko Yagi

Conception et production : Adam Barnett, Aaron Begley, Paul Benkman. John Francis, Geeta Karmarkar, Masayo Noda, Paul Rangel, Arena Reed, Mario Reynoso

Remerciements particuliers à Lisa Friendly, Bonnie Loo, Erick Vera, Florian de Joannès, les testeurs bêta ainsi qu'à toute l'équipe d'ingénierie et d'assurance qualité Flash Lite.

Première édition : Septembre 2005

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

Table des matières

Introduction 7

Exemple d'entrée utilisée pour la plupart des éléments ActionScript 7
Dossier Exemples 8
Conventions typographiques 9

Chapitre 1 : Fonctions globales de Flash Lite 11

call() 13
chr() 14
duplicateMovieClip() 15
eval () 16
getProperty() 17
getTimer() 18
getURL() 18
gotoAndPlay() 21
gotoAndStop() 22
ifFrameLoaded() 23
int() 24
length() 25
loadMovie() 25
loadMovieNum() 27
loadVariables() 28
loadVariablesNum() 30
mbchr() 31
mblength() 31
mbord() 32
mbsubstring() 33
nextFrame() 34
nextScene() 34
Number() 35
on() 36
ord() 37
play() 37
prevFrame() 38
prevScene() 39
random() 39

removeMovieClip() 40
set() 41
setProperty() 42
stop() 42
stopAllSounds() 43
String() 44
substring() 45
tellTarget() 45
toggleHighQuality() 46
trace() 47
unloadMovie() 48
unloadMovieNum() 49

Chapitre 2 : Propriétés de Flash Lite 51

/ (barre de fraction) 52
_alpha 53
_currentframe 54
_focusrect 54
_framesloaded 55
_height 55
_highquality 56
_level 57
maxscroll 58
_name 58
_rotation 59
scroll 60
_target 60
_totalframes 61
_visible 61
_width 62
_x 62
_xscale 63
_y 64
_yscale 65

Chapitre 3 : Instructions de Flash Lite 67

break 68
case 69
continue 70
do..while 72
else 73
else if 74
for 75
if 76

switch 77
while 78

Chapitre 4 : Opérateurs de Flash Lite 81

add (concaténation de chaîne) 84
+= (affectation d'addition) 85
and 86
= (affectation) 87
/* (bloc de commentaires) 87
, (virgule) 88
// (commentaire) 89
?: (conditionnel) 90
-- (décrément) 91
/ (division) 92
/= (affectation de division) 92
. (point) 93
++ (incrément) 94
&& (AND logique) 95
! (NOT logique) 96
|| (OR logique) 97
% (modulo) 98
%= (affectation modulo) 98
*= (affectation de multiplication) 99
* (produit) 100
+ (addition numérique) 101
== (égalité numérique) 101
> (numérique supérieur à) 102
>= (numérique supérieur ou égal à) 103
<> (inégalité numérique) 103
< (numérique inférieur à) 104
<= (numérique inférieur ou égal à) 105
() (parenthèses) 105
" " (séparateur de chaîne) 106
eq (égalité de chaîne) 107
gt (chaîne supérieure à) 108
ge (chaîne supérieure ou égale à) 109
ne (inégalité de chaîne) 110
lt (chaîne inférieure à) 110
le (chaîne inférieure ou égale à) 111
- (soustraction) 112
-= (affectation de soustraction) 113

Chapitre 5 : Eléments de langage spécifiques de Flash Lite . . . 115
Fonctionnalités 118
fscommand() 126
fscommand2() 128

Index 163

Introduction

Ce manuel décrit la syntaxe et le rôle des éléments ActionScript utilisés pour développer des applications pour Flash Lite 1.0 et Flash Lite 1.1, communément appelées Flash Lite 1.x. ActionScript de Flash Lite 1.x est basé sur la version d'ActionScript utilisée dans Flash 4. Pour utiliser les exemples de ce manuel dans un script, copiez le code et collez-le dans la fenêtre de script ou dans un fichier de script externe. Ce manuel répertorie l'ensemble des éléments ActionScript : opérateurs, mots-clés, instructions, commandes, propriétés, fonctions, classes et méthodes.

Exemple d'entrée utilisée pour la plupart des éléments ActionScript

L'exemple d'entrée suivant illustre les conventions utilisées pour la plupart des éléments ActionScript.

Titre de l'entrée

Les entrées sont classées par ordre alphabétique dans un chapitre. L'ordre alphabétique ne tient pas compte des majuscules, des traits de soulignement de préfixe, et ainsi de suite.

Disponibilité

Sauf mention contraire, la section Disponibilité indique les versions de Flash Lite qui prennent en charge l'élément.

Utilisation

Cette section présente la syntaxe correcte d'utilisation des éléments ActionScript dans votre code. La partie obligatoire de la syntaxe est indiquée en `police de code`. Le code que vous devez fournir ainsi que les informations de type de données sont en *`italique`*. Les deux-points (:) qui précèdent toujours les informations de type de données permettent de différencier les types de données du code que vous devez fournir. Les crochets (`[]`) indiquent des paramètres facultatifs.

Opérandes

Cette section décrit les paramètres répertoriés dans la syntaxe.

Description

Cette section identifie le type d'élément (par exemple, opérateur, fonction ou autre) ainsi que les valeurs renvoyées par l'élément (le cas échéant), et décrit son utilisation.

Exemple

Cette section fournit un exemple de code illustrant l'utilisation de l'élément.

Voir aussi

Cette section répertorie les entrées connexes du dictionnaire ActionScript.

Dossier Exemples

Un ensemble de fichiers d'exemples se trouve dans le répertoire /Samples and Tutorials/Samples/FlashLite/ du répertoire d'installation de Flash 8.

Les chemins d'accès types à ce dossier sont les suivants :

- Windows : /Program Files/Macromedia/Flash 8/Samples et Tutorials/Samples/FlashLite/
- Macintosh : DD/Applications/Macromedia/Flash 8/Samples et Tutorials/Samples/FlashLite/

Le dossier FlashLite contient un ensemble de fichiers FLA correspondant à des projets Flash Lite complets contenant du code ActionScript.

Conventions typographiques

Ce manuel utilise les conventions typographiques suivantes :

- *La police en italique* indique une valeur qui devrait être remplacée (par exemple, dans le chemin d'un dossier).
- `La police de code` indique le code ActionScript.
- *`La police de code en italique`* identifie les paramètres ActionScript.
- **La police en gras** indique une entrée que vous devez saisir exactement à l'identique.
- Les guillemets doubles (`" "`) figurant dans les exemples de code indiquent des chaînes délimitées. Les programmeurs peuvent néanmoins également utiliser des guillemets simples.

CHAPITRE 1

Fonctions globales de Flash Lite

1

Cette section décrit la syntaxe et le rôle des fonctions globales ActionScript de Macromedia Flash Lite 1.1. Elle inclut les fonctions suivantes.

Fonction	Description
`call()`	Exécute le script dans l'image appelée sans positionner la tête de lecture sur celle-ci.
`chr()`	Convertit les numéros de code ASCII en caractères.
`duplicateMovieClip()`	Crée une occurrence de clip pendant la lecture du fichier SWF.
`eval ()`	Accède aux variables, propriétés, objets ou clips en fonction de leur nom.
`getProperty()`	Renvoie la valeur de la propriété spécifiée pour le clip donné.
`getTimer()`	Renvoie le nombre de millisecondes qui se sont écoulées depuis le début de la lecture du fichier SWF.
`getURL()`	Charge un document en provenance d'une URL spécifique dans une fenêtre ou transmet des variables à une autre application sur une URL définie.
`gotoAndPlay()`	Place la tête de lecture sur l'image spécifiée dans une séquence et commence la lecture à partir de cette image. Si aucune séquence n'est spécifiée, la tête de lecture passe à l'image spécifiée de la séquence en cours.
`gotoAndStop()`	Place la tête de lecture sur l'image spécifiée dans une séquence et l'arrête à ce niveau. Si aucune séquence n'est spécifiée, la tête de lecture passe à l'image de la séquence en cours.
`ifFrameLoaded()`	Vérifie si le contenu d'une image spécifique est disponible localement.
`int()`	Tronque un nombre décimal en une valeur entière.
`length()`	Renvoie le nombre de caractères du nom de chaîne ou variable spécifié.

Fonction	Description
loadMovie()	Charge un fichier SWF dans Flash Lite pendant la lecture du fichier SWF d'origine.
loadMovieNum()	Charge un fichier SWF dans l'un des niveaux de Flash Lite pendant la lecture du fichier SWF.
loadVariables()	Lit les données dans un fichier externe, tel qu'un fichier texte ou du texte généré par un script ColdFusion, CGI, des pages ASP CGI, PHP ou un script Perl et définit les valeurs pour les variables dans un niveau de Flash Lite. Cette fonction permet également de mettre à jour les variables du fichier SWF actif en fonction des nouvelles valeurs.
loadVariablesNum()	Lit les données dans un fichier externe, tel qu'un fichier texte ou du texte généré par un script ColdFusion, CGI, des pages ASP (Active Server Pages), PHP ou un script Perl et définit les valeurs pour les variables dans un niveau de Flash Lite. Cette fonction permet également de mettre à jour les variables du fichier SWF actif en fonction des nouvelles valeurs.
mbchr()	Convertit un numéro de code ASCII en caractère multi-octets.
mblength()	Renvoie la longueur de la chaîne de caractères multi-octets.
mbord()	Convertit le caractère spécifié en nombre multi-octets.
mbsubstring()	Extrait une nouvelle chaîne de caractères multi-octets d'une chaîne de caractères multi-octets.
nextFrame()	Place la tête de lecture sur l'image suivante et l'arrête.
nextScene()	Place la tête de lecture sur l'image 1 de la séquence suivante et l'arrête.
Number()	Convertit une expression en nombre et renvoie une valeur.
on()	Spécifie l'événement de type utilisateur ou pression de touche devant déclencher un événement.
ord()	Convertit les caractères en numéros de code ASCII.
play()	Fait avancer la tête de lecture au sein du scénario.
prevFrame()	Place la tête de lecture sur l'image précédente et l'arrête. Si l'image active est l'image 1, la tête de lecture ne bouge pas.
prevScene()	Place la tête de lecture sur l'image 1 de la séquence précédente et l'arrête.
removeMovieClip()	Supprime le clip spécifié, initialement créé à l'aide de la méthode duplicateMovieClip().
set()	Associe une valeur à une variable.

Fonction	Description
setProperty()	Modifie la valeur des propriétés d'un clip pendant la lecture de ce dernier.
stop()	Arrête le fichier SWF en cours de lecture.
stopAllSounds()	Arrête tous les sons en cours de diffusion à partir d'un fichier SWF, sans arrêter la tête de lecture.
String()	Renvoie une chaîne représentant le paramètre spécifié.
substring()	Extrait une partie d'une chaîne.
tellTarget()	Applique les instructions spécifiées dans le paramètre *statement(s)* au scénario spécifié dans le paramètre *target*.
toggleHighQuality()	Active et désactive l'anti-aliasing dans Flash Lite. L'anti-aliasing adoucit les bords des objets mais ralentit la lecture du fichier SWF.
trace()	Evalue l'expression et affiche le résultat dans le panneau Sortie en mode test.
unloadMovie()	Supprime le clip qui a été chargé par l'intermédiaire de loadMovie(), loadMovieNum() ou duplicateMovieClip() de Flash Lite.
unloadMovieNum()	Supprime le clip qui a été chargé par l'intermédiaire de loadMovie(), loadMovieNum() ou duplicateMovieClip() d'un niveau de Flash Lite.

call()

Disponibilité

Flash Lite 1.0.

Utilisation

```
call(frame)

call(movieClipInstance:frame)
```

Opérandes

frame L'étiquette ou le numéro d'une image dans le scénario.

movieClipInstance Le nom d'occurrence d'un clip.

Description

Fonction ; exécute le script dans l'image appelée sans positionner la tête de lecture sur celle-ci. Les variables locales n'existent pas après l'exécution du script. La fonction `call()` peut avoir deux formes possibles :

- La forme par défaut exécute le script sur l'image spécifiée dans le même scénario que celui dans lequel a été exécutée la fonction `call()` sans positionner la tête de lecture sur celle-ci.
- La forme d'occurrence de clip spécifiée exécute le script sur l'image spécifiée de l'occurrence de clip sans positionner la tête de lecture sur celle-ci.

REMARQUE	La fonction `call()` fonctionne de façon synchronisée ; tout code ActionScript suivant une fonction `call()` ne s'exécute que lorsque l'intégralité du code ActionScript est terminée au niveau de l'image spécifiée.

Exemple

Les exemples suivants exécutent le script dans l'image `myScript` :

```
// exécution des fonctions de l'image ayant l'étiquette "myScript"
thisFrame = "myScript";
trace ("Calling the script in frame: " add thisFrame);

// exécution des fonctions d'une autre image dans le même scénario
call("myScript");
```

chr()

Disponibilité

Flash Lite 1.0.

Utilisation

`chr(`*`number`*`)`

Opérandes

`number` Un numéro de code ASCII.

Description

Fonction de la chaîne ; convertit les numéros de code ASCII en caractères.

Exemple

L'exemple suivant convertit le nombre 65 en lettre *A* et l'affecte à la variable `myVar` :

```
myVar = chr(65);
trace (myVar);// Sortie : A
```

duplicateMovieClip()

Disponibilité

Flash Lite 1.0.

Utilisation

```
duplicateMovieClip(target, newname, depth)
```

Opérandes

target Le chemin cible du clip à dupliquer.

newname Un identificateur unique pour le clip dupliqué.

depth Un niveau de profondeur unique pour le clip dupliqué. Le niveau de profondeur indique l'ordre d'empilement des clips dupliqués. Cet ordre d'empilement ressemble à l'ordre d'empilement des calques dans le scénario ; les clips dont le niveau de profondeur est inférieur sont masqués par les clips de niveau supérieur. Vous devez associer un niveau de profondeur unique à chaque clip dupliqué pour ne pas risquer d'écraser des clips existants figurant à des profondeurs déjà occupées.

Description

Fonction ; crée une occurrence de clip pendant la lecture du fichier SWF et renvoie une référence au clip dupliqué. La tête de lecture des clips dupliqués commence toujours à l'image 1, quelle que soit la position de la tête de lecture dans le clip d'origine (parent). Les variables du clip parent ne sont pas copiées dans le clip dupliqué. Si le clip parent est supprimé, le clip dupliqué l'est également. La fonction ou la méthode `removeMovieClip()` permet de supprimer une occurrence de clip créée avec `duplicateMovieClip()`.

Exemple

L'exemple suivant duplique un clip appelé `originalClip` pour créer un nouveau clip appelé `newClip` à un niveau de profondeur de 10. La position *x* du nouveau clip est définie sur 100 pixels.

```
duplicateMovieClip("originalClip", "newClip", 10);
setProperty("newClip", _x, 100);
```

L'exemple suivant utilise la méthode `duplicateMovieClip()` dans une boucle `for` pour créer plusieurs nouveaux clips simultanément. Une variable d'indice permet de conserver une trace de la profondeur d'empilement maximale occupée. Tous les noms des clips dupliqués contiennent un suffixe numérique correspondant à leur profondeur d'empilement (`clip1`, `clip2`, `clip3`).

```
for (i = 1; i <= 3; i++) {
  newName = "clip" add i;
  duplicateMovieClip("originalClip", newName); }
```

Voir aussi

`removeMovieClip()`

eval ()

Disponibilité

Flash Lite 1.0.

Utilisation

`eval(`*`expression`*`)`

Opérandes

`expression` Une chaîne contenant le nom d'une variable, d'une propriété, d'un objet ou d'un clip à extraire.

Description

Fonction ; accède aux variables, propriétés, objets ou clips en fonction de leur nom. Si *`expression`* est une variable ou une propriété, la fonction renvoie la valeur de cette variable ou de cette propriété. Si *`expression`* est un objet ou un clip, la fonction renvoie une référence de l'objet ou du clip. Si l'élément nommé dans *`expression`* est introuvable, la fonction renvoie `undefined` (non défini).

Vous pouvez utiliser la fonction `eval()` pour simuler des tableaux, ou définir et récupérer dynamiquement la valeur d'une variable.

Exemple

L'exemple suivant utilise `eval()` pour déterminer la valeur de l'expression `"piece" + x`. Le résultat étant un nom de variable, `piece3`, `eval()` renvoie la valeur de la variable et l'associe à `y` :

```
piece3 = "dangerous";
x = 3;
y = eval("piece" add x);
trace(y);// Sortie : dangerous.
```

L'exemple suivant décrit la simulation d'un tableau :

```
name1 = "mike";
name2 = "debbie";
name3 = "logan";
for(i = 1; i <= 3; i++) {
  trace (eval("name" add i));/// Sortie : mike, debbie, logan
}
```

getProperty()

Disponibilité

Flash Lite 1.0.

Utilisation

`getProperty(`*`my_mc`*`, `*`property`*`)`

Opérandes

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

`property` Une propriété d'un clip.

Description

Fonction ; renvoie la valeur de la propriété spécifiée pour le clip *`person_mc`*.

Exemple

L'exemple suivant récupère la coordonnée de l'axe horizontal (`_x`) pour le clip `my_mc` dans le scénario du clip racine :

```
xPos = getProperty("person_mc", _x);
trace (xPos);   // sortie : -75
```

Voir aussi

`setProperty()`

getTimer()

Disponibilité

Flash Lite 1.0.

Utilisation

```
getTimer()
```

Opérandes

Aucun.

Description

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

Exemple

L'exemple suivant définit la variable `timeElapsed` sur le nombre de millisecondes qui se sont écoulées depuis le début de la lecture du fichier SWF :

```
timeElapsed = getTimer();
trace (timeElapsed);// Sortie : nombre de millisecondes durant lesquelles
  le clip a été lu
```

getURL()

Disponibilité

Flash Lite 1.0.

Utilisation

```
getURL(url [ , window [, "variables"]])
```

Opérandes

`url` L'URL permettant d'obtenir le document.

`window` Un paramètre facultatif spécifiant la fenêtre ou le cadre HTML dans lequel le document doit se charger.

REMARQUE	Le paramètre *`window`* n'est pas spécifié pour les applications Flash Lite, car les navigateurs des téléphones cellulaires ne prennent pas en charge plusieurs fenêtres.

Vous pouvez entrer une chaîne vide ou le nom d'une fenêtre spécifique, ou le sélectionner à partir des noms cibles réservés suivants :

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

`variables` Une méthode `GET` ou `POST` permettant d'envoyer des variables. En l'absence de variables, omettez ce paramètre. La méthode `GET` ajoute les variables à la fin de l'URL et est utilisée lorsque les variables sont peu nombreuses. La méthode `POST` place les variables dans un en-tête HTTP distinct et permet d'envoyer des variables longues de type chaîne.

Description

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

Flash Lite 1.0 reconnaît uniquement les protocoles HTTP, HTTPS, mailto et tel. Outre ces protocoles, Flash Lite 1.1 reconnaît également les protocoles de fichier ainsi que les protocoles SMS et MMS.

Flash Lite transmet l'appel au système d'exploitation qui le gère à l'aide de l'application par défaut enregistrée pour le protocole spécifié.

Une seule fonction `getURL()` est traitée par image ou par gestionnaire d'événements.

Certains combinés limitent la fonction `getURL()` aux événements de touche uniquement, auquel cas l'appel `getURL()` n'est traité que s'il est déclenché dans un gestionnaire d'événements de touche. Même dans ces conditions, une seule fonction `getURL()` est traitée par gestionnaire d'événements.

Exemple

Dans le code ActionScript suivant, le lecteur Flash Lite ouvre la page mobile.macromedia.com dans le navigateur par défaut :

```
myURL = "http://mobile.macromedia.com";
 on(keyPress "1") {
  getURL(myURL);
}
```

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

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

Le code ActionScript suivant utilise la méthode `POST` pour placer les variables dans un en-tête HTTP :

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

Vous pouvez affecter une fonction de bouton permettant d'ouvrir une fenêtre de composition de message dont les champs de texte `address`, `subject` et `body` seraient déjà remplis. Pour affecter une fonction de bouton, utilisez l'une des méthodes suivantes : pour Shift-JIS ou le codage de caractères anglais, utilisez la méthode 1 ; la méthode 2 ne prend en charge que le codage de caractères anglais.

Méthode 1 : définissez des variables pour chacun des paramètres voulus, comme dans l'exemple suivant :

```
on (release, keyPress "#"){
subject = "email subject";
body = "email body";
getURL("mailto:somebody@anywhere.com", "", "GET");
}
```

Méthode 2 : définissez chaque paramètre dans la fonction `getURL()`, comme dans l'exemple suivant :

```
on (release, keyPress "#"){
getURL("mailto:somebody@anywhere.com?cc=cc@anywhere.com&bcc=bcc@anywhere.
com&subject=I am the email subject&body=I am the email body");
}
```

La méthode 1 renvoie automatiquement un codage URL tandis que la méthode 2 préserve les espaces dans les chaînes. Par exemple, les chaînes résultant de l'utilisation de la méthode 1 ont l'aspect suivant :

```
email+subject
email+body
```

Par opposition, la méthode 2 renvoie les chaînes suivantes :

```
email subject
email body
```

L'exemple suivant utilise le protocole tel :

```
on (release, keyPress "#"){
  getURL("tel:117");
}
```

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

```
mySMS = "sms:4156095555?body=sample sms message";
on(keyPress "5") {
  getURL(mySMS);
}
```

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

```
// exemple mms
myMMS = "mms:4156095555?body=sample mms message";
on(keyPress "6") {
  getURL(myMMS);
}
```

Dans l'exemple suivant, la fonction `getURL()` est utilisée pour ouvrir un fichier texte stocké dans le système de fichiers local :

```
// exemple de protocole de fichier
filePath = "file://c:/documents/flash/myApp/myvariables.txt";
on(keyPress "7") {
  getURL(filePath);
}
```

gotoAndPlay()

Disponibilité

Flash Lite 1.0.

Utilisation

```
gotoAndPlay([scene,] frame)
```

Opérandes

scene Une chaîne facultative qui spécifie le nom de la séquence cible de la tête de lecture.

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

Description

Fonction ; place la tête de lecture sur l'image spécifiée dans une séquence et commence la lecture à partir de cette image. Si aucune séquence n'est spécifiée, la tête de lecture passe à l'image spécifiée de la séquence en cours.

Le paramètre *scene* est réservé au scénario racine. Vous ne pouvez pas l'utiliser dans les scénarios des clips ou autres objets du document.

Exemple

Dans l'exemple suivant, lorsque l'utilisateur clique sur un bouton auquel est affectée la fonction `gotoAndPlay()`, la tête de lecture se positionne sur l'image 16 de la séquence active et commence la lecture du fichier SWF :

```
on(keyPress "7") {
  gotoAndPlay(16);
}
```

gotoAndStop()

Disponibilité

Flash 1.0.

Utilisation

`gotoAndStop([`*scene*`,] `*frame*`)`

Opérandes

scene Une chaîne facultative qui spécifie le nom de la séquence cible de la tête de lecture.

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

Description

Fonction ; place la tête de lecture sur l'image spécifiée sur une séquence et l'arrête à ce niveau. Si aucune séquence n'est spécifiée, la tête de lecture passe à l'image de la séquence en cours.

Le paramètre *scene* est réservé au scénario racine. Vous ne pouvez pas l'utiliser dans les scénarios des clips ou autres objets du document.

Exemple

Dans l'exemple suivant, lorsque l'utilisateur clique sur un bouton auquel est affectée la fonction `gotoAndStop()`, la tête de lecture se positionne sur l'image 5 de la séquence active et la lecture du fichier SWF est interrompue :

```
on(keyPress "8") {
  gotoAndStop(5);
}
```

ifFrameLoaded()

Disponibilité

Flash Lite 1.0.

Utilisation

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

Opérandes

`scene` Une chaîne facultative qui spécifie le nom de la séquence à charger.

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

`statement(s)` Les instructions à exécuter en cas de chargement de la séquence, ou de la séquence et de l'image, spécifiée(s).

Description

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

Exemple

L'exemple suivant utilise la fonction `ifFrameLoaded` pour vérifier si l'image 10 du fichier SWF est chargée. Si c'est le cas, la commande `trace()` imprime le texte « frame number 10 is loaded » dans le panneau Sortie. La variable de sortie est également définie avec le chargement d'une variable d'`image : 10`.

```
ifFrameLoaded(10) {
  trace ("frame number 10 is loaded");
  output = "frame loaded: 10";
 }
```

Voir aussi

`_framesloaded`

int()

Disponibilité

Flash Lite 1.0.

Utilisation

`int(`*`value`*`)`

Opérandes

`value` Un nombre ou une chaîne devant être tronqué en un entier.

Description

Fonction ; tronque un nombre décimal en valeur entière.

Exemple

L'exemple suivant tronque les nombres des variables `distance` et `myDistance` :

```
distance = 6.04 - 3.96;
//trace ("distance = " add distance add " and rounded to:" add
  int(distance));
// Sortie : distance = 2,08 et arrondie à : 2
myDistance = "3.8";
//trace ("myDistance = " add int(myDistance));
// Sortie : 3
```

length()

Disponibilité

Flash Lite 1.0.

Utilisation

```
length(expression)

length(variable)
```

Opérandes

expression Une chaîne.

variable Le nom d'une variable.

Description

Fonction de la chaîne ; renvoie le nombre de caractères du nom de chaîne ou variable spécifié.

Exemple

L'exemple suivant renvoie la longueur de la chaîne « Hello » :

```
length("Hello");
```

Le résultat est 5.

L'exemple suivant valide une adresse de messagerie en vérifiant qu'elle contienne au moins six caractères :

```
email = "someone@macromedia.com";
if (length(email) > 6) {
  //trace ("email appears to have enough characters to be valid");
}
```

loadMovie()

Disponibilité

Flash Lite 1.1.

Utilisation

```
loadMovie(url, target [, method])
```

Opérandes

url Une chaîne qui spécifie l'URL absolue ou relative du fichier SWF à charger. Un chemin relatif doit être relatif au fichier SWF au niveau 0. Les URL absolues doivent inclure la référence de protocole, telle que http:// ou file:///.

target Une référence à un clip ou une chaîne représentant le chemin d'un clip cible. Le clip cible est remplacé par le fichier SWF chargé.

method Un paramètre de chaîne facultatif spécifiant la méthode HTTP à utiliser pour l'envoi de variables. Ce paramètre doit correspondre à la chaîne `GET` ou `POST`. En l'absence de variable à envoyer, omettez ce paramètre. La méthode `GET` ajoute les variables à la fin de l'URL et est utilisée lorsque les variables sont peu nombreuses. La méthode `POST` place les variables dans un en-tête HTTP distinct et est utilisée pour des variables longues de type chaîne.

Description

Fonction ; charge un fichier SWF dans Flash Lite pendant la lecture du fichier SWF d'origine.

Pour charger un fichier SWF dans un niveau spécifique, utilisez la fonction `loadMovieNum()` plutôt que `loadMovie()`.

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

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

Exemple

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

```
loadMovie("circle.swf", "mySquare");
// Instruction équivalente : loadMovie("circle.swf", _level0.mySquare);
```

Voir aussi

`_level`, `loadMovieNum()`, `unloadMovie()`, `unloadMovieNum()`

loadMovieNum()

Disponibilité

Flash Lite 1.1.

Utilisation

`loadMovieNum(url, level [, method])`

Opérandes

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

`level` Un entier spécifiant le niveau de Flash Lite dans lequel le fichier SWF doit se charger.

`method` Un paramètre de chaîne facultatif spécifiant la méthode HTTP à utiliser pour l'envoi de variables. La valeur de ce paramètre doit être `GET` ou `POST`. En l'absence de variable à envoyer, omettez ce paramètre. La méthode `GET` ajoute les variables à la fin de l'URL et est utilisée lorsque les variables sont peu nombreuses. La méthode `POST` place les variables dans un en-tête HTTP distinct et est utilisée pour des variables longues de type chaîne.

Description

Fonction ; charge un fichier SWF dans l'un des niveaux de Flash Lite pendant la lecture du fichier SWF.

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

Pour spécifier une cible plutôt qu'un niveau, utilisez la fonction `loadMovie()` plutôt que `loadMovieNum()`.

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

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

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

Exemple

L'exemple suivant charge le fichier SWF dans le niveau 2 :

```
loadMovieNum("http://www.someserver.com/flash/circle.swf", 2);
```

Voir aussi

`_level`, `loadMovie()`, `unloadMovieNum()`

loadVariables()

Disponibilité

Flash Lite 1.1.

Utilisation

```
loadVariables(url, target [, variables])
```

Opérandes

`url` Une chaîne représentant une URL absolue ou relative par rapport à l'emplacement des variables. Si le fichier SWF effectuant cet appel s'exécute dans un navigateur Web, *`url`* doit appartenir au même domaine que le fichier SWF.

`target` Le chemin cible d'un clip devant recevoir les variables chargées.

`variables` Un paramètre de chaîne facultatif spécifiant la méthode HTTP à utiliser pour l'envoi de variables. Ce paramètre doit correspondre à la chaîne `GET` ou `POST`. En l'absence de variable à envoyer, omettez ce paramètre. La méthode `GET` ajoute les variables à la fin de l'URL et est utilisée lorsque les variables sont peu nombreuses. La méthode `POST` place les variables dans un en-tête HTTP distinct et est utilisée pour des variables longues de type chaîne.

Description

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

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

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

Pour charger des variables dans un niveau spécifique, utilisez la fonction `loadVariablesNum()` plutôt que `loadVariables()`.

Exemple

Les exemples suivants chargent des variables à partir d'un fichier texte et d'un serveur :

```
// chargement de variables à partir d'un fichier texte sur le système de
  fichiers local (Symbian Series 60)
on(release, keyPress "1") {
  filePath = "file://c:/documents/flash/myApp/myvariables.txt";
  loadVariables(filePath, _root);
}

// charger des variables (à partir d'un serveur) dans un clip
urlPath = "http://www.someserver.com/myvariables.txt";
loadVariables(urlPath, "myClip_mc");
```

Voir aussi

`loadMovieNum()`, `loadVariablesNum()`, `unloadMovie()`

loadVariablesNum()

Disponibilité

Flash Lite 1.1.

Utilisation

```
loadVariablesNum(url, level [, variables])
```

Opérandes

url Une chaîne représentant une URL absolue ou relative par rapport à l'emplacement des variables à charger. Si le fichier SWF effectuant cet appel s'exécute dans un navigateur Web, *url* doit appartenir au même domaine que le fichier SWF. Pour plus de détails, voir la section Description.

level Un entier spécifiant le niveau de Flash Lite devant recevoir les variables.

variables Un paramètre de chaîne facultatif spécifiant la méthode HTTP à utiliser pour l'envoi de variables. Ce paramètre doit correspondre à la chaîne `GET` ou `POST`. En l'absence de variable à envoyer, omettez ce paramètre. La méthode `GET` ajoute les variables à la fin de l'URL et est utilisée lorsque les variables sont peu nombreuses. La méthode `POST` place les variables dans un en-tête HTTP distinct et est utilisée pour des variables longues de type chaîne.

Description

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

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

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

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

Pour charger des variables dans un clip cible, préférez la fonction `loadVariables()` à la fonction `loadVariablesNum()`.

Voir aussi

`getURL()`, `loadMovie()`, `loadMovieNum()`, `loadVariables()`

mbchr()

Disponibilité

Flash Lite 1.0.

Utilisation

`mbchr(`*`number`*`)`

Opérandes

`number` Le nombre à convertir en caractère multi-octets.

Description

Fonction de la chaîne ; convertit un numéro de code ASCII en caractère multi-octets.

Exemple

L'exemple suivant convertit des numéros de code ASCII en caractère multi-octets équivalents :

```
trace (mbchr(65));// Sortie : A
trace (mbchr(97));// Sortie : a
trace (mbchr(36));// Sortie : $

myString = mbchr(51) - mbchr(49);
trace ("result = " add myString);// Sortie : résultat = 2
```

Voir aussi

`mblength()`, `mbsubstring()`

mblength()

Disponibilité

Flash Lite 1.0.

Utilisation

`mblength(`*`string`*`)`

Opérandes

`string` Une chaîne.

Description

Fonction de la chaîne ; renvoie la longueur de la chaîne de caractères multi-octets.

Exemple

L'exemple suivant affiche la longueur de la chaîne dans la variable `myString` :

```
myString = mbchr(36) add mbchr(50);
trace ("string length = " add mblength(myString));
// Sortie : longueur de chaîne = 2
```

Voir aussi

`mbchr()`, `mbsubstring()`

mbord()

Disponibilité

Flash Lite 1.0.

Utilisation

`mbord(`*`character`*`)`

Opérandes

`character` Le caractère à convertir en nombre multi-octets.

Description

Fonction de la chaîne ; convertit le caractère spécifié en nombre multi-octets.

Exemple

L'exemple suivant convertit les caractères de la variable `myString` en nombres multi-octets :

```
myString = "A";
trace ("ord = " add mbord(myString));// Sortie : 65

myString = "$120";
for (i=1; i<=length(myString); i++)
  char = substring(myString, i, 1);
  trace ("char ord = " add mbord(char));// Sortie : 36, 49, 50, 48
}
```

Voir aussi

`mbchr()`, `mbsubstring()`

mbsubstring()

Disponibilité

Flash Lite 1.0.

Utilisation

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

Opérandes

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

index Le numéro du premier caractère à extraire.

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

Description

Fonction de la chaîne ; extrait une nouvelle chaîne de caractères multi-octets d'une chaîne de caractères multi-octets.

Exemple

L'exemple suivant extrait une nouvelle chaîne de caractères multi-octets de la chaîne contenue dans la variable `myString` :

```
myString = mbchr(36) add mbchr(49) add mbchr(50) add mbchr(48);
trace (mbsubstring(myString, 0, 2));// Sortie : $1
```

Voir aussi

`mbchr()`

nextFrame()

Disponibilité

Flash Lite 1.0.

Utilisation

```
nextFrame()
```

Opérandes

Aucun.

Description

Fonction : envoie la tête de lecture jusqu'à l'image suivante et l'arrête.

Exemple

Dans l'exemple suivant, lorsque l'utilisateur clique sur le bouton, la tête de lecture se déplace jusqu'à l'image suivante et s'arrête :

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

Voir aussi

prevFrame()

nextScene()

Disponibilité

Flash Lite 1.0.

Utilisation

```
nextScene()
```

Opérandes

Aucun.

Description

Fonction : place la tête de lecture sur l'image 1 de la séquence suivante et l'arrête.

Exemple

Dans l'exemple suivant, lorsque l'utilisateur relâche le bouton, la tête de lecture se déplace jusqu'à l'image 1 de la séquence suivante :

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

Voir aussi

`prevScene()`

Number()

Disponibilité

Flash Lite 1.0.

Utilisation

`Number(`*`expression`*`)`

Opérandes

`expression` Une expression à convertir en nombre.

Description

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

- Si *`expression`* est un nombre, la valeur renvoyée est *`expression`*.
- Si *`expression`* est une valeur booléenne, la valeur renvoyée est 1 si *`expression`* est `true` ; 0 si *`expression`* est `false`.
- Si *`expression`* est une chaîne, la fonction tente d'analyser *`expression`* en tant que nombre décimal avec un exposant facultatif à la fin (ainsi, 1,57505e-3).
- Si *`expression`* est `undefined`, la valeur renvoyée est -1.

Exemple

L'exemple suivant convertit la chaîne de la variable `myString` en nombre, stocke ce nombre dans la variable `myNumber`, y ajoute 5 et enregistre le résultat dans la variable `myResult`. La ligne finale affiche le résultat lorsque la fonction `Number()` est appelée sur une valeur booléenne.

```
myString = "55";
myNumber = Number(myString);
myResult = myNumber + 5;

trace (myResult); // Sortie : 60

trace (Number(true));// Sortie : 1
```

on()

Disponibilité

Flash Lite 1.0.

Utilisation

```
on(event) {
  // statement(s)
}
```

Opérandes

`statement(s)` Les instructions à exécuter si un *`event`* se produit.

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

- `press` L'utilisateur appuie sur le bouton pendant que le pointeur de la souris le survole.
- `release` L'utilisateur relâche le bouton pendant que le pointeur de la souris le survole.
- `rollOut` Le pointeur quitte la zone du bouton.
- `rollOver` Le pointeur survole le bouton.
- `keyPress "`*`key`*`"` L'utilisateur appuie sur la touche spécifiée. Pour la section key du paramètre, spécifiez un code ou une constante de touche.

Description

Gestionnaire d'événement ; spécifie l'événement de type utilisateur ou pression de touche devant déclencher une fonction. Tous les événements ne sont pas pris en charge.

Exemple

Le code suivant qui fait défiler le champ `myText` d'une ligne vers le bas lorsque l'utilisateur appuie sur la touche 8, teste par rapport à `maxscroll` avant de faire défiler :

```
on (keyPress "8") {
  if (myText.scroll < myText.maxscroll) {
    myText.scroll++;
  }
}
```

ord()

Disponibilité

Flash Lite 1.0.

Utilisation

`ord(`*`character`*`)`

Opérandes

`character` Le caractère à convertir en numéro de code ASCII.

Description

Fonction de la chaîne ; convertit les caractères en numéros de code ASCII.

Exemple

L'exemple suivant utilise la fonction `ord()` pour afficher le code ASCII du caractère *A* :

```
trace ("multibyte number = " add ord("A"));// Sortie : nombre multi-octets
  = 65
```

play()

Disponibilité

Flash Lite 1.0.

Utilisation

`play()`

Opérandes

Aucun.

Description

Fonction ; fait avancer la tête de lecture au sein du scénario.

Exemple

L'exemple suivant utilise une instruction `if` pour vérifier la valeur d'un nom saisi par l'utilisateur. Si l'utilisateur entre `Steve`, la fonction `play()` est appelée, et la tête de lecture avance au sein du scénario. Si l'utilisateur entre un élément différent de `Steve`, le fichier SWF n'est pas lu, et un champ de texte contenant le nom de variable `alert` apparaît.

```
stop();
if (name == "Steve") {
  play();
} else {
  alert="You are not Steve!";
}
```

prevFrame()

Disponibilité

Flash Lite 1.0.

Utilisation

```
prevFrame()
```

Opérandes

Aucun.

Description

Fonction ; envoie la tête de lecture jusqu'à l'image précédente et l'arrête. Si l'image active est l'image 1, la tête de lecture ne bouge pas.

Exemple

Lorsque l'utilisateur clique sur un bouton auquel est associé le gestionnaire suivant, la tête de lecture est positionnée sur la séquence précédente :

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

Voir aussi

`nextFrame()`

prevScene()

Disponibilité

Flash Lite 1.0.

Utilisation

```
prevScene()
```

Opérandes

Aucun.

Description

Fonction ; place la tête de lecture sur l'image 1 de la séquence précédente et l'arrête.

Exemple

Dans cet exemple, lorsque l'utilisateur clique sur un bouton auquel est associé le gestionnaire suivant, la tête de lecture est positionnée sur l'image précédente :

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

Voir aussi

`nextScene()`

random()

Disponibilité

Flash Lite 1.0.

Utilisation

```
random(value)
```

Opérandes

value Un entier.

Description

Fonction ; renvoie un entier aléatoire compris entre 0 et un nombre inférieur à l'entier spécifié dans le paramètre *value*.

Exemple

Les exemples suivants génèrent un nombre en fonction d'un entier spécifiant la plage :

```
//choisir un nombre aléatoire compris entre 0 et 5
myNumber = random(5);
trace (myNumber);/// Sortie : 0,1,2,3 ou 4

//choisir un nombre aléatoire compris entre 5 et 10
myNumber = random(5) + 5;
trace (myNumber);// Sortie : 5,6,7,8 ou 9
```

Les exemples suivants génèrent un nombre, puis le concatènent à la fin d'une chaîne en cours d'évaluation sous forme de nom de variable. L'exemple suivant montre comment utiliser la syntaxe Flash Lite 1.1 pour simuler des tableaux.

```
// sélectionner un nom aléatoire dans la liste
myNames1 = "Mike";
myNames2 = "Debbie";
myNames3 = "Logan";

ran = random(3) + 1;
ranName = "myNames" add ran;
trace (eval(ranName));// Sortie : mike, debbie ou logan
```

removeMovieClip()

Disponibilité

Flash Lite 1.0.

Utilisation

`removeMovieClip(`*`target`*`)`

Opérandes

`target` Le chemin cible d'une occurrence de clip créée avec `duplicateMovieClip()`.

Description

Fonction ; supprime le clip spécifié initialement créé à l'aide de la méthode `duplicateMovieClip()`.

Exemple

L'exemple suivant supprime le clip dupliqué intitulé `second_mc` :

```
duplicateMovieClip("person_mc", "second_mc", 1);
second_mc:_x = 55;
second_mc:_y = 85;
removeMovieClip("second_mc");
```

set()

Disponibilité

Flash Lite 1.0.

Utilisation

`set(variable, expression)`

Opérandes

`variable` Un identificateur devant contenir la valeur du paramètre *`expression`*.

`expression` Une valeur affectée à la variable.

Description

Instruction ; associe une valeur à une variable. Une *variable* est un conteneur qui stocke des données. Le conteneur reste toujours le même, c'est le contenu qui peut varier. La modification de la valeur d'une variable pendant la lecture du fichier SWF permet d'enregistrer les informations relatives aux actions de l'utilisateur, d'enregistrer les valeurs modifiées pendant la lecture du fichier SWF ou d'évaluer si une condition est `true` ou `false`.

Les variables peuvent recouvrir tous les types de données, tels que String, Number, Boolean ou MovieClip. Le scénario de chaque fichier SWF et clip comporte son propre jeu de variables, et chaque variable dispose de sa propre valeur, indépendamment des variables des autres scénarios.

Exemple

L'exemple suivant définit une variable nommée `orig_x_pos` qui stocke la position d'origine de l'axe *x* du clip `ship` pour réinitialiser le bateau à sa position de départ ultérieurement dans le fichier SWF :

```
on(release) {
  set("orig_x_pos", getProperty("ship", _x));
}
```

Le code précédent donne le même résultat que le code suivant :

```
on(release) {
  orig_x_pos = ship._x;
}
```

setProperty()

Disponibilité

Flash Lite 1.0.

Utilisation

```
setProperty(target, property, value/expression)
```

Opérandes

target Le chemin du nom d'instance du clip dont la propriété doit être définie.

property La propriété à définir.

value La nouvelle valeur littérale de la propriété.

expression Une équation qui renvoie la nouvelle valeur de la propriété.

Description

Fonction ; modifie la valeur des propriétés d'un clip pendant la lecture de ce dernier.

Exemple

L'instruction suivante définit la propriété `_alpha` du clip `star` sur 30 pourcent lorsque l'utilisateur clique sur le bouton associé à ce gestionnaire d'événements :

```
on(release) {
  setProperty("star", _alpha, "30");
}
```

Voir aussi

`getProperty()`

stop()

Disponibilité

Flash Lite 1.0.

Utilisation

```
stop()
```

Opérandes

Aucun.

Description

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

Exemple

L'instruction suivante appelle la fonction `stop()` lorsque l'utilisateur clique sur le bouton associé à ce gestionnaire d'événements :

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

stopAllSounds()

Disponibilité

Flash Lite 1.0.

Utilisation

```
stopAllSounds()
```

Opérandes

Aucun.

Description

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

Exemple

Le code suivant peut être appliqué à un bouton qui permet d'arrêter tous les sons dans le fichier SWF lorsque l'utilisateur clique dessus :

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

String()

Disponibilité

Flash Lite 1.0.

Utilisation

`String(`*`expression`*`)`

Opérandes

`expression` Une expression à convertir en chaîne.

Description

Fonction ; renvoie une chaîne représentant le paramètre spécifié, comme indiqué dans la liste suivante :

- Si *`expression`* est un nombre, la chaîne renvoyée représente le nombre sous forme de texte.
- Si *`expression`* est une chaîne, la chaîne renvoyée est *`expression`*.
- Si *`expression`* est une valeur booléenne, la valeur renvoyée est `true` ou `false`.
- Si *`expression`* est un clip, la valeur renvoyée est le chemin cible du clip utilisant la notation à barre oblique (/).

Exemple

L'exemple suivant définit `birthYearNum` sur 1976, la convertit en chaîne à l'aide de la fonction `String()`, puis la compare à la chaîne « 1976 » en utilisant l'opérateur `eq`.

```
birthYearNum = 1976;
birthYearStr = String(birthYearNum);
if (birthYearStr eq "1976") {
  trace ("birthYears match");
}
```

substring()

Disponibilité

Flash Lite 1.0.

Utilisation

```
substring(string, index, count)
```

Opérandes

string La chaîne à partir de laquelle il convient d'extraire la nouvelle chaîne.

index Le numéro du premier caractère à extraire.

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

Description

Fonction ; extrait une partie d'une chaîne. Cette fonction est de base 1 tandis que les méthodes de la classe String sont de base 0.

Exemple

L'exemple suivant extrait les cinq premiers caractères de la chaîne « Hello World » :

```
origString = "Hello World!";
newString = substring(origString, 0, 5);
trace (newString);/// Sortie : Hello
```

tellTarget()

Disponibilité

Flash Lite 1.0.

Utilisation

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

Opérandes

target Une chaîne qui spécifie le chemin cible du scénario à contrôler.

statement(s) Les instructions à exécuter lorsque la condition renvoie la valeur `true`.

Description

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

Exemple

Dans l'exemple suivant, `tellTarget()` contrôle l'occurrence du clip `ball` dans le scénario principal. L'image 1 de l'occurrence `ball` est vide et est associée à une fonction `stop()` : elle n'est donc pas visible sur la scène. Lorsque l'utilisateur clique sur la touche 5, `tellTarget()` indique à la tête de lecture de `ball` d'atteindre l'image 2, où l'animation démarre.

```
on(keyPress "5") {
  tellTarget("ball") {
    gotoAndPlay(2);
  }
}
```

toggleHighQuality()

Disponibilité

Flash Lite 1.0.

Utilisation

```
toggleHighQuality()
```

Opérandes

Aucun.

Description

Fonction ; active et désactive l'anti-aliasing dans Flash Lite. L'anti-aliasing adoucit les bords des objets mais ralentit la lecture du fichier SWF. Cette fonction affecte tous les fichiers SWF dans Flash Lite.

Exemple

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

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

trace()

Disponibilité

Flash Lite 1.0.

Utilisation

`trace(`*`expression`*`)`

Opérandes

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

Description

Fonction ; évalue l'expression et affiche le résultat dans le panneau Sortie en mode test.

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

Vous pouvez également utiliser la commande Omettre les actions Trace pour supprimer les fonctions `trace()` du fichier SWF exporté.

Exemple

L'exemple suivant utilise la fonction `trace()` pour observer le comportement d'une boucle `while` :

```
i = 0;
while (i++ < 5){
  trace("this is execution " add i);
}
```

unloadMovie()

Disponibilité

Flash Lite 1.0.

Utilisation

`unloadMovie(target)`

Opérandes

target Le chemin cible d'un clip.

Description

Fonction ; supprime le clip qui a été chargé par l'intermédiaire de `loadMovie()`, `loadMovieNum()` ou *duplicateMovieClip()* de Flash Lite.

Exemple

Lorsque l'utilisateur clique sur la touche 3, le code suivant répond en purgeant le clip `draggable_mc` dans le scénario principal et en chargeant movie.swf dans le niveau 4 de la pile du document :

```
on (keypress "3") {
  unloadMovie ("/draggable_mc");
  loadMovieNum("movie.swf", 4);
}
```

Lorsque l'utilisateur clique sur la touche 3, l'exemple suivant purge le clip qui a été chargé au niveau 4 :

```
on (keypress "3") {
    unloadMovieNum(4);
}
```

Voir aussi

`loadMovie()`

unloadMovieNum()

Disponibilité

Flash Lite 1.0.

Utilisation

`unloadMovieNum(level)`

Opérandes

level Le niveau (_levelN) d'une animation chargée.

Description

Fonction ; supprime le clip qui a été chargé par l'intermédiaire de `loadMovie()`, `loadMovieNum()` ou *duplicateMovieClip()* de Flash Lite.

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

Voir aussi

`loadMovieNum()`

CHAPITRE 2

Propriétés de Flash Lite

2

Cette section décrit les propriétés reconnues par Macromedia Flash Lite 1.x. Les entrées sont classées par ordre alphabétique sans tenir compte des traits de soulignement de préfixe. Les propriétés sont présentées dans le tableau suivant.

Propriété	Description
`/ (barre de fraction)`	Spécifie ou renvoie une référence au scénario du clip principal.
`_alpha`	Renvoie la valeur de transparence alpha d'un clip.
`_currentframe`	Renvoie le numéro de l'image dans laquelle se trouve la tête de lecture dans le scénario.
`_focusrect`	Spécifie si un rectangle jaune doit s'afficher autour du bouton ou du champ de texte ayant le focus actuel.
`_framesloaded`	Renvoie le nombre d'images ayant été chargées à partir d'un fichier SWF lui-même chargé de façon dynamique.
`_height`	Spécifie la hauteur du clip, en pixels.
`_highquality`	Spécifie le niveau d'anti-aliasing appliqué au fichier SWF actuel.
`_level`	Renvoie une référence au scénario racine de `_levelN`. Vous devez utiliser la fonction `loadMovieNum()` pour charger des fichiers SWF dans le lecteur Flash Lite avant d'utiliser la propriété `_level` pour les cibler. Vous pouvez également utiliser `_levelN` pour cibler un fichier SWF chargé au niveau affecté par *N*.
`maxscroll`	Indique le numéro de ligne de la première ligne de texte visible dans un champ de texte défilant lorsque la dernière ligne du champ est également visible.
`_name`	Renvoie le nom d'occurrence d'un clip. Elle s'applique uniquement aux clips et non au scénario principal.
`_rotation`	Renvoie la rotation du clip, en degrés, à partir de son orientation d'origine.

Propriété	Description
scroll	Contrôle l'affichage des informations dans un champ de texte associé à une variable. La propriété `scroll` définit l'emplacement à partir duquel le champ de texte commence à afficher le contenu ; une fois l'emplacement défini, Flash Lite le met à jour lorsque l'utilisateur fait défiler le champ de texte.
_target	Renvoie le chemin cible de l'occurrence du clip.
_totalframes	Renvoie le nombre total d'images dans un clip.
_visible	Indique si un clip est visible.
_width	Renvoie la largeur du clip, en pixels.
_x	Contient un entier qui définit la coordonnée *x* d'un clip.
_xscale	Détermine le *pourcentage* de redimensionnement horizontal du clip tel qu'il est appliqué à partir du point d'alignement du clip.
_y	Contient un entier qui définit la coordonnée *y* d'un clip par rapport aux coordonnées locales du clip parent.
_yscale	Détermine le *pourcentage* de redimensionnement vertical du clip tel qu'il est appliqué à partir du point d'alignement du clip.

/ (barre de fraction)

Disponibilité

Flash Lite 1.0.

Utilisation

```
/

/targetPath

/:varName
```

Description

Identificateur ; spécifie ou renvoie une référence au scénario du clip principal. La fonctionnalité proposée par cette propriété est similaire à celle proposée par la propriété `_root` dans Flash 5.

Exemple

Pour spécifier une variable dans un scénario, utilisez une syntaxe à barre oblique (/) associée aux deux-points (:).

Exemple 1 : La variable `car` du scénario principal :

```
/:car
```

Exemple 2 : La variable `car` de l'occurrence mc1 du scénario principal :

```
/mc1/:car
```

Exemple 3 : La variable `car` de l'occurrence de clip mc2 imbriquée dans l'occurrence de clip mc1 du scénario principal :

```
/mc1/mc2/:car
```

Exemple 4 : La variable `car` de l'occurrence de clip mc2 du scénario actif :

```
mc2/:car
```

_alpha

Disponibilité

Flash Lite 1.0.

Utilisation

my_mc`:_alpha`

Propriété ; la valeur de transparence alpha du clip spécifié par la variable *my_mc*. Les valeurs possibles sont comprises entre 0 (entièrement transparent) et 100 (entièrement opaque), qui est la valeur par défaut. Les objets d'un clip dont la propriété `_alpha` est définie sur 0 sont actifs, même s'ils sont invisibles. Par exemple, vous pouvez cliquer sur un bouton dans un clip dont la propriété `_alpha` est définie sur 0.

Exemple

Le code suivant utilisé pour un gestionnaire d'événements définit la propriété `_alpha` du clip `my_mc` sur 30 % lorsque l'utilisateur clique sur le bouton associé à ce gestionnaire d'événements :

```
on(release) {
  tellTarget("my_mc") {
    _alpha = 30;
  }
}
```

_currentframe

Disponibilité

Flash Lite 1.0.

Utilisation

```
my_mc:_currentframe
```

Description

Propriété (en lecture seule) : renvoie le numéro de l'image dans laquelle se trouve la tête de lecture dans le scénario spécifié par la variable *my_mc*.

Exemple

L'exemple suivant utilise la propriété `_currentframe` et la fonction `gotoAndStop()` pour faire avancer de cinq images la tête de lecture du clip `my_mc`, par rapport à sa position actuelle :

```
tellTarget("my_mc") {
  gotoAndStop(_currentframe + 5);
}
```

Voir aussi

`gotoAndStop()`

_focusrect

Disponibilité

Flash Lite 1.0.

Utilisation

```
_focusrect = Boolean;
```

Description

Propriété (globale) : spécifie si un rectangle jaune doit s'afficher autour du bouton ou du champ de texte ayant le focus actuel. La valeur par défaut, `true`, affiche un rectangle jaune autour du bouton ou champ de texte ayant actuellement le focus lorsque l'utilisateur appuie sur les flèches Haut ou Bas de son téléphone ou périphérique portable pour naviguer dans les objets d'un fichier SWF. Spécifiez `false` si vous ne souhaitez pas afficher ce rectangle jaune.

Exemple

L'exemple suivant désactive l'affichage du rectangle de focus jaune dans l'application :

```
_focusrect = false;
```

_framesloaded

Disponibilité

Flash Lite 1.0.

Utilisation

my_mc:_framesloaded

Description

Propriété (en lecture seule) : le nombre d'images ayant été chargées à partir d'un fichier SWF lui-même chargé de façon dynamique. Cette propriété est utile pour déterminer si le contenu d'une image spécifique, et de toutes les images qui la précèdent, a été chargé et est disponible localement dans le navigateur. Elle est également utile pour contrôler le téléchargement de fichiers SWF volumineux. Par exemple, si désiré, affichez un message aux utilisateurs leur indiquant que le chargement du fichier SWF ne commencera pas tant que le chargement d'une image spécifiée dans le fichier SWF ne sera pas terminé.

Exemple

L'exemple suivant utilise la propriété `_framesloaded` pour activer un fichier SWF lorsque toutes les images sont chargées. Si certaines images ne sont pas chargées, la propriété `_xscale` du `chargeur` d'occurrence de clip est augmentée proportionnellement pour créer une barre de progression.

```
if (_framesloaded >= _totalframes) {
  gotoAndPlay ("Scene 1", "start");
} else {
  tellTarget(“loader”) {
    _xscale = (_framesloaded/_totalframes)*100;
  }
}
```

_height

Disponibilité

Flash Lite 1.0.

Utilisation

my_mc:_height

Description

Propriété (en lecture seule) : la hauteur du clip, en pixels.

Exemple

L'exemple suivant du code de gestionnaire d'événements définit la hauteur d'un clip lorsque l'utilisateur clique sur le bouton de la souris :

```
on(release) {
  tellTarget("my_mc") {
    _height = 200;
  }
}
```

_highquality

Disponibilité

Flash Lite 1.0.

Utilisation

```
_highquality
```

Description

Propriété (globale) : spécifie le niveau d'anti-aliasing appliqué au fichier SWF actuel. Spécifiez la valeur 2 pour obtenir un anti-aliasing de qualité optimale. Spécifiez la valeur 1 pour obtenir un anti-aliasing de qualité élevée. Spécifiez 0 pour empêcher l'anti-aliasing.

Exemple

L'instruction suivante applique un anti-aliasing de qualité élevée au fichier SWF actuel :

```
_highquality = 1;
```

Voir aussi

```
toggleHighQuality()
```

_level

Disponibilité

Flash Lite 1.0.

Utilisation

`_levelN`

Description

Identificateur ; une référence au scénario racine de `_levelN`. Vous devez utiliser la fonction `loadMovieNum()` pour charger des fichiers SWF dans le lecteur Flash Lite avant d'utiliser la propriété `_level` pour les cibler. Vous pouvez également utiliser `_levelN` pour cibler un fichier SWF chargé au niveau affecté par *N*.

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

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

Exemple

L'exemple suivant charge un fichier SWF dans le Niveau 1, puis arrête la tête de lecture du fichier SWF chargé sur l'image 6 :

```
loadMovieNum("mySWF.swf", 1);

// au moins 1 image plus loin
tellTarget(_level1) {
  gotoAndStop(6);
}
```

Voir aussi

`loadMovie()`

maxscroll

Disponibilité

Flash Lite 1.1

Utilisation

variable_name:maxscroll

Description

Propriété (en lecture seule) : indique le numéro de ligne de la première ligne de texte visible dans un champ de texte défilant lorsque la dernière ligne du champ est également visible. La propriété `maxscroll` travaille conjointement avec la propriété `scroll` pour contrôler la façon dont les informations apparaissent dans un champ de texte. Il est possible d'extraire cette propriété mais pas de la modifier.

Exemple

Le code suivant qui fait défiler le champ de texte `myText` d'une ligne vers le bas lorsque l'utilisateur appuie sur la touche 8, effectue un test par rapport à la propriété `maxscroll` avant de faire défiler :

```
on(keyPress "8") {
  if (myText:scroll < myText:maxscroll) {
    myText:scroll++;
  }
}
```

Voir aussi

scroll

_name

Disponibilité

Flash Lite 1.0.

Utilisation

my_mc:_name

Description

Propriété : le nom d'occurrence du clip spécifié par *my_mc*. Elle s'applique uniquement aux clips et non au scénario principal.

Exemple

L'exemple suivant affiche le nom du clip `bigRose` dans le panneau Sortie sous forme de chaîne :

```
trace(bigRose:_name);
```

_rotation

Disponibilité

Flash Lite 1.0.

Utilisation

my_mc`:_rotation`

Description

Propriété : rotation du clip, en degrés, à partir de son orientation d'origine. Les valeurs comprises entre 0 et 180 représentent la rotation en sens horaire ; les valeurs comprises entre 0 et -180 représentent la rotation en sens anti-horaire. Les valeurs non comprises dans cette plage sont ajoutées ou soustraites à 360 pour obtenir une valeur comprise dans la plage. Par exemple, l'instruction `my_mc:_rotation = 450` est identique à `my_mc:_rotation = 90`.

Exemple

L'exemple suivant fait pivoter le clip `my_mc` de 15 degrés dans le sens horaire lorsque l'utilisateur appuie sur la touche 2 :

```
on (keyPress "2") {
  my_mc:_rotation += 15;
}
```

scroll

Disponibilité

Flash Lite 1.1.

Utilisation

textFieldVariableName:scroll

Description

Propriété ; contrôle l'affichage des informations dans un champ de texte associé à une variable. La propriété `scroll` définit l'emplacement à partir duquel le champ de texte commence à afficher le contenu ; une fois l'emplacement défini, Flash Lite le met à jour lorsque l'utilisateur fait défiler le champ de texte. Vous pouvez utiliser la propriété `scroll` pour créer un champ de texte défilant ou pour diriger un utilisateur vers un paragraphe spécifique dans un long passage.

Exemple

Le code suivant fait défiler le champ de texte `myText` d'une ligne vers le haut chaque fois que l'utilisateur clique sur la touche 2 :

```
on(keyPress "2") {
  if (myText:scroll > 1) {
    myText:scroll--;
  }
}
```

Voir aussi

maxscroll

_target

Disponibilité

Flash Lite 1.0.

Utilisation

my_mc:_target

Description

Propriété (en lecture seule) ; renvoie le chemin cible de l'occurrence du clip spécifié par *my_mc*

_totalframes

Disponibilité

Flash Lite 1.0.

Utilisation

my_mc:_totalframes

Description

Propriété (en lecture seule) ; renvoie le nombre total d'images contenues dans le clip *my_mc*.

Exemple

Le code suivant charge mySWF.swf dans le niveau 1, puis vérifie 25 images plus loin s'il est bien chargé :

```
loadMovieNum("mySWF.swf", 1);

// 25 images plus loin dans le scénario principal
if (_level1._framesloaded >= _level1._totalframes) {
  tellTarget("_level1/") {
    gotoAndStop("myLabel");
  }
} else {
  // boucle...
}
```

_visible

Disponibilité

Flash Lite 1.0.

Utilisation

my_mc:_visible

Description

Propriété ; une valeur booléenne indiquant si le clip spécifié par *my_mc* est visible. Les clips qui ne sont pas visibles (propriété _visible définie sur false) sont désactivés. Par exemple, l'utilisateur ne peut pas cliquer sur le bouton d'un clip dont la propriété _visible est définie sur false. Les clips sont toujours visibles sauf action contraire explicitement définie.

Exemple

Le code suivant désactive le clip `my_mc` lorsque l'utilisateur appuie sur la touche 3, et l'active lorsque l'utilisateur appuie sur la touche 4 :

```
on(keyPress "3") {
  my_mc:_visible = 0;
 }

 on(keyPress "4") {
  my_mc:_visible = 1;
 }
```

_width

Disponibilité

Flash Lite 1.0.

Utilisation

my_mc:_width

Description

Propriété : largeur du clip, en pixels.

Exemple

L'exemple suivant définit les propriétés de largeur d'un clip lorsque l'utilisateur appuie sur la touche 5 :

```
on(keyPress "5") {
  my_mc:_width = 10;
}
```

_x

Disponibilité

Flash Lite 1.0.

Utilisation

my_mc:_x

Description

Propriété : un entier qui définit la coordonnée *x* d'un clip (représenté ici par *my_mc*) par rapport aux coordonnées locales du clip parent. Si un clip se trouve dans le scénario principal, son système de coordonnées se réfère alors au coin supérieur gauche de la scène : (0, 0).

Si le clip est imbriqué dans un autre clip subissant des transformations, le clip se trouve dans le système de coordonnées local du clip qui l'encadre. Par exemple, dans le cas d'un clip qui a effectué une rotation à 90 degrés en sens anti-horaire, les clips enfants héritent d'un système de coordonnées ayant effectué une rotation à 90 degrés en sens anti-horaire. Les coordonnées du clip renvoient à la position du point d'alignement.

Exemple

L'exemple suivant modifie la position horizontale du clip `my_mc` lorsque l'utilisateur appuie sur la touche 6 :

```
on(keyPress "6") {
  my_mc:_x = 10;
}
```

Voir aussi

`_xscale`, `_y`, `_yscale`

_xscale

Disponibilité

Flash Lite 1.0.

Utilisation

my_mc:_xscale

Description

Propriété ; détermine le *pourcentage* de redimensionnement horizontal du clip tel qu'il est appliqué à partir du point d'alignement du clip. Le point d'alignement par défaut est (0, 0).

Le redimensionnement du système de coordonnées local affecte les paramètres des propriétés _x et _y, définis en pixels. Par exemple, si le clip parent est redimensionné à 50 %, le paramétrage de la propriété _x déplace un objet dans le clip selon un nombre de pixels réduit de moitié par rapport à celui qui serait appliqué si le clip était défini sur 100 %.

Exemple

L'exemple suivant modifie le redimensionnement horizontal du clip `my_mc` lorsque l'utilisateur appuie sur la touche 7 :

```
on(keyPress "7") {
  my_mc:_xscale = 10;
}
```

Voir aussi

_x, _y, _yscale

_y

Disponibilité

Flash Lite 1.0.

Utilisation

my_mc:_y

Description

Propriété : un entier qui définit la coordonnée *y* d'un clip (représenté ici par *my_mc*) par rapport aux coordonnées locales du clip parent. Si un clip se trouve dans le scénario principal, son système de coordonnées se réfère alors au coin supérieur gauche de la scène : (0, 0).

Si le clip est imbriqué dans un autre clip subissant des transformations, le clip se trouve dans le système de coordonnées local du clip qui l'encadre. Par exemple, dans le cas d'un clip qui a effectué une rotation à 90 degrés en sens anti-horaire, les clips enfants héritent d'un système de coordonnées ayant effectué une rotation à 90 degrés en sens anti-horaire. Les coordonnées du clip renvoient à la position du point d'alignement.

Exemple

Le code suivant définit les coordonnées *y* du clip `my_mc` sur 10 pixels au-dessous de la coordonnée (0, 0) de son clip parent lorsque l'utilisateur appuie sur la touche 1 :

```
on(keyPress "1") {
  my_mc:_y = 10;
}
```

Voir aussi

_x, _xscale, _yscale

_yscale

Disponibilité

Flash Lite 1.0.

Utilisation

my_mc:_yscale

Description

Propriété ; détermine le *pourcentage* de redimensionnement vertical du clip tel qu'il est appliqué à partir du point d'alignement du clip. Le point d'alignement par défaut est (0, 0).

Le redimensionnement du système de coordonnées local affecte les paramètres des propriétés _x et _y, définis en pixels. Par exemple, si le clip parent est redimensionné à 50 %, le paramétrage de la propriété _y déplace un objet dans le clip selon un nombre de pixels réduit de moitié par rapport à celui qui serait appliqué si le clip était défini sur 100 %.

Exemple

L'exemple suivant redéfinit le redimensionnement vertical du clip my_mc sur 10 % lorsque l'utilisateur appuie sur la touche 1 :

```
on(keyPress "1") {
  my_mc:_yscale = 10;
}
```

Voir aussi

_x, _xscale, _y

CHAPITRE 3

Instructions de Flash Lite

3

Cette section décrit la syntaxe et le rôle des instructions ActionScript de Macromedia Flash Lite 1.x qui sont des éléments de langage permettant d'effectuer ou de spécifier une action. Les instructions sont présentées dans le tableau suivant.

Instruction	Description
break	Force Flash Lite à ignorer le reste du corps de la boucle, arrête l'action de la boucle et exécute l'instruction suivant l'instruction de bouclage.
case	Définit une condition pour l'instruction `switch`. Les instructions du paramètre *`statements`* s'exécutent si le paramètre *`expression`* suivant le mot-clé `case` correspond au paramètre `expression` de l'instruction `switch`.
continue	Ignore toutes les instructions restantes dans la boucle imbriquée de plus bas niveau et passe à l'itération suivante, comme si le contrôle avait été transmis à la fin de la boucle normalement.
do..while	Exécute les instructions, puis évalue la condition dans une boucle tant que la condition renvoie `true`.
else	Spécifie les instructions à exécuter si la condition incluse dans l'instruction `if` renvoie `false`.
else if	Evalue une condition et spécifie les instructions à exécuter si la condition incluse dans l'instruction `if` initiale renvoie `false`.
for	Evalue l'expression *`init`* (initialiser) une fois, puis amorce une séquence de bouclage à partir de laquelle, tant que *`condition`* renvoie `true`, *`statement`* est exécutée et l'expression suivante évaluée.
if	Evalue une condition pour déterminer l'action suivante d'un fichier SWF. Lorsque cette condition est `true`, Flash Lite exécute les instructions qui suivent la condition entre accolades (`{}`). Si la condition est `false`, Flash Lite ignore les instructions entre accolades et exécute les instructions qui suivent ces accolades.

Instruction	Description
switch	Comme pour l'instruction `if`, l'instruction `switch` teste une condition et exécute des instructions si cette condition renvoie la valeur `true`.
while	Teste une expression et exécute une instruction ou série d'instructions à plusieurs reprises dans une boucle tant que l'expression renvoie `true`.

break

Disponibilité

Flash Lite 1.0.

Utilisation

```
break
```

Paramètres

Aucun.

Description

Instruction ; apparaît au sein d'une boucle (`for`, `do..while` ou `while`) ou dans un bloc d'instructions associées à un cas donné au sein d'une instruction `switch`. L'instruction `break` force Flash Lite à ignorer le reste du corps de la boucle, arrête l'action de la boucle et exécute l'instruction suivant l'instruction de bouclage. Lorsque vous utilisez l'instruction `break`, l'interpréteur d'ActionScript ignore le reste des instructions de ce bloc `case` et passe à la première instruction suivant l'instruction `switch` qui l'encadre. Utilisez cette instruction pour sortir d'une série de boucles imbriquées.

Exemple

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

```
i = 0;
while (true) {
  if (i >= 100) {
    break;
  }
  i++;
}
```

Voir aussi

`case`, `do..while`, `for`, `switch`, `while`

case

Disponibilité

Flash Lite 1.0.

Utilisation

`case` *`expression`*`:` *`statements`*

Paramètres

`expression` Toute expression.

`statements` Toute instruction.

Description

Instruction ; définit une condition pour l'instruction `switch`. Les instructions du paramètre *`statements`* s'exécutent si le paramètre *`expression`* suivant le mot-clé `case` correspond au paramètre *`expression`* de l'instruction `switch`.

Si vous utilisez l'instruction `case` en dehors d'une instruction `switch`, ceci produit une erreur et le code ne se compile pas.

Exemple

Dans l'exemple suivant, si le paramètre `myNum` renvoie 1, l'instruction `trace()` suivant `case 1` s'exécute ; si le paramètre `myNum` renvoie 2, l'instruction `trace()` suivant `case 2` s'exécute, etc. Si aucune expression `case` ne correspond au paramètre `number`, l'instruction `trace()` suivant le mot-clé `default` s'exécute.

```
switch (myNum) {
  case 1:
    trace ("case 1 tested true");
    break;
  case 2:
    trace ("case 2 tested true");
    break;
  case 3:
    trace ("case 3 tested true");
    break;
  default:
    trace ("no case tested true")
}
```

Dans l'exemple suivant, aucune instruction break n'est effectuée dans le premier groupe de cas, par conséquent si le nombre est 1, les valeurs `A` et `B` apparaissent dans le panneau Sortie :

```
switch (myNum) {
  case 1:
    trace ("A");
  case 2:
    trace ("B");
    break;
  default:
    trace ("D")
}
```

Voir aussi

`switch`

continue

Disponibilité

Flash Lite 1.0.

Utilisation

```
continue
```

Paramètres

Aucun.

Description

Instruction ; ignore toutes les instructions restantes dans la boucle imbriquée de plus bas niveau et passe à l'itération suivante, comme si le contrôle avait été transmis à la fin de la boucle normalement. Elle n'a aucun effet en dehors d'une boucle.

- Dans une boucle `while`, l'instruction `continue` oblige l'interpréteur de Flash à ignorer le reste du corps de la boucle et à atteindre le haut de la boucle, où la condition est testée.
- Dans une boucle `do..while`, l'instruction `continue` oblige l'interpréteur de Flash à ignorer le reste du corps de la boucle et à atteindre le bas de la boucle, où la condition est testée.
- Dans une boucle `for`, l'instruction `continue` oblige l'interpréteur de Flash à ignorer le reste du corps de la boucle et à atteindre l'évaluation de la post-expression `for` de la boucle.

Exemple

Dans la boucle `while` suivante, l'instruction `continue` oblige Flash Lite à ignorer le reste du corps de la boucle et à atteindre le haut de la boucle, où la condition est testée :

```
i = 0;
while (i < 10) {
  if (i % 3 == 0) {
    i++;
    continue;
  }
  trace(i);
  i++;
}
```

Dans la boucle `do..while` suivante, l'instruction `continue` oblige Flash Lite à ignorer le reste du corps de la boucle et à atteindre le bas de la boucle, où la condition est testée :

```
i = 0;
do {
  if (i % 3 == 0) {
    i++;
    continue;
  }
  trace(i);
  i++;
} while (i < 10);
```

Dans une boucle `for`, l'instruction `continue` oblige Flash Lite à ignorer le reste du corps de la boucle. Dans l'exemple suivant, si `i` modulo 3 est égal à 0, l'instruction `trace(i)` est ignorée :

```
for (i = 0; i < 10; i++) {
  if (i % 3 == 0) {
    continue;
  }
  trace(i);
}
```

Voir aussi

`do..while`, `for`, `while`

do..while

Disponibilité

Flash Lite 1.0.

Utilisation

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

Paramètres

statement(s) Les instructions à exécuter tant que le paramètre *condition* renvoie la valeur `true`.

condition La condition à évaluer.

Description

Instruction ; exécute les instructions, puis évalue la condition dans une boucle tant que la condition renvoie `true`.

Exemple

L'exemple suivant incrémente la variable d'indice tant que la valeur de cette dernière est inférieure à 10 :

```
i = 0;
 do {
  //trace (i);    // sortie : 0,1,2,3,4,5,6,7,8,9
  i++;
 } while (i<10);
```

Voir aussi

`break`, `continue`, `for`, `while`

else

Disponibilité

Flash Lite 1.0.

Utilisation

```
if (condition){
   t-statement(s);
} else {
   f-statement(s);
}
```

Disponibilité

condition Une expression qui renvoie `true` ou `false`.

t-statement(s) Les instructions à exécuter si la condition renvoie la valeur `true`.

f-statement(s) Une autre série d'instructions à exécuter si la condition renvoie la valeur `false`.

Description

Instruction ; spécifie les instructions à exécuter si la condition incluse dans l'instruction `if` renvoie `false`.

Exemple

L'exemple suivant indique comment utiliser l'instruction `else` avec une condition. Un exemple réel comprendrait un code devant effectuer une action quelconque en fonction de la condition.

```
currentHighestDepth = 1;
 if (currentHighestDepth == 2) {
  //trace ("currentHighestDepth is 2");
 } else {
  //trace ("currentHightestDepth is not 2");
 }
```

Voir aussi

`if`

else if

Disponibilité

Flash Lite 1.0.

Utilisation

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

Paramètres

condition Une expression qui renvoie `true` ou `false`.

statement(s) Une série d'instructions à exécuter si la condition spécifiée dans l'instruction `if` renvoie `false`.

Description

Instruction ; évalue une condition et spécifie les instructions à exécuter si la condition incluse dans l'instruction `if` initiale renvoie `false`. Lorsque la condition `else if` renvoie `true`, l'interpréteur Flash exécute les instructions qui suivent la condition `else if` entre accolades ({}). Si la condition `else if` renvoie `false`, Flash ignore les instructions entre accolades et exécute les instructions qui suivent ces accolades. Utilisez l'instruction `else if` pour créer des arborescences logiques dans vos scripts.

Exemple

L'exemple suivant utilise les instructions `else if` pour vérifier si tous les côtés d'un objet se trouvent dans une limite spécifique :

```
person_mc.xPos = 100;
leftBound = 0;
rightBound = 100;

if (person_mc.xPos <= leftBound) {
  //trace ("Clip is to the far left");
} else if (person_mc.xPos >= rightBound) {
  //trace ("Clip is to the far right");
} else {
  //trace ("Your clip is somewhere in between");
}
```

Voir aussi

`if`

for

Disponibilité

Flash Lite 1.0.

Utilisation

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

Paramètres

init Une expression à évaluer avant d'amorcer la séquence de bouclage, généralement une expression d'affectation.

condition Une expression qui renvoie `true` ou `false`. La condition est évaluée avant chaque itération de la boucle ; la boucle s'arrête lorsque la condition renvoie `false`.

next Une expression à évaluer après chaque itération de la boucle ; généralement une expression d'affectation utilisant les opérateurs d'incrémentation (++) ou de décrémentation (--).

statement(s) Une ou plusieurs instructions à exécuter dans la boucle.

Description

Instruction ; une construction de boucle qui évalue l'expression *init* (initialiser) une fois, puis amorce une séquence de bouclage à partir de laquelle, tant que *condition* renvoie `true`, *statement* est exécutée et l'expression suivante évaluée.

Certaines propriétés ne peuvent pas être énumérées par les instructions `for` ou `for..in`. Par exemple, les propriétés de clip, telles que `_x` et `_y`, ne sont pas énumérées.

Exemple

L'exemple suivant utilise la boucle `for` pour additionner les nombres de 1 à 100 :

```
sum = 0;
  for (i = 1; i <= 100; i++) {
    sum = sum + i;
  }
```

Voir aussi

`++ (incrément)`, `-- (décrément)`, `do..while`, `while`

if

Disponibilité

Flash Lite 1.0.

Utilisation

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

Paramètres

condition Une expression qui renvoie `true` ou `false`.

statement(s) Les instructions à exécuter lorsque la condition renvoie la valeur `true`.

Description

Instruction ; évalue une condition pour déterminer l'action suivante d'un fichier SWF. Lorsque cette condition est `true`, Flash Lite exécute les instructions qui suivent la condition entre accolades ({}). Si la condition est `false`, Flash Lite ignore les instructions entre accolades et exécute les instructions qui suivent ces accolades. Utilisez l'instruction `if` pour créer des arborescences logiques dans vos scripts.

Exemple

Dans l'exemple suivant, la condition entre parenthèses évalue la variable `name` pour vérifier que sa valeur littérale est bien `"Erica"`. Le cas échéant, la fonction `play()` s'exécute.

```
if(name eq "Erica"){
  play();
}
```

switch

Disponibilité

Flash Lite 1.0.

Utilisation

```
switch (expression){
  caseClause:
  [defaultClause:]
}
```

Paramètres

expression Toute expression numérique.

caseClause Un mot-clé `case` suivi par une expression, deux points et un groupe d'instructions à exécuter si l'expression correspond au paramètre *expression* de l'instruction switch.

defaultClause Un mot-clé `default` facultatif suivi par les instructions à exécuter si aucune des expressions case ne correspond au paramètre *expression* de l'instruction switch.

Description

Instruction ; crée une structure arborescente pour les instructions ActionScript. Comme pour l'instruction `if`, l'instruction `switch` teste une condition et exécute des instructions si cette condition renvoie la valeur `true`.

Les instructions switch contiennent une option de secours appelée default. Si aucune autre instruction n'est vraie, l'instruction default est exécutée.

Exemple

Dans l'exemple suivant, si le paramètre `myNum` renvoie 1, l'instruction `trace()` suivant `case 1` s'exécute ; si le paramètre `myNum` renvoie 2, l'instruction `trace()` suivant `case 2` s'exécute, etc. Si aucune expression `case` ne correspond au paramètre `number`, l'instruction `trace()` suivant le mot-clé `default` s'exécute.

```
switch (myNum) {
  case 1:
    trace ("case 1 tested true");
    break;
  case 2:
    trace ("case 2 tested true");
    break;
  case 3:
    trace ("case 3 tested true");
    break;
  default:
```

```
    trace ("no case tested true")
}
```

Dans l'exemple suivant, le premier groupe de cas ne contient pas d'instruction break, par conséquent si le nombre est 1, les valeurs A et B apparaissent dans le panneau Sortie :

```
switch (myNum) {
  case 1:
    trace ("A");
  case 2:
    trace ("B");
    break;
  default:
    trace ("D")
}
```

Voir aussi

`case`

while

Disponibilité

Flash Lite 1.0.

Utilisation

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

`Disponibilité`

`condition` L'expression qui est évaluée chaque fois que l'instruction `while` s'exécute.

`statement(s)` Les instructions à exécuter lorsque la condition renvoie la valeur `true`.

Description

Instruction ; teste une expression et exécute une instruction ou série d'instructions à plusieurs reprises dans un boucle tant que l'expression renvoie `true`.

Avant d'exécuter le bloc d'instructions, la condition est testée ; si le test renvoie la valeur `true`, le bloc d'instructions est exécuté. Si la condition renvoie `false`, le bloc d'instructions est ignoré et la première instruction après le bloc d'instructions de l'instruction `while` est exécutée.

Les boucles permettent d'exécuter une action tant que la valeur de la variable de décompte est inférieure à la valeur spécifiée. A la fin de chaque boucle, le compteur est incrémenté jusqu'à ce qu'il atteigne la valeur maximale spécifiée. A ce stade, la condition n'est plus `true` et la boucle se termine.

L'instruction `while` exécute les séries d'instructions suivantes. Toute répétition des étapes 1 à 4 constitue une *itération* de la boucle. La condition est testée au début de chaque itération :

1. L'expression *`condition`* est évaluée.
2. Si *`condition`* renvoie `true` ou une valeur convertie en valeur booléenne `true`, telle qu'un nombre différent de zéro, passez à l'étape 3.

   Sinon, l'instruction `while` se termine et l'exécution reprend au niveau de l'instruction qui suit la boucle `while`.
3. Exécutez le bloc d'instructions *`statement(s)`*.
4. Passez à l'étape 1.

Exemple

L'exemple suivant exécute une boucle tant que la valeur de la variable d'indice `i` est inférieure à 10 :

```
i = 0;
while(i < 10) {
  trace ("i = " add ++i);// Output: 1,2,3,4,5,6,7,8,9
}
```

Voir aussi

`continue`, `do..while`, `for`

CHAPITRE 4

Opérateurs de Flash Lite

4

Cette section décrit la syntaxe et le rôle des opérateurs ActionScript de Macromedia Flash Lite 1.x. Toutes les entrées sont classées par ordre alphabétique. Cependant, certains opérateurs sont des symboles et sont classés par rapport à leurs descriptions.

Les opérateurs de cette section sont présentés dans le tableau suivant.

Opérateur	Description
`add (concaténation de chaîne)`	Concatène (combine) au moins deux chaînes.
`+= (affectation d'addition)`	Affecte à *`expression1`* la valeur de *`expression1 + expression2`*.
`and`	Effectue une opération AND logique.
`= (affectation)`	Affecte la valeur d'*`expression2`* (l'opérande de droite) à la variable ou à la propriété dans *`expression1`*.
`/* (bloc de commentaires)`	Démarque une ou plusieurs lignes de commentaires de script. Tout caractère qui s'affiche entre la balise ouvrante de commentaires (/*) et la balise fermante (*/) est interprété en tant que commentaire et ignoré par l'interpréteur d'ActionScript.
`, (virgule)`	Evalue *`expression1`*, puis *`expression2`* et renvoie la valeur de *`expression2`*.
`// (commentaire)`	Signale le début d'un commentaire de script. Tout caractère qui s'affiche entre le séparateur de commentaires (//) et le caractère de fin de ligne est interprété en tant que commentaire et ignoré par l'interpréteur d'ActionScript.
`?: (conditionnel)`	Oblige Flash Lite à évaluer *`expression1`*, et si la valeur de *`expression1`* est `true`, l'opérande renvoie la valeur de *`expression2`* ; sinon, la valeur de *`expression3`* est renvoyée.

Opérateur	Description
`-- (décrément)`	Soustrait 1 de *expression*. La forme pré-décrémentale de l'opérateur (*--expression*) soustrait 1 de *expression* et renvoie le résultat sous forme de nombre. La forme post-décrémentale de l'opérateur (*expression--*) soustrait 1 de *expression* et renvoie la valeur initiale de *expression* (la valeur avant soustraction).
`/ (division)`	Divise *expression1* par *expression2*.
`/= (affectation de division)`	Affecte à *expression1* la valeur de *expression1 / expression2*.
`. (point)`	Permet de naviguer au sein des hiérarchies de clips pour accéder aux clips imbriqués (enfants), aux variables ou aux propriétés.
`++ (incrément)`	Ajoute 1 à *expression*. L'expression peut être une variable, un élément de tableau ou une propriété d'objet. La forme pré-incrémentale de l'opérateur (*++expression*) ajoute 1 à *expression* et renvoie le résultat sous forme de nombre. La forme post-incrémentale de l'opérateur (*expression++*) ajoute 1 à *expression* et renvoie la valeur initiale de *expression* (la valeur avant addition).
`&& (AND logique)`	Evalue *expression1* (l'expression située à gauche de l'opérateur) et renvoie `false` si cette expression renvoie `false`. Si *expression1* renvoie `true`, *expression2* (l'expression située à droite de l'opérateur) est évaluée. Si *expression2* renvoie `true`, le résultat final est `true` ; sinon, `false`.
`! (NOT logique)`	Inverse la valeur booléenne d'une variable ou d'une expression. Si *expression* est une variable dont la valeur absolue ou convertie est `true`, la valeur de `!`*expression* est `false`. Si l'expression `x && y` renvoie `false`, l'expression `!(x && y)` renvoie `true`.
`|| (OR logique)`	Evalue *expression1* et *expression2*. Le résultat est `true` si l'une des expressions, voire les deux, renvoie(nt) `true`. Le résultat est `false` si et uniquement si les deux expressions renvoient `false`. Vous pouvez utiliser l'opérateur OR logique avec autant d'opérandes que nécessaire. Si l'un des opérandes renvoie `true`, le résultat est `true`.
`% (modulo)`	Calcule le reste de *expression1* divisée par *expression2*. Si un opérande *expression* n'est pas numérique, l'opérateur modulo tente de le convertir en nombre.
`%= (affectation modulo)`	Affecte à *expression1* la valeur de *expression1 % expression2*.
`*= (affectation de multiplication)`	Affecte à *expression1* la valeur de *expression1 * expression2*.
`* (produit)`	Multiplie deux expressions numériques.

Opérateur	Description
+ (addition numérique)	Ajoute des expressions numériques.
== (égalité numérique)	Teste l'égalité ; si *expression1* est égale à *expression2*, le résultat est `true`.
> (numérique supérieur à)	Compare deux expressions et détermine si *expression1* est supérieure à *expression2* ; le cas échéant, l'opérateur renvoie *true*. Si *expression1* est inférieure ou égale à *expression2*, l'opérateur renvoie `false`.
>= (numérique supérieur ou égal à)	Compare deux expressions et détermine si *expression1* est supérieure ou égale à *expression2* `(true)` ou si *expression1* est inférieure à *expression2* `(false)`.
<> (inégalité numérique)	Teste l'égalité ; si *expression1* est égale à *expression2*, le résultat est `false`.
< (numérique inférieur à)	Compare deux expressions et détermine si *expression1* est inférieure à *expression2* ; le cas échéant, l'opérateur renvoie `true`. Si *expression1* est supérieure ou égale à *expression2*, l'opérateur renvoie `false`.
<= (numérique inférieur ou égal à)	Compare deux expressions et détermine si *expression1* est inférieure ou égale à *expression2*. Le cas échéant, l'opérateur renvoie `true` ; `false` dans le cas contraire.
() (parenthèses)	Regroupe un ou plusieurs paramètres, évalue les expressions de façon séquentielle ou entoure un ou plusieurs paramètres et les transmet en tant que paramètres à une fonction en dehors des parenthèses.
" " (séparateur de chaîne)	Lorsqu'ils entourent une séquence de zéro ou plusieurs caractères, les guillemets indiquent que ces caractères ont une valeur littérale et doivent être traités en tant que *chaîne* et non pas en tant que variable, valeur numérique ou tout autre élément ActionScript.
eq (égalité de chaîne)	Compare l'égalité de deux expressions et renvoie `true` si la chaîne représentant *expression1* est égale à celle de *expression2* ; sinon, l'opération renvoie `false`.
gt (chaîne supérieure à)	Compare la chaîne représentant *expression1* à la chaîne représentant *expression2* et renvoie `true` si *expression1* est supérieure à *expression2* ; `false` dans le cas contraire.
ge (chaîne supérieure ou égale à)	Compare la chaîne représentant *expression1* à la chaîne représentant *expression2* et renvoie une valeur `true` si *expression1* est supérieure ou égale à *expression2* ; `false` dans le cas contraire.

Opérateur	Description
ne (inégalité de chaîne)	Compare les chaînes représentant *expression1* et *expression2* et renvoie `true` si *expression1* n'est pas égale à *expression2* ; `false` dans le cas contraire.
lt (chaîne inférieure à)	Compare la chaîne représentant *expression1* à la chaîne représentant *expression2* et renvoie `true` si *expression1* est inférieure à *expression2* ; `false` dans le cas contraire.
le (chaîne inférieure ou égale à)	Compare la chaîne représentant *expression1* à la chaîne représentant *expression2* et renvoie une valeur `true` si *expression1* est inférieure ou égale à *expression2* ; `false` dans le cas contraire.
- (soustraction)	Utilisé pour la négation ou la soustraction.
-= (affectation de soustraction)	Affecte à *expression1* la valeur de *expression1 - expression2*.

add (concaténation de chaîne)

Disponibilité

Flash Lite 1.0.

Utilisation

string1 `add` *string2*

Opérandes

string1, string2 Chaînes.

Description

Opérateur ; concatène (combine) au moins deux chaînes.

Exemple

L'exemple suivant combine deux valeurs de chaîne pour générer la chaîne *catalogue*.

```
conStr = "cat" add "alog";
trace (conStr);// sortie : catalog
```

Voir aussi

+ (addition numérique)

+= (affectation d'addition)

Disponibilité

Flash Lite 1.0.

Utilisation

expression1 += *expression2*

Opérandes

expression1, expression2 Nombres ou chaînes.

Description

Opérateur (affectation de composant arithmétique) ; affecte à *expression1* la valeur de *expression1* + *expression2*. Par exemple, les deux instructions suivantes ont le même résultat :

```
x += y;
x = x + y;
```

Toutes les règles de l'opérateur d'addition (+) s'appliquent à l'opérateur d'affectation de l'addition (+=).

Exemple

L'exemple suivant utilise l'opérateur d'affectation d'addition (+=) pour ajouter à la valeur de `x` la valeur de `y` :

```
x = 5;
y = 10;
x += y;
trace(x);// sortie : 15
```

Voir aussi

+ (addition numérique)

and

Disponibilité

Flash Lite 1.0.

Utilisation

condition1 and *condition2*

Opérandes

condition1, condition2 Conditions ou expressions qui renvoient `true` ou `false`.

Description

Opérateur ; effectue une opération AND logique.

Exemple

L'exemple suivant utilise l'opérateur `and` pour vérifier si un joueur a remporté la partie. Les variables `turns` et `score` sont mises à jour lorsqu'un joueur prend un tour ou marque des points durant la partie. Le script suivant affiche « You Win the Game! » dans le panneau Sortie lorsque le joueur obtient un score de 75 ou plus en trois tours maximum.

```
turns = 2;
score = 77;
winner = (turns <= 3) and (score >= 75);
if (winner) {
  trace("You Win the Game!");
} else {
  trace("Try Again!");
}
// sortie : You Win the Game!
```

Voir aussi

`&& (AND logique)`

= (affectation)

Disponibilité

Flash Lite 1.0.

Utilisation

```
expression1 = expression2
```

Opérandes

expression1 Une variable ou une propriété.

expression2 Une valeur.

Description

Opérateur ; affecte la valeur de *expression2* (l'opérande de droite) à la variable ou à la propriété dans *expression1*.

Exemple

L'exemple suivant utilise l'opérateur d'affectation (=) pour affecter une valeur numérique à la variable `weight` :

```
weight = 5;
```

L'exemple suivant utilise l'opérateur d'affectation (=) pour affecter une valeur de chaîne à la variable `greeting` :

```
greeting = "Hello, " and personName;
```

/* (bloc de commentaires)

Disponibilité

Flash Lite 1.0

Utilisation

```
/* comment */
/* comment
comment */
```

Opérandes

comment Tout caractère.

Description

Séparateur de commentaires ; démarque une ou plusieurs lignes de commentaires de script. Tout caractère qui s'affiche entre la balise ouvrante de commentaires (/*) et la balise fermante (*/) est interprété en tant que commentaire et ignoré par l'interpréteur d'ActionScript.

Préférez l'opérateur // (séparateur de commentaires) pour les commentaires sur une ligne. Retenez l'opérateur /* pour identifier les commentaires répartis sur plusieurs lignes. L'omission de la balise fermante (*/) renvoie un message d'erreur. Le fait d'incorporer plusieurs balises de commentaires les unes dans les autres renvoie également un message d'erreur.

Ainsi, lorsque vous utilisez une balise ouvrante (/*), la première balise fermante (*/) termine ce commentaire, quel que soit le nombre de balises (/*) intercalées.

Voir aussi

`// (commentaire)`

, (virgule)

Disponibilité

Flash Lite 1.0.

Utilisation

`expression1, expression2`

Opérandes

`expression1, expression2` Nombres ou expressions renvoyant un nombre.

Description

Opérateur ; évalue *`expression1`*, puis *`expression2`* et renvoie la valeur de *`expression2`*.

Exemple

L'exemple suivant utilise l'opérateur virgule (,) sans l'opérateur de parenthèses `()` et affiche que l'opérateur virgule renvoie uniquement la valeur de la première expression ne contenant pas l'opérateur parenthèses `()` :

```
v = 0;
v = 4, 5, 6;
trace(v); // sortie : 4
```

L'exemple suivant utilise l'opérateur virgule (,) sans l'opérateur de parenthèses `()` et affiche que l'opérateur virgule renvoie uniquement la valeur de la dernière expression lorsqu'il est utilisé avec l'opérateur parenthèses `()` :

```
v = 0;
v = (4, 5, 6);
trace(v); // sortie : 6
```

L'exemple suivant utilise l'opérateur virgule (,) sans l'opérateur parenthèses () et affiche que l'opérateur virgule évalue de façon séquentielle toutes les expressions mais renvoie uniquement la valeur de la première. La deuxième expression, z++, est évaluée et z est incrémentée de 1.

```
v = 0;
z = 0;
v = v + 4 , z++, v + 6;
trace(v); // sortie : 4
trace(z); // sortie : 1
```

L'exemple suivant est identique à l'exemple précédent, à l'exception de l'ajout de l'opérateur de parenthèses () ; il affiche une nouvelle fois que, lorsqu'il est utilisé avec l'opérateur parenthèses (), l'opérateur virgule (,) renvoie la valeur de la dernière expression de la série :

```
v = 0;
z = 0;
v = (v + 4, z++, v + 6);
trace(v); // sortie : 6
trace(z); // sortie : 1
```

Voir aussi

for, () (parenthèses)

// (commentaire)

Disponibilité

Flash Lite 1.0

Utilisation

```
// (commentaire)
```

Opérandes

comment Tout caractère.

Description

Séparateur de commentaires ; signale le début d'un commentaire de script. Tout caractère qui s'affiche entre le séparateur de commentaires (//) et le caractère de fin de ligne est interprété en tant que commentaire et ignoré par l'interpréteur d'ActionScript.

Exemple

L'exemple suivant utilise des séparateurs de commentaires pour identifier les première, troisième, cinquième et septième lignes sous forme de commentaires :

```
// Enregistrer la position X du clip ball.
ballX = ball._x;
// Enregistrer la position Y du clip ball.
ballY = ball._y;
// Enregistrement de la position X du clip bat.
batX = bat._x;
// Enregistrement de la position Y du clip bat.
batY = bat._y;
```

Voir aussi

`/* (bloc de commentaires)`

?: (conditionnel)

Disponibilité

Flash Lite 1.0.

Utilisation

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

Opérandes

`expression1` Une expression qui renvoie une valeur booléenne, généralement une expression de comparaison, telle que `x < 5`.

`expression2`, *`expression3`* Valeurs de tout type.

Description

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

Exemple

L'exemple suivant affecte la valeur de la variable `x` à la variable `z` car *`expression1`* renvoie `true` :

```
x = 5;
y = 10;
z = (x < 6) ? x: y;
trace (z);// sortie : 5
```

-- (décrément)

Disponibilité

Flash Lite 1.0.

Utilisation

--expression

expression--

Opérandes

Aucun.

Description

Opérateur (arithmétique) ; un opérateur unaire de pré et post-décrémentation qui soustrait 1 de *expression*. La forme pré-décrémentale de l'opérateur (*--expression*) soustrait 1 de *expression* et renvoie le résultat sous forme de nombre. La forme post-décrémentale de l'opérateur (*expression--*) soustrait 1 de *expression* et renvoie la valeur initiale de *expression* (la valeur avant soustraction).

Exemple

Dans l'exemple suivant, la forme pré-décrémentale de l'opérateur décrémente `aWidth` pour obtenir 2 (`aWidth - 1 = 2`) et renvoie le résultat dans `bWidth` :

```
aWidth = 3;
bWidth = --aWidth;
// La valeur bWidth est égale à 2.
```

Dans l'exemple suivant, la forme post-décrémentale de l'opérateur décrémente `aWidth` pour obtenir 2 (`aWidth - 1 = 2`) et renvoie la valeur `aWidth` d'origine dans le résultat `bWidth` :

```
aWidth = 3;
bWidth = aWidth--;
// La valeur bWidth est égale à 3.
```

/ (division)

Disponibilité

Flash Lite 1.0.

Utilisation

expression1 / expression2

Opérandes

expression1, expression2 Nombres ou expressions renvoyant un nombre.

Description

Opérateur (arithmétique) ; divise *expression1* par *expression2*. Le résultat de l'opération de division est un nombre à virgule flottante comportant deux décimales.

Exemple

L'instruction suivante divise le nombre à virgule flottante 22,0 par 7,0 et affiche le résultat dans le panneau Sortie :

```
trace(22.0 / 7.0);
```

Le résultat est 3,1429 qui correspond à un nombre à virgule flottante.

/= (affectation de division)

Disponibilité

Flash Lite 1.0.

Utilisation

expression1 /= expression2

Opérandes

expression1, expression2 Nombres ou expressions renvoyant un nombre.

Description

Opérateur (affectation de composant arithmétique) ; affecte à *expression1* la valeur de *expression1 / expression2*. Par exemple, les deux instructions suivantes sont équivalentes :

```
x /= y
x = x / y
```

Exemple

L'exemple suivant utilise l'opérateur /= avec des variables et des nombres :

```
x = 10;
y = 2;
x /= y;
// L'expression x contient désormais la valeur 5.
```

. (point)

Disponibilité

Flash Lite 1.0.

Utilisation

instancename.variable

instancename.childinstance.variable

Opérandes

instancename Le nom d'occurrence d'un clip.

childinstance Une occurrence de clip qui est un enfant d'un autre clip ou qui est imbriquée dans ce dernier.

variable Une variable du scénario du nom d'occurrence de clip spécifié.

Description

Opérateur ; permet de naviguer au sein des hiérarchies de clips pour accéder aux clips imbriqués (enfants), aux variables ou aux propriétés.

Exemple

L'exemple suivant identifie la valeur actuelle de la variable `hairColor` dans le clip `person_mc` :

```
person_mc.hairColor
```

Cela correspond à la syntaxe de notation à barre oblique suivante :

```
/person_mc:hairColor
```

Voir aussi

/ (barre de fraction)

++ (incrément)

Disponibilité

Flash Lite 1.0.

Utilisation

++expression

expression++

Opérandes

Aucun.

Description

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

Exemple

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

```
i = 0;
while (i++ < 5){
  trace("this is execution " + i);
}
```

L'exemple suivant utilise ++ comme opérateur de pré-incrémentation :

```
a = "";
i = 0;
while (i < 10) {
  a = a add (++i) add ",";
}
trace(a);// sortie : 1,2,3,4,5,6,7,8,9,10,
```

Ce script donne le résultat suivant dans le panneau Sortie :

```
1,2,3,4,5,6,7,8,9,10,
```

L'exemple suivant utilise ++ comme opérateur de post-incrémentation :

```
a = "";
i = 0;
while (i < 10) {
  a = a add (i++) add ",";
}
trace(a);// sortie : 0,1,2,3,4,5,6,7,8,9,
```

Ce script donne le résultat suivant dans le panneau Sortie :

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

&& (AND logique)

Disponibilité

Flash Lite 1.0.

Utilisation

expression1 && *expression2*

Opérandes

expression1, expression2 Valeurs ou expressions booléennes converties en valeurs booléennes.

Description

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

Exemple

L'exemple suivant utilise l'opérateur `&&` pour vérifier si un joueur a remporté la partie. Les variables `turns` et `score` sont mises à jour lorsqu'un joueur prend un tour ou marque des points durant la partie. Le script suivant affiche « You Win the Game! » dans le panneau Sortie lorsque le joueur obtient un score de 75 ou plus en trois tours maximum.

```
turns = 2;
score = 77;
winner = (turns <= 3) && (score >= 75);
if (winner) {
  trace("You Win the Game!");
} else {
```

```
  trace("Try Again!");
}
```

L'exemple suivant décrit le test effectué pour vérifier s'il existe une position `x` imaginaire intermédiaire dans une plage :

```
xPos = 50;
if (xPos >= 20 && xPos <= 80) {
  trace ("the xPos is in between 20 and 80");
}
```

! (NOT logique)

Disponibilité

Flash Lite 1.0.

Utilisation

`!`*`expression`*

Opérandes

Aucun.

Description

Opérateur (logique) ; inverse la valeur booléenne d'une variable ou d'une expression. Si *`expression`* est une variable dont la valeur absolue ou convertie est `true`, la valeur de `!`*`expression`* est `false`. Si l'expression `x && y` renvoie `false`, l'expression `!(x && y)` renvoie `true`.

Les expressions suivantes illustrent le résultat de l'utilisation de l'opérateur `!` :

`!`*`true`* renvoie `false`

`!`*`false`* renvoie `true`

Exemple

Dans l'exemple suivant, la variable `happy` est définie sur `false`. La condition `if` évalue la condition `!happy`, et si cette dernière est `true`, la fonction `trace()` envoie une chaîne au panneau Sortie.

```
happy = false;
if (!happy) {
  trace("don't worry, be happy");
}
```

|| (OR logique)

Disponibilité

Flash Lite 1.0.

Utilisation

expression1 || *expression2*

Opérandes

expression1, expression2 Valeurs ou expressions booléennes converties en valeurs booléennes.

Description

Opérateur (logique) ; évalue *expression1* et *expression2*. Le résultat est `true` si l'une des expressions, voire les deux, renvoie(nt) `true`. Le résultat est `false` si et uniquement si les deux expressions renvoient `false`. Vous pouvez utiliser l'opérateur OR logique avec autant d'opérandes que nécessaire. Si l'un des opérandes renvoie `true`, le résultat est `true`.

Dans le cas d'expressions non booléennes, l'opérateur OR logique oblige Flash Lite à évaluer l'expression de gauche ; si elle peut être convertie en `true`, le résultat est `true`. Sinon, l'opérateur évalue l'expression de droite, et le résultat est la valeur de cette expression.

Exemple

Utilisation 1 : l'exemple suivant utilise l'opérateur `||` dans une instruction `if`. La deuxième expression renvoie `true`, par conséquent le résultat final est `true` :

```
theMinimum = 10;
theMaximum = 250;
start = false;
if (theMinimum > 25 || theMaximum > 200 || start){
  trace("the logical OR test passed");
}
```

% (modulo)

Flash Lite 1.0.

Utilisation

expression1 % *expression2*

Opérandes

expression1, *expression2* Nombres ou expressions renvoyant un nombre.

Description

Opérateur (arithmétique) ; calcule le reste d'*expression1* divisée par *expression2*. Si un opérande *expression* n'est pas numérique, l'opérateur modulo tente de le convertir en nombre. L'expression peut être un nombre ou une chaîne à convertir en valeur numérique.

Selon que vous choisissez Flash Lite 1.0 ou 1.1, le compilateur Flash développe l'opérateur % dans le fichier SWF publié au moyen de la formule suivante :

expression1 - int(*expression1*/*expression2*) * *expression2*

Les performances de cette approximation ne seront peut-être pas aussi rapides ou aussi efficaces que celles des versions de Flash Player qui prennent en charge de façon native l'opérateur modulo.

Exemple

Le code suivant illustre une utilisation numérique de l'opérateur (%) modulo :

```
trace (12 % 5);// sortie : 2
trace (4.3 % 2.1);// sortie : 0.0999...
```

%= (affectation modulo)

Disponibilité

Flash Lite 1.0.

Utilisation

expression1 %= *expression2*

Opérandes

expression1, *expression2* Nombres ou expressions renvoyant un nombre.

Description

Opérateur (affectation de composant arithmétique) ; affecte à *expression1* la valeur d'*expression1* `%` *expression2*. Par exemple, les deux expressions suivantes sont équivalentes :

```
x %= y
x = x % y
```

Exemple

L'exemple suivant affecte la valeur 4 à la variable `x` :

```
x = 14;
y = 5;
trace(x %= y);// sortie : 4
```

Voir aussi

`% (modulo)`

*= (affectation de multiplication)

Disponibilité

Flash Lite 1.0.

Utilisation

expression1 `*=` *expression2*

Opérandes

expression1, *expression2* Nombres ou expressions renvoyant un nombre.

Description

Opérateur (affectation de composant arithmétique) ; affecte à *expression1* la valeur de *expression1* `*` *expression2*.

Par exemple, les deux expressions suivantes sont équivalentes :

```
x *= y
x = x * y
```

Exemple

Utilisation 1 : l'exemple suivant affecte la valeur 50 à la variable `x` :

```
x = 5;
y = 10;
trace (x *= y);// sortie : 50
```

Utilisation 2 : les deuxième et troisième lignes de l'exemple suivant calculent les expressions situées à droite du signe égal (=) et affectent les résultats à `x` et `y` :

```
i = 5;
x = 4 - 6;
y = i + 2;
trace(x *= y);// sortie : -14
```

* (produit)

Disponibilité

Flash Lite 1.0.

Utilisation

`expression1 * expression2`

Opérandes

`expression1, expression2` Expressions numériques.

Description

Opérateur (arithmétique) ; multiplie deux expressions numériques. Lorsque les deux expressions sont des entiers, le produit est un entier. Lorsque l'une ou les deux expressions sont des nombres à virgule flottante, le produit est un nombre à virgule flottante.

Exemple

Utilisation 1 : l'instruction suivante multiplie les entiers 2 et 3 :

```
2 * 3
```

Le résultat est 6 qui correspond à un entier.

Utilisation 2 : l'instruction suivante multiplie les nombres à virgule flottante 2,0 et 3,1416 :

```
2.0 * 3.1416
```

Le résultat est 6,2832 qui correspond à un nombre à virgule flottante.

+ (addition numérique)

Disponibilité

Flash Lite 1.0.

Utilisation

expression1 + *expression2*

Opérandes

expression1, *expression2* Nombres.

Description

Opérateur ; additionne des expressions numériques. Le + est un opérateur numérique uniquement ; il ne peut être utilisé pour la concaténation de chaîne.

Si les deux expressions sont des entiers, la somme est un entier. Si l'une ou les deux expressions sont des nombres à virgule flottante, la somme est un nombre à virgule flottante.

Exemple

L'exemple suivant additionne les entiers 2 et 3, puis affiche l'entier obtenu, 5, dans le panneau Sortie :

```
trace (2 + 3);
```

L'exemple suivant additionne les entiers 2,5 et 3,25, puis affiche le nombre à virgule flottante obtenu (5,75) dans le panneau Sortie :

```
trace (2.5 + 3.25);
```

Voir aussi

add (concaténation de chaîne)

== (égalité numérique)

Disponibilité

Flash Lite 1.0.

Utilisation

expression1 == *expression2*

Opérandes

expression1, *expression2* Nombres, valeurs booléennes ou variables.

Description

Opérateur (comparaison) ; teste l'égalité ; l'inverse de l'opérateur <>. Si *expression1* est égale à *expression2*, le résultat est `true`. Comme pour l'opérateur <>, la définition de l'*égalité* dépend des types de données comparés :

- Les nombres et les valeurs booléennes sont comparés en fonction de leur valeur.
- Les variables sont comparées en fonction de leur référence.

Exemple

Les exemples suivants illustrent les valeurs renvoyées `true` et `false` :

```
trees = 7;
bushes = "7";
shrubs = "seven";

trace (trees == "7");// sortie : 1(true)
trace (trees == bushes);// sortie : 1(true)
trace (trees == shrubs);// sortie : 0(false)
```

Voir aussi

eq (égalité de chaîne)

> (numérique supérieur à)

Disponibilité

Flash Lite 1.0.

Utilisation

expression1 > *expression2*

Opérandes

expression1, expression2 Nombres ou expressions renvoyant un nombre.

Opérateur (comparaison) ; compare deux expressions et détermine si *expression1* est supérieure à *expression2* ; le cas échéant, cet opérateur renvoie `true`. Si *expression1* est inférieure ou égale à *expression2*, l'opérateur renvoie `false`.

Exemple

Les exemples suivants illustrent les résultats `true` et `false` des comparaisons numériques :

```
trace(3.14 > 2);// sortie : 1(true)
trace(1 > 2);// sortie : 0(false)
```

Voir aussi

gt (chaîne supérieure à)

>= (numérique supérieur ou égal à)

Disponibilité

Flash Lite 1.0.

Utilisation

expression1 >= *expression2*

Opérandes

expression1, *expression2* Entiers ou nombres à virgule flottante.

Description

Opérateur (comparaison) ; compare deux expressions et détermine si *expression1* est supérieure à *expression2* (`true`) ou si *expression1* est inférieure à *expression2* (`false`).

Exemple

Les exemples suivants illustrent les résultats `true` et `false` :

```
trace(3.14 >= 2);// sortie : 1(true)
trace(3.14 >= 4);// sortie : 0(false)
```

Voir aussi

ge (chaîne supérieure ou égale à)

<> (inégalité numérique)

Disponibilité

Flash Lite 1.0.

Utilisation

expression1 <> *expression2*

Opérandes

expression1, *expression2* Nombres, valeurs booléennes ou variables.

Description

Opérateur (comparaison) ; teste l'inégalité ; l'inverse de l'opérateur d'égalité (==). Si *expression1* est égal à *expression2*, le résultat est `false`. Comme pour l'opérateur d'égalité (==), la définition de l'*égalité* dépend des types de données comparés :

- Les nombres et les valeurs booléennes sont comparés en fonction de leur valeur.
- Les variables sont comparées en fonction de leur référence.

Exemple

Les exemples suivants illustrent les valeurs renvoyées `true` et `false` :

```
trees = 7;
B = "7";

trace(trees <> 3);// sortie : 1(true)
trace(trees <> B);// sortie : 0(false)
```

Voir aussi

`ne (inégalité de chaîne)`

< (numérique inférieur à)

Disponibilité

Flash Lite 1.0.

Utilisation

`expression1 < expression2`

Opérandes

`expression1, expression2` Nombres.

Description

Opérateur (comparaison) ; compare deux expressions et détermine si *`expression1`* est inférieure à *`expression2`* ; le cas échéant, cet opérateur renvoie `true`. Si *`expression1`* est supérieure ou égale à*`expression2`*, l'opérateur renvoie `false`. L'opérateur < (inférieur à) est un opérateur numérique.

Exemple

Les exemples suivants illustrent les résultats `true` et `false` des comparaisons numériques et de chaîne :

```
trace (3 < 10);// sortie : 1(true)

trace (10 < 3);// sortie : 0(false)
```

Voir aussi

`lt (chaîne inférieure à)`

<= (numérique inférieur ou égal à)

Flash Lite 1.0.

Utilisation

expression1 <= *expression2*

Opérandes

expression1, *expression2* Nombres.

Description

Opérateur (comparaison) ; compare deux expressions et détermine si *expression1* est inférieure ou égale à *expression2*. Le cas échéant, l'opérateur renvoie `true` ; `false` dans le cas contraire. Cet opérateur est utilisé uniquement pour la comparaison numérique.

Exemple

Les exemples suivants illustrent les résultats `true` et `false` des comparaisons numériques :

```
trace(5 <= 10);// sortie : 1(true)
trace(2 <= 2);// sortie : 1(true)
trace (10 <= 3);// sortie : 0(false)
```

Voir aussi

le (chaîne inférieure ou égale à)

() (parenthèses)

Disponibilité

Flash Lite 1.0.

Utilisation

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

expression1, *expression2* Nombres, chaînes, variables ou texte.

parameter1,..., *parameterN* Une série de paramètres à exécuter avant de transmettre les résultats sous forme de paramètres à la fonction située en-dehors des parenthèses.

Description

Opérateur ; regroupe un ou plusieurs paramètres, évalue les expressions de façon séquentielle ou entoure un ou plusieurs paramètres et les transmet en tant que paramètres à une fonction en dehors des parenthèses.

Utilisation 1 : contrôle l'ordre suivant lequel les opérateurs s'exécutent dans l'expression. Les parenthèses remplacent la séquence normale et entraînent l'évaluation des expressions entre parenthèses en premier. Lorsque les parenthèses sont imbriquées, le contenu entre les parenthèses de plus bas niveau est évalué en premier.

Utilisation 2 : évalue une série d'expressions, séparées par des virgules, dans la séquence et renvoie le résultat de l'expression finale.

Exemple

Utilisation 1 : les instructions suivantes illustrent l'utilisation des parenthèses pour contrôler l'ordre d'exécution des expressions (la valeur de chaque expression apparaît dans le panneau Sortie) :

```
trace((2 + 3) * (4 + 5)); // affiche 45
trace(2 + (3 * (4 + 5))); // affiche 29
trace(2 + (3 * 4) + 5); // affiche 19
```

Utilisation 2 : les instructions suivantes illustrent l'utilisation des parenthèses pour contrôler l'ordre d'exécution des expressions (la valeur de chaque expression est écrite dans le fichier journal) :

```
trace((2 + 3) * (4 + 5)); // écrit 45
trace(2 + (3 * (4 + 5))); // écrit 29
trace(2 + (3 * 4) + 5); // écrit 19
```

" " (séparateur de chaîne)

Disponibilité

Flash Lite 1.0.

Utilisation

"*text*"

Opérandes

text Aucun ou plusieurs caractères.

Description

Séparateur de chaîne ; lorsqu'ils entourent une séquence de zéro ou plusieurs caractères, les guillemets indiquent que ces caractères ont une valeur littérale et doivent être traités en tant que *chaîne* et non pas en tant que variable, valeur numérique ou tout autre élément ActionScript.

Exemple

L'exemple suivant utilise les guillemets pour indiquer que la valeur de la variable `yourGuess` correspond à la chaîne littérale `"Prince Edward Island"` et non au nom d'une variable. La valeur de `province` est une variable, et non un littéral ; pour déterminer la valeur de `province`, la valeur de `yourGuess` doit être déterminée.

```
yourGuess = "Prince Edward Island";

on(release){
  province = yourGuess;
  trace(province);// sortie : Prince Edward Island
}
```

eq (égalité de chaîne)

Disponibilité

Flash Lite 1.0.

Utilisation

expression1 eq *expression2*

Opérandes

expression1, expression2 Nombres, chaînes ou variables.

Description

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

Exemple

Les exemples suivants illustrent les résultats `true` et `false` :

```
word = "persons";
figure = "55";

trace("persons" eq "people");// sortie : 0(false)
trace("persons" eq word);// sortie : 1(true)
trace(figure eq 50 + 5);// sortie : 1(true)
trace(55.0 eq 55);// sortie : 1(true)
```

Voir aussi

== (égalité numérique)

gt (chaîne supérieure à)

Disponibilité

Flash Lite 1.0.

Utilisation

expression1 gt *expression2*

Opérandes

expression1, expression2 Nombres, chaînes ou variables.

Description

Opérateur (comparaison) ; compare la chaîne représentant *expression1* à la chaîne représentant *expression2* et renvoie `true` si *expression1* est supérieure à *expression2* ; `false` dans le cas contraire. Les chaînes sont comparées en fonction de leur ordre alphabétique ; les chiffres précèdent les lettres et les lettres majuscules précèdent les lettres minuscules.

Exemple

Les exemples suivants illustrent les résultats `true` et `false` :

```
animals = "cats";
breeds = 7;

trace ("persons" gt "people");// sortie : 1(true)
trace ("cats" gt "cattle");// sortie : 0(false)
trace (animals gt "cats");// sortie : 0(false)
trace (animals gt "Cats");// sortie : 1(true)
trace (breeds gt "5");// sortie : 1(true)
trace (breeds gt 7);// sortie : 0(false)
```

Voir aussi

> (numérique supérieur à)

ge (chaîne supérieure ou égale à)

Disponibilité

Flash Lite 1.0.

Utilisation

expression1 ge *expression2*

Opérandes

expression1, *expression2* Nombres, chaînes ou variables.

Description

Opérateur (comparaison) ; compare la chaîne représentant *expression1* à la chaîne représentant *expression2* et renvoie `true` si *expression1* est supérieure ou égale à *expression2* ; `false` dans le cas contraire. Les chaînes sont comparées en fonction de leur ordre alphabétique ; les chiffres précèdent les lettres et les lettres majuscules précèdent les lettres minuscules.

Exemple

Les exemples suivants illustrent les résultats `true` et `false` :

```
animals = "cats";
breeds = 7;

trace ("cats" ge "cattle");// sortie : 0(false)
trace (animals ge "cats");// sortie : 1(true)
trace ("persons" ge "people");// sortie : 1(true)
trace (animals ge "Cats");// sortie : 1(true)
trace (breeds ge "5");// sortie : 1(true)
trace (breeds ge 7);// sortie : 1(true)
```

Voir aussi

>= (numérique supérieur ou égal à)

ne (inégalité de chaîne)

Disponibilité

Flash Lite 1.0.

Utilisation

expression1 ne *expression2*

Opérandes

expression1, expression2 Nombres, chaînes ou variables.

Description

Opérateur (comparaison) ; compare la chaîne représentant *expression1* à la chaîne représentant *expression2* et renvoie `true` si *expression1* n'est pas égale à *expression2* ; `false` dans le cas contraire.

Exemple

Les exemples suivants illustrent les résultats `true` et `false` :

```
word = "persons";
figure = "55";

trace ("persons" ne "people");// sortie : 1(true)
trace ("persons" ne word);// sortie : 0(false)
trace (figure ne 50 + 5);// sortie : 0(false)
trace (55.0 ne 55); // sortie : 0(false)
```

Voir aussi

<> (inégalité numérique)

lt (chaîne inférieure à)

Disponibilité

Flash Lite 1.0.

Utilisation

expression1 lt *expression2*

Opérandes

expression1, *expression2* Nombres, chaînes ou variables.

Description

Opérateur (comparaison) ; compare la chaîne représentant *expression1* à la chaîne représentant *expression2* et renvoie `true` si *expression1* est inférieure à *expression2* ; `false` dans le cas contraire. Les chaînes sont comparées en fonction de leur ordre alphabétique ; les chiffres précèdent les lettres et les lettres majuscules précèdent les lettres minuscules.

Exemple

Les exemples suivants montrent le résultat de différentes comparaisons de chaînes. Dans la dernière ligne, notez que `lt` ne renvoie pas d'erreur lorsque vous comparez une chaîne à un entier puisque la syntaxe de ActionScript 1.0 tente de convertir le type de données entier en chaîne et renvoie `false`.

```
animals = "cats";
breeds = 7;

trace ("persons" lt "people");// sortie : 0(false)
trace ("cats" lt "cattle");// sortie : 1(true)
trace (animals lt "cats");// sortie : 0(false)
trace (animals lt "Cats");// sortie : 0(false)
trace (breeds lt "5");// sortie : 0(false)
trace (breeds lt 7);// sortie : 0(false)
```

Voir aussi

`< (numérique inférieur à)`

le (chaîne inférieure ou égale à)

Disponibilité

Flash Lite 1.0.

Utilisation

expression1 `le` *expression2*

Opérandes

expression1, expression2 Nombres, chaînes ou variables.

Description

Opérateur (comparaison) ; compare la chaîne représentant *expression1* à la chaîne représentant *expression2* et renvoie `true` si *expression1* est inférieure ou égale à *expression2* ; `false` dans le cas contraire. Les chaînes sont comparées en fonction de leur ordre alphabétique ; les chiffres précèdent les lettres et les lettres majuscules précèdent les lettres minuscules.

Exemple

Les exemples suivants montrent le résultat de différentes comparaisons de chaînes :

```
animals = "cats";
breeds = 7;

trace ("persons" le "people");// sortie : 0(false)
trace ("cats" le "cattle");// sortie : 1(true)
trace (animals le "cats");// sortie : 1(true)
trace (animals le "Cats");// sortie : 0(false)
trace (breeds le "5");// sortie : 0(false)
trace (breeds le 7);// sortie : 1(true)
```

Voir aussi

<= (numérique inférieur ou égal à)

- (soustraction)

Disponibilité

Flash Lite 1.0.

Utilisation

(Négation) *-expression*

(Soustraction) *expression1 - expression2*

Opérandes

expression1, expression2 Nombres ou expressions renvoyant un nombre.

Description

Opérateur (arithmétique) ; utilisé pour la négation ou la soustraction.

Utilisation 1 : lorsque cet opérateur est utilisé pour la négation, il inverse le signe de l'expression numérique.

Utilisation 2 : lorsqu'il est utilisé pour la soustraction, il effectue une soustraction arithmétique sur deux expressions numériques, en soustrayant *expression2* de *expression1*. Lorsque les deux expressions sont des entiers, la différence est un entier. Lorsque l'une ou les deux expressions sont des nombres à virgule flottante, la différence est un nombre à virgule flottante.

Exemple

Utilisation 1 : l'instruction suivante inverse le signe de l'expression 2 + 3 .

```
trace(-(2 + 3));
// sortie : -5.
```

Utilisation 2 : l'instruction suivante soustrait l'entier 2 de l'entier 5 .

```
trace(5 - 2);
// sortie : 3.
```

Le résultat est 3 qui correspond à un entier.

Utilisation 3 : l'instruction suivante soustrait le nombre à virgule flottante 1,5 du nombre à virgule flottante 3,25 .

```
trace(3.25 - 1.5);
// sortie : 1.75.
```

Le résultat est 1,75 qui correspond à un nombre à virgule flottante.

-= (affectation de soustraction)

Disponibilité

Flash Lite 1.0.

Utilisation

expression1 -= expression2

Opérandes

expression1, expression2 Nombres ou expressions renvoyant un nombre.

Description

Opérateur (affectation de composant arithmétique) ; affecte à *expression1* la valeur de *expression1 - expression2*. Aucune valeur n'est renvoyée.

Par exemple, les deux instructions suivantes sont équivalentes :

```
x -= y;
x = x - y;
```

Les expressions de type String doivent être converties en nombres. Sinon, -1 est renvoyé.

Exemple

Utilisation 1 : l'exemple suivant utilise l'opérateur -= pour soustraire 10 de 5 et affecte le résultat à la variable x .

```
x = 2;
y = 3;
x -= y
trace(x);// sortie : -1
```

Utilisation 2 : l'exemple suivant indique comment convertir des chaînes en nombres .

```
x = "2";
y = "5";
x -= y;
trace(x);// sortie : -3
```

CHAPITRE 5

Eléments de langage spécifiques de Flash Lite

5

Cette section présente les variables et fonctionnalités de la plate-forme reconnues par Macromedia Flash Lite 1.1 ainsi que les commandes Flash Lite que vous pouvez exécuter à l'aide des fonctions `fscommand()` et `fscommand2()`. Les fonctionnalités décrites dans cette section sont spécifiques à Flash Lite.

Le contenu de cette section est présenté dans le tableau suivant.

Elément de langage	Description
`_capCompoundSound`	Indique si Flash Lite peut traiter un son composé.
`_capEmail`	Indique si le client Flash Lite peut envoyer des messages électroniques à l'aide de la commande ActionScript `GetURL()`.
`_capLoadData`	Indique si l'application hôte peut charger de façon dynamique des données supplémentaires en appelant les fonctions `loadMovie()`, `loadMovieNum()`, `loadVariables()` et `loadVariablesNum()`.
`_capMFi`	Indique si le périphérique peut lire des données audio au format MFi (Melody Format for i-mode)
`_capMIDI`	Indique si le périphérique peut lire des données audio au format audio MIDI (Musical Instrument Digital Interface).
`_capMMS`	Indique si Flash Lite peut envoyer des messages MMS (Multimedia Messaging Service) à l'aide de la commande ActionScript `GetURL()`.
`_capMP3`	Indique si le périphérique peut lire des données audio au format audio MP3 (MPEG Audio Layer 3).
`_capSMAF`	Indique si le périphérique peut lire des fichiers multimédia au format SMAF (Synthetic music Mobile Application Format).
`_capSMS`	Indique si Flash Lite peut envoyer des messages SMS (Short Message Service) à l'aide de la commande ActionScript `GetURL()`.

Elément de langage	Description
`_capStreamSound`	Indique si le périphérique peut lire du son en continu (synchronisé).
`_cap4WayKeyAS`	Indique si Flash Lite exécute des expressions ActionScript associées à des gestionnaires d'événements de touche, eux-mêmes associés aux flèches Droite, Gauche, Haut et Bas.
`$version`	Contient le numéro de version de Flash Lite.
`fscommand()`	Une fonction permettant d'exécuter la commande `Launch` (voir entrée suivante).
`Launch`	(La seule commande prise en charge pour `fscommand()`) Permet au fichier SWF de communiquer avec Flash Lite ou bien l'environnement hôte, comme par exemple le système d'exploitation du téléphone ou périphérique portable.
`fscommand2()`	Une fonction permettant d'exécuter les autres commandes de ce tableau, à l'exception de `fscommand()`.
`Escape`	Code une chaîne arbitraire en un format garantissant un transfert réseau sécurisé.
`FullScreen`	Définit la taille de la zone d'affichage à utiliser pour le rendu.
`GetBatteryLevel`	Renvoie le niveau de batterie actuel.
`GetDateDay`	Renvoie le jour de la date courante sous forme de valeur numérique.
`GetDateMonth`	Renvoie le mois de la date courante sous forme de valeur numérique.
`GetDateWeekday`	Renvoie le chiffre correspondant au jour de la date courante sous forme de valeur numérique.
`GetDateYear`	Renvoie une valeur numérique à 4 chiffres représentant l'année de la date courante.
`GetDevice`	Définit un paramètre qui identifie le périphérique sur lequel s'exécute Flash Lite.
`GetDeviceID`	Définit un paramètre qui représente l'identificateur unique du périphérique (par exemple, le numéro de série).
`GetFreePlayerMemory`	Renvoie la quantité de mémoire de tas, en kilo-Octets, actuellement disponible pour Flash Lite.
`GetLanguage`	Définit un paramètre qui identifie la langue actuellement utilisée par le périphérique.

Elément de langage	Description
GetLocaleLongDate	Définit un paramètre en chaîne représentant la date courante, au format long, formatée en fonction des paramètres régionaux spécifiés.
GetLocaleShortDate	Définit un paramètre en chaîne représentant la date courante, au format abrégé, formatée en fonction des paramètres régionaux spécifiés.
GetLocaleTime	Définit un paramètre en chaîne représentant l'heure courante, formatée en fonction des paramètres régionaux spécifiés.
GetMaxBatteryLevel	Renvoie le niveau de batterie maximum du périphérique.
GetMaxSignalLevel	Renvoie le niveau d'intensité maximal du signal.
GetMaxVolumeLevel	Renvoie le niveau de volume maximum du périphérique sous forme de valeur numérique.
GetNetworkConnectStatus	Renvoie une valeur indiquant l'état actuel de la connexion réseau.
GetNetworkName	Définit un paramètre en nom du réseau actif.
GetNetworkRequestStatus	Renvoie une valeur indiquant l'état de la requête HTTP la plus récente.
GetNetworkStatus	Renvoie une valeur indiquant l'état du réseau du téléphone (autrement dit, si un réseau est enregistré et si le téléphone est doté de la fonctionnalité d'itinérance).
GetPlatform	Définit un paramètre qui identifie la plate-forme actuelle et donne ainsi une brève description de la classe du périphérique. Pour les périphériques équipés de systèmes d'exploitation ouverts, cet identificateur correspond généralement au nom et à la version du système d'exploitation.
GetPowerSource	Renvoie une valeur qui indique si l'alimentation provient d'une batterie ou d'une source d'alimentation externe.
GetSignalLevel	Renvoie l'intensité courante du signal sous forme de valeur numérique.
GetTimeHours	Renvoie l'heure courante du jour, en fonction d'une journée de 24 heures, sous forme de valeur numérique.
GetTimeMinutes	Renvoie les minutes de l'heure courante du jour actuel sous forme de valeur numérique.
GetTimeSeconds	Renvoie les secondes de l'heure courante du jour actuel sous forme de valeur numérique.
GetTimeZoneOffset	Définit un paramètre en nombre de minutes entre le fuseau horaire local et l'heure universelle (UTC).

Elément de langage	Description
GetTotalPlayerMemory	Renvoie la quantité totale de mémoire de tas, en kilo-Octets, allouée à Flash Lite.
GetVolumeLevel	Renvoie le niveau de volume courant du périphérique sous forme de valeur numérique.
Quit	Oblige le lecteur Flash Lite à arrêter la tête de lecture et à quitter l'application.
ResetSoftKeys	Réinitialise les touches programmables sur leurs paramètres d'origine.
SetInputTextType	Spécifie le mode dans lequel ouvrir le champ de texte de saisie.
SetQuality	Définit la qualité de rendu de l'animation.
SetSoftKeys	Remappe les touches programmables Gauche et Droite du périphérique, sous réserve qu'elles soient accessibles et remappables.
StartVibrate	Lance la fonction vibration du téléphone.
StopVibrate	Arrête la vibration en cours, le cas échéant.
Unescape	Décode une chaîne arbitraire qui a été codée pour garantir un transfert sécurisé sur le réseau et revenir à sa forme normale non codée.

Fonctionnalités

Cette section décrit les fonctionnalités et variables de la plate-forme reconnues par Macromedia Flash Lite 1.1. Les entrées sont classées par ordre alphabétique sans tenir compte des traits de soulignement de préfixe.

_capCompoundSound

Disponibilité

Flash Lite 1.1.

Utilisation

```
_capCompoundSound
```

Description

Variable numérique ; indique si Flash Lite peut traiter des données audio composées. Le cas échéant, cette variable est définie et sa valeur est 1 ; dans le cas contraire, cette variable n'est pas définie.

Par exemple, un fichier Flash simple peut contenir le même son représenté aux formats MIDI et MFi. Le lecteur lira alors les données au format approprié en fonction de celui pris en charge par le périphérique. Cette variable définit si le lecteur Flash Lite prend en charge cette fonctionnalité sur le combiné actuel.

Dans l'exemple suivant, `useCompoundSound` est définie sur 1 dans Flash Lite 1.1, mais n'est pas définie dans Flash Lite 1.0 :

```
useCompoundSound = _capCompoundSound;

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

_capEmail

Disponibilité

Flash Lite 1.1.

Utilisation

`_capEmail`

Description

Variable numérique ; indique si le client Flash Lite peut envoyer des messages électroniques à l'aide de la commande ActionScript `GetURL()`. Le cas échéant, cette variable est définie et sa valeur est 1 ; dans le cas contraire, cette variable n'est pas définie.

Exemple

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

```
canEmail = _capEmail;

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

_capLoadData

Disponibilité

Flash Lite 1.1.

Utilisation

```
_capLoadData
```

Description

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

Exemple

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

```
canLoad = _capLoadData;

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

_capMFi

Disponibilité

Flash Lite 1.1.

Utilisation

```
_capMFi
```

Description

Variable numérique ; indique si le périphérique peut lire des données audio au format audio MFi (Melody Format for i-mode). Le cas échéant, cette variable est définie et sa valeur est 1 ; dans le cas contraire, cette variable n'est pas définie.

Exemple

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

```
canMFi = _capMFi;
```

```
if (canMFi == 1) {
  // envoyer des boutons movieclip à une image contenant des boutons
  déclenchant des sons d'événements
  tellTarget("buttons") {
    gotoAndPlay(2);
  }
}
```

_capMIDI

Disponibilité

Flash Lite 1.1.

Utilisation

```
_capMIDI
```

Description

Variable numérique ; indique si le périphérique peut lire des données audio au format audio MIDI (Musical Instrument Digital Interface). Le cas échéant, cette variable est définie et sa valeur est 1 ; dans le cas contraire, cette variable n'est pas définie.

Exemple

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

```
canMIDI = _capMIDI;

if (canMIDI == 1) {
  // envoyer des boutons movieclip à une image contenant des boutons
  déclenchant des sons d'événements
  tellTarget("buttons") {
    gotoAndPlay(2);
  }
}
```

_capMMS

Disponibilité

Flash Lite 1.1.

Utilisation

```
_capMMS
```

Description

Variable numérique ; indique si Flash Lite peut envoyer des messages MMS (Multimedia Messaging Service) à l'aide de la commande ActionScript `GetURL()`. Le cas échéant, cette variable est définie et sa valeur est 1 ; dans le cas contraire, cette variable n'est pas définie.

Exemple

L'exemple suivant définit `canMMS` sur 1 dans Flash Lite 1.1, mais la laisse inchangée dans Flash Lite 1.0 (cependant, tous les téléphones Flash Lite 1.1 ne peuvent pas envoyer de messages MMS, par conséquent ce code dépend toujours du téléphone) :

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

_capMP3

Disponibilité

Flash Lite 1.1.

Utilisation

```
_capMP3
```

Description

Variable numérique ; indique si le périphérique peut lire des données audio au format audio MP3 (MPEG Audio Layer 3). Le cas échéant, cette variable est définie et sa valeur est 1 ; dans le cas contraire, cette variable n'est pas définie.

Exemple

Si le périphérique peut lire des données audio MP3, l'exemple suivant définit `canMP3` sur 1 :

```
canMP3 = _capMP3;
if (canMP3 == 1) {
  tellTarget("soundClip") {
    gotoAndPlay(2);
  }
}
```

_capSMAF

Disponibilité

Flash Lite 1.1.

Utilisation

`_capSMAF`

Description

Variable numérique ; indique si le périphérique peut lire des fichiers multimédias au format SMAF (Synthetic music Mobile Application Format). Le cas échéant, cette variable est définie et sa valeur est 1 ; dans le cas contraire, cette variable n'est pas définie.

Exemple

L'exemple suivant définit `canSMAF` sur 1 dans Flash Lite 1.1, mais la laisse inchangée dans Flash Lite 1.0 (cependant, tous les téléphones Flash Lite 1.1 ne peuvent pas envoyer de messages SMAF, par conséquent ce code dépend toujours du téléphone) :

```
canSMAF = _capSMAF;

if (canSMAF) {
  // envoyer des boutons movieclip à une image contenant des boutons
  déclenchant des sons d'événements
  tellTarget("buttons") {
    gotoAndPlay(2);
  }
}
```

_capSMS

Disponibilité

Flash Lite 1.1.

Utilisation

`_capSMS`

Description

Variable numérique ; indique si Flash Lite peut envoyer des messages SMS (*Short Message Service*) à l'aide de la commande ActionScript `GetURL()`. Le cas échéant, cette variable est définie et sa valeur est 1 ; dans le cas contraire, cette variable n'est pas définie.

Exemple

L'exemple suivant définit `canSMS` sur 1 dans Flash Lite 1.1, mais la laisse inchangée dans Flash Lite 1.0 (cependant, tous les téléphones Flash Lite 1.1 ne peuvent pas envoyer de messages SMS, par conséquent ce code dépend toujours du téléphone) :

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

_capStreamSound

Disponibilité

Flash Lite 1.1.

Utilisation

```
_capStreamSound
```

Description

Variable numérique ; indique si le périphérique peut lire du son en continu (synchronisé). Le cas échéant, cette variable est définie et sa valeur est 1 ; dans le cas contraire, cette variable n'est pas définie.

Exemple

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

```
on(press) {
  canStreamSound = _capStreamSound;
  if (canStreamSound) {
    // lire un son en continu dans un clip à l'aide de ce bouton
    tellTarget("music") {
      gotoAndPlay(2);
    }
  }
}
```

_cap4WayKeyAS

Disponibilité

Flash Lite 1.1.

Utilisation

```
_cap4WayKeyAS
```

Description

Variable numérique ; indique si Flash Lite exécute des expressions ActionScript associées à des gestionnaires d'événements de touche, eux-mêmes associés aux flèches Droite, Gauche, Haut et Bas. Cette variable est définie et sa valeur est 1 seulement si l'application hôte utilise le mode de navigation quadridirectionnelle pour passer d'une commande Flash à une autre (boutons et champs de texte de saisie). Dans le cas contraire, elle n'est pas définie.

Lorsque l'utilisateur appuie sur l'une des quatre touches de direction, si la valeur de cette variable est 1, Flash Lite recherche d'abord le gestionnaire de cette touche. S'il n'en trouve aucun, la navigation à l'aide des commandes Flash est utilisée. En revanche, en cas de détection d'un gestionnaire d'événements, aucune action de navigation n'est effectuée pour cette touche. Par exemple, si un gestionnaire de touches est détecté pour la flèche Bas, l'utilisateur ne peut pas naviguer.

Exemple

L'exemple suivant définit `canUse4Way` sur 1 dans Flash Lite 1.1, mais la laisse inchangée dans Flash Lite 1.0 (cependant, tous les téléphones Flash Lite 1.1 ne prennent pas en charge le mode de navigation quadridirectionnelle, par conséquent ce code dépend toujours du téléphone) :

```
canUse4Way = _cap4WayKeyAS;
 if (canUse4Way == 1) {
  msg = "Use your directional joypad to navigate this application";
 } else {
  msg = "Please use the 2 key to scroll up, the 6 key to scroll right, the
  8 key to scroll down, and the 4 key to scroll left.";
 }
```

$version

Disponibilité

Flash Lite 1.1.

Utilisation

```
$version
```

Description

Variable de chaîne ; contient le numéro de version de Flash Lite. Elle contient un numéro majeur, un numéro mineur, un numéro de construction et un numéro de construction interne, généralement 0 dans toutes les versions officielles.

Le numéro majeur rapporté pour tous les produits Flash Lite 1.x est 5. Flash Lite 1.0 possède le numéro mineur 1 ; Flash Lite 1.1 possède le numéro mineur 2.

Exemple

L'exemple suivant donne la valeur « 5, 2, 12, 0 » à `myVersion` :

```
myVersion = $version;
```

fscommand()

Disponibilité

Flash Lite 1.1.

Utilisation

```
status = fscommand("Launch", "application-path, arg1, arg2,..., argn")
```

Paramètres

`"Launch"` Le spécificateur de commandes. La commande `Launch` est la seule commande nécessitant l'exécution de la fonction `fscommand()`.

"*application-path, arg1, arg2, . . . , argn*" Le nom de l'application démarrée et ses paramètres, séparés par des virgules.

Description

Fonction ; permet au fichier SWF de communiquer avec Flash Lite ou bien l'environnement hôte, comme par exemple le système d'exploitation du téléphone ou périphérique portable.

Voir aussi

`fscommand2()`

Launch

Disponibilité

Flash Lite 1.1.

Utilisation

```
status = fscommand("Launch", "application-path, arg1, arg2,..., argn")
```

Paramètres

`"Launch"` Le spécificateur de commandes. Dans Flash Lite, la fonction `fscommand()` est utilisée uniquement pour exécuter la commande `Launch`.

"*application-path*, *arg1*, *arg2*, . . . , *argn*" Le nom de l'application démarrée et ses paramètres, séparés par des virgules.

Description

Commande exécutée à l'aide de la fonction `fscommand()` ; lance une autre application sur le périphérique. Le nom de l'application lancée et les paramètres qui lui sont associés sont transmis sous forme d'argument unique.

REMARQUE	Cette fonctionnalité dépend du système d'exploitation.

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

Exemple

L'exemple suivant ouvre wap.yahoo.com dans les services/le navigateur Web sur les téléphones de la gamme Series 60 :

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

Voir aussi

`fscommand2()`

fscommand2()

Disponibilité

Flash Lite 1.1.

Utilisation

```
returnValue = fscommand2(command [, expression1 ... expressionN])
```

Paramètres

command Une chaîne transmise à l'application hôte ou une commande passée à Flash Lite.

parameter1...parameterN Une liste délimitée par des virgules des chaînes transmises sous forme de paramètres à la commande spécifiée par *command*.

Description

Fonction ; permet au fichier SWF de communiquer avec Flash Lite ou bien l'environnement hôte, comme par exemple le système d'exploitation du téléphone ou périphérique portable. La valeur renvoyée par `fscommand2()` dépend de la commande spécifique.

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

- La fonction `fscommand2()` peut contenir un nombre arbitraire d'arguments.
- Flash Lite exécute la fonction `fscommand2()` immédiatement, alors que la fonction `fscommand()` est exécutée à la fin de l'image en cours de traitement.
- La fonction `fscommand2()` renvoie une valeur permettant de rapporter si la commande a réussi, échoué ou bien son résultat.

Les chaînes et expressions transmises à la fonction sous forme de commandes et de paramètres sont décrites dans les tableaux de cette section.

Les tableaux contiennent les trois colonnes suivantes :

- La colonne Commande affiche le paramètre de chaîne littéral qui identifie la commande.
- La colonne Paramètres décrit les types de valeurs à transmettre pour les paramètres supplémentaires, le cas échéant.
- La colonne Valeur renvoyée décrit les valeurs renvoyées attendues.

Exemple

Des exemples sont fournis avec les commandes spécifiques exécutées à l'aide de la fonction `fscommand2()` ; ils sont décrits dans la suite de cette section.

Voir aussi

`fscommand()`

Escape

Disponibilité

Flash Lite 1.1.

Description

Code une chaîne arbitraire en un format garantissant un transfert réseau sécurisé. Remplace tous les caractères qui ne sont pas de type alphanumérique par une séquence d'échappement hexadécimale (`%xx` ou `%xx%xx` dans le cas de caractères multioctets).

Commande	Paramètres	Valeur renvoyée
`"Escape"`	*`original`* Chaîne à encoder dans un format prévu pour les URL. *`encoded`* Chaîne de code obtenue. Ces paramètres correspondent soit à des noms de variables, soit à des valeurs de chaîne constantes (par exemple, `"Encoded_String"`).	0 : Echec. 1 : Succès.

Exemple

L'exemple suivant illustre la conversion d'une chaîne dans sa forme codée :

```
original_string = "Hello, how are you?";
status = fscommand2("escape", original_string, "encoded_string");
trace (encoded_string);  // sortie : Hello%2C%20how%20are%20you%3F
```

Voir aussi

`Unescape`

FullScreen

Disponibilité

Flash Lite 1.1.

Description

Définit la taille de la zone d'affichage à utiliser pour le rendu. Cette taille peut être de type plein écran ou de taille inférieure.

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

Commande	Paramètres	Valeur renvoyée
`"FullScreen"`	*`size`* Une variable ou une valeur de chaîne constante définie dont la valeur serait l'une des suivantes : `true` (plein écran) ou `false` (taille inférieure au plein écran). Toute autre valeur est traitée comme la valeur `false`.	-1 : Non prise en charge. 0 : Prise en charge.

Exemple

L'exemple suivant tente de définir la zone d'affichage sur plein écran. Si la valeur renvoyée est différente de 0, elle positionne la tête de lecture sur l'image intitulée smallScreenMode.

```
status = fscommand2("FullScreen", true);
if(status != 0) {
  gotoAndPlay("smallScreenMode");
}
```

GetBatteryLevel

Disponibilité

Flash Lite 1.1.

Description

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

Commande	Paramètres	Valeur renvoyée
`"GetBatteryLevel"`	Aucun.	-1 : Non prise en charge. Autres valeurs numériques : Le niveau de batterie actuel.

Exemple

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

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

Voir aussi

`GetMaxBatteryLevel`

GetDateDay

Disponibilité

Flash Lite 1.1.

Description

Renvoie le jour de la date courante. Il s'agit d'une valeur numérique (sans zéro de tête). Les jours valides sont compris entre 1 et 31.

Commande	Paramètres	Valeur renvoyée
`"GetDateDay"`	Aucun.	-1 : Non prise en charge. 1 à 31 : Le jour du mois.

Exemple

L'exemple suivant collecte les informations relatives à la date et crée une chaîne de date complète :

```
today = fscommand2("GetDateDay");
weekday = fscommand2("GetDateWeekday");
thisMonth = fscommand2("GetDateMonth");
thisYear = fscommand2("GetDateYear");
when = weekday add ", " add ThisMonth add " " add today add ", " add
  thisYear;
```

Voir aussi

`GetDateMonth`, `GetDateWeekday`, `GetDateYear`

GetDateMonth

Disponibilité

Flash Lite 1.1.

Description

Renvoie le mois de la date courante sous forme de valeur numérique (sans zéro de tête).

Commande	Paramètres	Valeur renvoyée
`"GetDateMonth"`	Aucun.	-1 : Non prise en charge. 1 à 12 : Le nombre du mois en cours.

Exemple

L'exemple suivant collecte les informations relatives à la date et crée une chaîne de date complète :

```
today = fscommand2("GetDateDay");
weekday = fscommand2("GetDateWeekday");
thisMonth = fscommand2("GetDateMonth");
thisYear = fscommand2("GetDateYear");
when = weekday add ", " add thisMonth add " " add today add ", " add
  thisYear;
```

Voir aussi

GetDateDay, GetDateWeekday, GetDateYear

GetDateWeekday

Disponibilité

Flash Lite 1.1.

Description

Renvoie une valeur numérique représentant le nom du jour de la date courante.

Commande	Paramètres	Valeur renvoyée
"GetDateWeekday"	Aucun.	-1 : Non prise en charge. 0 : Dimanche. 1 : Lundi. 2 : Mardi. 3 : Mercredi. 4 : Jeudi. 5 : Vendredi. 6 : Samedi.

Exemple

L'exemple suivant collecte les informations relatives à la date et crée une chaîne de date complète :

```
today = fscommand2("GetDateDay");
weekday = fscommand2("GetDateWeekday");
thisMonth = fscommand2("GetDateMonth");
thisYear = fscommand2("GetDateYear");
when = weekday add ", " add thisMonth add " " add today add ", " add
  thisYear;
```

Voir aussi

GetDateDay, GetDateMonth, GetDateYear

GetDateYear

Renvoie une valeur numérique à 4 chiffres représentant l'année de la date courante.

Commande	Paramètres	Valeur renvoyée
"GetDateYear"	Aucun.	-1 : Non prise en charge. 0 à 9999 : L'année en cours.

Disponibilité

Flash Lite 1.1.

Exemple

L'exemple suivant collecte les informations relatives à la date et crée une chaîne de date complète :

```
today = fscommand2("GetDateDay");
weekday = fscommand2("GetDateWeekday");
thisMonth = fscommand2("GetDateMonth");
thisYear = fscommand2("GetDateYear");
when = weekday add ", " add thisMonth add " " add today add ", " add
  thisYear;
```

Voir aussi

GetDateDay, GetDateMonth, GetDateWeekday

GetDevice

Définit un paramètre qui identifie le périphérique sur lequel s'exécute Flash Lite. Cet identificateur correspond généralement au nom du modèle.

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

Disponibilité

Flash Lite 1.1.

Exemple

L'exemple suivant affecte l'identificateur de périphérique à la variable `statusdevice`, puis met à jour un champ de texte avec le nom du périphérique générique.

Voici quelques exemples de résultats et les périphériques auxquels ils se rapportent :

`D506i` Téléphone Mitsubishi 506i.

`DFOMA1` Téléphone Mitsubishi FOMA1.

`F506i` Téléphone Fujitsu 506i.

`FFOMA1` Téléphone Fujitsu FOMA1.

`N506i` Téléphone NEC 506i.

`NFOMA1` Téléphone NEC FOMA1.

`Nokia3650` Téléphone Nokia 3650.

`p506i` Téléphone Panasonic 506i.

`PFOMA1` Téléphone Panasonic FOMA1.

`SH506i` Téléphone Sharp 506i.

`SHFOMA1` Téléphone Sharp FOMA1.

`SO506i` Téléphone Sony 506i.

```
statusdevice = fscommand2("GetDevice", "devicename");
switch(devicename) {
  case "D506i":
    /:myText += "device: Mitsubishi 506i" add newline;
    break;
  case "DFOMA1":
    /:myText += "device: Mitsubishi FOMA1" add newline;
    break;
  case "F506i":
    /:myText += "device: Fujitsu 506i" add newline;
    break;
  case "FFOMA1":
    /:myText += "device: Fujitsu FOMA1" add newline;
    break;
  case "N506i":
    /:myText += "device: NEC 506i" add newline;
    break;
  case "NFOMA1":
    /:myText += "device: NEC FOMA1" add newline;
    break;
  case "Nokia 3650":
    /:myText += "device: Nokia 3650" add newline;
    break;
  case "P506i":
    /:myText += "device: Panasonic 506i" add newline;
```

```
    break;
  case "PFOMA1":
    /:myText += "device: Panasonic FOMA1" add newline;
    break;
  case "SH506i":
    /:myText += "device: Sharp 506i" add newline;
    break;
  case "SHFOMA1":
    /:myText += "device: Sharp FOMA1" add newline;
    break;
  case "SO506i":
    /:myText += "device: Sony 506i" add newline;
    break;
}
```

GetDeviceID

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

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

Disponibilité

Flash Lite 1.1.

Exemple

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

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

GetFreePlayerMemory

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

Commande	Paramètres	Valeur renvoyée
`"GetFreePlayerMemory"`	Aucun.	-1 : Non prise en charge. 0 ou valeur positive : kilooctets de mémoire de tas disponibles.

Disponibilité

Flash Lite 1.1.

Exemple

L'exemple suivant définit l'état comme équivalent à la quantité de mémoire disponible :

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

Voir aussi

`GetTotalPlayerMemory`

GetLanguage

Disponibilité

Flash Lite 1.1.

Définit un paramètre qui identifie la langue actuellement utilisée par le périphérique. La langue est renvoyée sous forme de chaîne dans une variable transmise en fonction du nom.

Commande	Paramètres	Valeur renvoyée
"GetLanguage"	*language* Chaîne devant recevoir le code de langue. Il peut s'agir du nom d'une variable ou d'une valeur de chaîne contenant le nom d'une variable. La valeur renvoyée est l'une des suivantes : cs : Tchèque. da : Danois. de : Allemand. en-UK : Anglais international. en-US : Anglais américain. es : Espagnol. fi : Finnois. fr : Français. hu : Hongrois. it : Italien. jp : Japonais. ko : Coréen. nl : Néerlandais. no : Norvégien. pl : Polonais. pt : Portugais. ru : Russe. sv : Suédois. tr : Turc. xu : Langue non déterminée. zh-CN : Chinois simplifié. zh-TW : Chinois traditionnel.	-1 : Non prise en charge. 0 : Prise en charge.

REMARQUE

Lorsque les téléphones japonais sont configurés pour afficher l'anglais, la valeur en_US est renvoyée pour *language*.

Exemple

L'exemple suivant affecte le code de langue à la variable `language`, puis met à jour un champ de texte avec la langue reconnue par le lecteur Flash Lite :

```
statuslanguage = fscommand2("GetLanguage", "language");
switch(language) {
  case "cs":
    /:myText += "language is Czech" add newline;
    break;
  case "da":
    /:myText += "language is Danish" add newline;
    break;
  case "de":
    /:myText += "language is German" add newline;
    break;
  case "en-UK":
    /:myText += "language is UK" add newline;
    break;
  case "en-US":
    /:myText += "language is US" add newline;
    break;
  case "es":
    /:myText += "language is Spanish" add newline;
    break;
  case "fi":
    /:myText += "language is Finnish" add newline;
    break;
  case "fr":
    /:myText += "language is French" add newline;
    break;
  case "hu":
    /:myText += "language is Hungarian" add newline;
    break;
  case "it":
    /:myText += "language is Italian" add newline;
    break;
  case "jp":
    /:myText += "language is Japanese" add newline;
    break;
  case "ko":
    /:myText += "language is Korean" add newline;
    break;
  case "nl":
    /:myText += "language is Dutch" add newline;
    break;
  case "no":
    /:myText += "language is Norwegian" add newline;
    break;
  case "pl":
    /:myText += "language is Polish" add newline;
```

```
    break;
  case "pt":
    /:myText += "language is Portuguese" add newline;
    break;
  case "ru":
    /:myText += "language is Russian" add newline;
    break;
  case "sv":
    /:myText += "language is Swedish" add newline;
    break;
  case "tr":
    /:myText += "language is Turkish" add newline;
    break;
  case "xu":
    /:myText += "language is indeterminable" add newline;
    break;
  case "zh-CN":
    /:myText += "language is simplified Chinese" add newline;
    break;
  case "zh-TW":
    /:myText += "language is traditional Chinese" add newline;
    break;
}
```

GetLocaleLongDate

Disponibilité

Flash Lite 1.1.

Description

Définit un paramètre en chaîne représentant la date courante, au format long, formatée en fonction des paramètres régionaux spécifiés.

Commande	Paramètres	Valeur renvoyée
`"GetLocaleLongDate"`	*`longdate`* Variable de chaîne devant recevoir le format long de la valeur de la date courante, comme par exemple « Octobre 2004, 16 » ou « 16 octobre 2004 ». Il peut s'agir du nom d'une variable ou d'une valeur de chaîne contenant le nom d'une variable. La valeur renvoyée dans *`longdate`* est une chaîne à caractères multiples de longueur variable. La mise en forme réelle dépend du périphérique et des paramètres régionaux.	-1 : Non prise en charge. 0 : Prise en charge.

Exemple

L'exemple suivant tente de renvoyer le format long de la date courante dans la variable `longDate`. Il détermine également la valeur de `status` afin de vérifier s'il peut le faire.

```
status = fscommand2("GetLocaleLongDate", "longdate");
trace (longdate);      // sortie : Mardi 14 juin 2005
```

Voir aussi

`GetLocaleShortDate`, `GetLocaleTime`

GetLocaleShortDate

Disponibilité

Flash Lite 1.1.

Description

Définit un paramètre en chaîne représentant la date courante, au format abrégé, formatée en fonction des paramètres régionaux spécifiés.

Commande	Paramètres	Valeur renvoyée
`"GetLocaleShortDate"`	*`shortdate`* Variable de chaîne devant recevoir le format abrégé de la valeur de la date courante, comme par exemple « `16/10/2004` » ou « `16-10-2004` ». Il peut s'agir du nom d'une variable ou d'une valeur de chaîne contenant le nom d'une variable. La valeur renvoyée dans *`shortdate`* est une chaîne à caractères multiples de longueur variable. La mise en forme réelle dépend du périphérique et des paramètres régionaux.	-1 : Non prise en charge. 0 : Prise en charge.

Exemple

L'exemple suivant tente de récupérer le format abrégé de la date courante dans la variable `shortDate`. Il détermine également la valeur de `status` afin de vérifier s'il peut le faire.

```
status = fscommand2("GetLocaleShortDate", "shortdate");
trace (shortdate);     // sortie : 06/14/05
```

Voir aussi

`GetLocaleLongDate`, `GetLocaleTime`

GetLocaleTime

Disponibilité

Flash Lite 1.1.

Description

Définit un paramètre en chaîne représentant l'heure courante, formatée en fonction des paramètres régionaux spécifiés.

Commande	Paramètres	Valeur renvoyée
`"GetLocaleTime"`	*time* Variable de chaîne devant recevoir la valeur de l'heure actuelle, comme par exemple `"6:10:44 PM"` ou `"18:10:44"`. Il peut s'agir du nom d'une variable ou d'une valeur de chaîne contenant le nom d'une variable. La valeur renvoyée dans *time* est une chaîne à caractères multiples de longueur variable. La mise en forme réelle dépend du périphérique et des paramètres régionaux.	-1 : Non prise en charge. 0 : Prise en charge.

Exemple

L'exemple suivant tente de récupérer l'heure locale courante dans la variable `time`. Il détermine également la valeur de `status` afin de vérifier s'il peut le faire.

```
status = fscommand2("GetLocaleTime", "time");
trace(time);                // sortie : 14:30:21
```

Voir aussi

`GetLocaleLongDate`, `GetLocaleShortDate`

GetMaxBatteryLevel

Disponibilité

Flash Lite 1.1.

Description

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

Commande	Paramètres	Valeur renvoyée
`"GetMaxBatteryLevel"`	Aucun.	-1 : Non prise en charge. Autres valeurs : le niveau de batterie maximum.

Exemple

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

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

GetMaxSignalLevel

Disponibilité

Flash Lite 1.1.

Description

Renvoie le niveau d'intensité maximal du signal. Il s'agit d'une valeur numérique supérieure à 0.

Commande	Paramètres	Valeur renvoyée
`"GetMaxSignalLevel"`	Aucun.	-1 : Non prise en charge. Autres valeurs numériques : le niveau maximal du signal.

Exemple

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

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

GetMaxVolumeLevel

Disponibilité

Flash Lite 1.1.

Description

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

Commande	Paramètres	Valeur renvoyée
`"GetMaxVolumeLevel"`	Aucun.	-1 : Non prise en charge. Autres valeurs : le niveau de volume maximal.

Exemple

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

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

Voir aussi

`GetVolumeLevel`

GetNetworkConnectStatus

Disponibilité

Flash Lite 1.1.

Description

Renvoie une valeur indiquant l'état actuel de la connexion réseau.

Commande	Paramètres	Valeur renvoyée
`"GetNetworkConnectStatus"`	Aucun.	-1 : Non prise en charge. 0 : Une connexion réseau est actuellement active. 1 : Le périphérique tente de se connecter au réseau. 2 : Aucune connexion réseau n'est actuellement active. 3 : La connexion réseau est suspendue. 4 : La connexion réseau ne peut pas être déterminée.

Exemple

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

```
connectstatus = fscommand2("GetNetworkConnectStatus");
switch (connectstatus) {
  case -1 :
    /:myText += "connectstatus not supported" add newline;
    break;
  case 0 :
    /:myText += "connectstatus shows active connection" add newline;
    break;
  case 1 :
    /:myText += "connectstatus shows attempting connection" add newline;
    break;
  case 2 :
    /:myText += "connectstatus shows no connection" add newline;
    break;
  case 3 :
    /:myText += "connectstatus shows suspended connection" add newline;
    break;
  case 4 :
    /:myText += "connectstatus shows indeterminable state" add newline;
    break;
}
```

GetNetworkName

Disponibilité

Flash Lite 1.1.

Description

Définit un paramètre sur le nom du réseau actif.

Commande	Paramètres	Valeur renvoyée
`"GetNetworkName"`	*networkName* Chaîne représentant le nom du réseau. Il peut s'agir du nom d'une variable ou d'une valeur de chaîne contenant le nom d'une variable. Si le réseau est enregistré et que son nom peut être déterminé, *networkname* est définie sur le nom du réseau ; sinon, elle est définie sur la chaîne vide.	-1 : Non prise en charge. 0 : Aucun réseau n'est enregistré. 1 : Un réseau est enregistré, mais le nom du réseau est inconnu. 2 : Un réseau est enregistré et le nom du réseau est connu.

Exemple

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

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

GetNetworkRequestStatus

Disponibilité

Flash Lite 1.1.

Description

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

Commande	Paramètres	Valeur renvoyée
`"GetNetworkRequestStatus"`	Aucun.	-1 : La commande n'est pas prise en charge. 0 : Une requête est en attente de traitement, une connexion réseau a été établie, le nom d'hôte du serveur a été résolu et une connexion au serveur a été établie. 1 : Une requête est en attente de traitement et une connexion réseau est en cours d'établissement. 2 : Une requête est en attente de traitement, mais la connexion réseau n'a pas encore été établie. 3 : Une requête est en attente de traitement, une connexion réseau a été établie et le nom d'hôte du serveur est déterminé. 4 : La requête a échoué en raison d'une erreur de réseau. 5 : La requête a échoué en raison d'un échec de connexion au serveur. 6 : Le serveur a renvoyé une erreur HTTP (par exemple, 404). 7 : La requête a échoué en raison d'un échec lors de la tentative d'accès au serveur DNS ou de la tentative de détermination du nom de serveur. 8 : La requête a été traitée avec succès. 9 : La requête a échoué car la session a expiré. 10 : La requête n'a pas encore été effectuée.

Exemple

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

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

GetNetworkStatus

Disponibilité

Flash Lite 1.1.

Description

Renvoie une valeur indiquant l'état du réseau du téléphone (autrement dit, si un réseau est enregistré et si le téléphone est doté de la fonctionnalité d'itinérance).

Commande	Paramètres	Valeur renvoyée
`"GetNetworkStatus"`	Aucun.	-1 : La commande n'est pas prise en charge. 0 : Aucun réseau n'est enregistré. 1 : Sur un réseau domestique. 2 : Sur un réseau domestique étendu. 3 : Itinérance (loin du réseau domestique).

Exemple

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

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

GetPlatform

Disponibilité

Flash Lite 1.1.

Description

Définit un paramètre qui identifie la plate-forme actuelle et donne ainsi une brève description de la classe du périphérique. Pour les périphériques équipés de systèmes d'exploitation ouverts, cet identificateur correspond généralement au nom et à la version du système d'exploitation.

Commande	Paramètres	Valeur renvoyée
`"GetPlatform"`	*`platform`* Chaîne devant recevoir l'identificateur de la plate-forme. Il peut s'agir du nom d'une variable ou d'une valeur de chaîne contenant le nom d'une variable.	-1 : Non prise en charge. 0 : Prise en charge.

Exemple

L'exemple de code suivant affecte l'identificateur de la plate-forme à la variable `statusplatform`, puis met à jour un champ de texte avec le nom générique de la plate-forme.

Les exemples suivants illustrent quelques résultats pour `myPlatform` et les classes de périphérique auxquels ils se rapportent :

`506i` Téléphone 506i.

`FOMA1` Téléphone FOMA1.

`Symbian6.1_s60.1` Téléphone Symbian 6.1, Series 60 version 1.

`Symbian7.0` Téléphone Symbian 7.0

```
statusplatform = fscommand2("GetPlatform", "platform");
switch(platform){
  case "506i":
    /:myText += "platform: 506i" add newline;
    break;
  case "FOMA1":
    /:myText += "platform: FOMA1" add newline;
    break;
  case "Symbian6.1-Series60v1":
    /:myText += "platform: Symbian6.1, Series 60 version 1 phone" add
  newline;
    break;
  case "Symbian7.0":
    /:myText += "platform: Symbian 7.0" add newline;
    break;
}
```

GetPowerSource

Disponibilité

Flash Lite 1.1.

Description

Renvoie une valeur qui indique si l'alimentation provient d'une batterie ou d'une source d'alimentation externe.

Commande	Paramètres	Valeur renvoyée
`"GetPowerSource"`	Aucun.	-1 : Non prise en charge. 0 : Le périphérique fonctionne sur batterie. 1 : Le périphérique fonctionne sur une source d'alimentation externe.

Exemple

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

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

GetSignalLevel

Disponibilité

Flash Lite 1.1.

Description

Renvoie l'intensité courante du signal sous forme de valeur numérique.

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

Exemple

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

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

GetTimeHours

Disponibilité

Flash Lite 1.1.

Description

Renvoie l'heure courante du jour, en fonction d'une journée de 24 heures. Il s'agit d'une valeur numérique (sans zéro de tête).

Commande	Paramètres	Valeur renvoyée
`"GetTimeHours"`	Aucun.	-1 : Non prise en charge. 0 à 23 : L'heure actuelle.

Exemple

L'exemple suivant définit la variable `hour` sur la tranche horaire de l'heure actuelle, ou sur -1 :

```
hour = fscommand2("GetTimeHours");
trace (hour);            // sortie : 14
```

Voir aussi

`GetTimeMinutes`, `GetTimeSeconds`, `GetTimeZoneOffset`

GetTimeMinutes

Disponibilité

Flash Lite 1.1.

Description

Renvoie les minutes de l'heure actuelle. Il s'agit d'une valeur numérique (sans zéro de tête).

Commande	Paramètres	Valeur renvoyée
`"GetTimeMinutes"`	Aucun.	-1 : Non prise en charge. 0 à 59 : Les minutes actuelles.

Exemple

L'exemple suivant définit la variable `minutes` sur les minutes de l'heure actuelle, ou sur -1 :

```
minutes = fscommand2("GetTimeMinutes");
trace (minutes);         // sortie : 38
```

Voir aussi

GetTimeHours, GetTimeSeconds, GetTimeZoneOffset

GetTimeSeconds

Disponibilité

Flash Lite 1.1.

Description

Renvoie les secondes de l'heure actuelle. Il s'agit d'une valeur numérique (sans zéro de tête).

Commande	Paramètres	Valeur renvoyée
`"GetTimeSeconds"`	Aucun.	-1 : Non prise en charge. 0 à 59 : Les secondes actuelles.

Exemple

L'exemple suivant définit la variable `secondes` sur les secondes de l'heure actuelle, ou sur -1 :

```
seconds = fscommand2("GetTimeSeconds");
trace (seconds);         // sortie : 41
```

Voir aussi

GetTimeHours, GetTimeMinutes, GetTimeZoneOffset

GetTimeZoneOffset

Disponibilité

Flash Lite 1.1.

Description

Définit un paramètre en nombre de minutes entre le fuseau horaire local et l'heure universelle (UTC).

Commande	Paramètres	Valeur renvoyée
`"GetTimeZoneOffset"`	*`timezoneOffset`* Nombre de minutes entre le fuseau horaire local et l'heure universelle (UTC). Il peut s'agir du nom d'une variable ou d'une valeur de chaîne contenant le nom d'une variable. Une valeur numérique positive ou négative est renvoyée, telle que : 540 : heure normale du Japon -420 : heure d'été du Pacifique	-1 : Non prise en charge. 0 : Prise en charge.

Exemple

L'exemple suivant affecte les minutes de décalage de l'heure universelle (UTC) à la variable `timezoneoffset` et définit `status` sur 0 ou bien définit `status` sur -1 :

```
status = fscommand2("GetTimeZoneOffset", "timezoneoffset");
trace (timezoneoffset);// sortie : 300
```

Voir aussi

`GetTimeHours`, `GetTimeMinutes`, `GetTimeSeconds`

GetTotalPlayerMemory

Disponibilité

Flash Lite 1.1.

Description

Renvoie la quantité totale de mémoire de tas, en kilooctets, allouée à Flash Lite.

Commande	Paramètres	Valeur renvoyée
`"GetTotalPlayerMemory"`	Aucun.	-1 : Non prise en charge. 0 ou valeur positive : Nombre de kilooctets total de mémoire de tas.

Exemple

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

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

Voir aussi

`GetFreePlayerMemory`

GetVolumeLevel

Disponibilité

Flash Lite 1.1.

Description

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

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

Exemple

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

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

Voir aussi

GetVolumeLevel

Quit

Disponibilité

Flash Lite 1.1.

Description

Oblige le lecteur Flash Lite à arrêter la tête de lecture et à quitter l'application.

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

Commande	Paramètres	Valeur renvoyée
`"Quit"`	Aucun.	-1 : Non prise en charge.

Exemple

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

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

ResetSoftKeys

Disponibilité

Flash Lite 1.1.

Description

Réinitialise les touches programmables sur leurs paramètres d'origine.

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

Commande	Paramètres	Valeur renvoyée
`"ResetSoftKeys"`	Aucun.	-1 : Non prise en charge. 0 : Prise en charge.

Exemple

L'instruction suivante réinitialise les touches programmables sur leurs paramètres d'origine.

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

Voir aussi

SetSoftKeys

SetInputTextType

Disponibilité

Flash Lite 1.1.

Description

Spécifie le mode dans lequel ouvrir le champ de texte de saisie :

Commande	Paramètres	Valeur renvoyée
"SetInputTextType"	*variableName* Nom du champ de texte de saisie. Il peut s'agir du nom d'une variable ou d'une valeur de chaîne contenant le nom d'une variable. *type* Une des valeurs Numeric, Alpha, Alphanumeric, Latin, NonLatin ou NoRestriction.	0 : Echec. 1 : Succès.

Flash Lite prend en charge la fonction de texte de saisie et demande pour cela à l'application hôte de lancer l'interface générique de saisie de texte spécifique au périphérique, souvent appelée *processeur frontal* (FEP). Si la commande SetInputTextType n'est pas utilisée, le mode par défaut du FEP est ouvert.

Le tableau suivant montre les effets de chaque mode ainsi que les modes de remplacement :

Mode spécifié	Définit le FEP sur un de ces modes qui s'excluent mutuellement	S'il n'est pas pris en charge sur le périphérique actif, ouvre le FEP dans ce mode
Numeric	Nombres uniquement (0 à 9)	Alphanumeric
Alpha	Caractères alphabétiques uniquement (A à Z, a à z)	Alphanumeric
Alphanumeric	Caractères alphanumériques uniquement (0 à 9, A à Z, a à z)	Latin
Latin	Caractères latins uniquement (alphanumériques et ponctuation)	NoRestriction
NonLatin	Caractères non latins uniquement (par exemple, Kanji et Kana)	NoRestriction
NoRestriction	Mode par défaut (ne définit aucune restriction sur le FEP)	

REMARQUE

Certains téléphones portables ne prennent pas en charge ces types de champ de texte de saisie. Pour cette raison, vous devez valider les données des champs de texte de saisie.

Exemple

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

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

SetQuality

Disponibilité

Flash Lite 1.1.

Description

Définit la qualité de rendu de l'animation.

Commande	Paramètres	Valeur renvoyée
`"SetQuality"`	*`quality`* La qualité de rendu ; doit être `"high"`, `"medium"` ou `"low"`.	-1 : Non prise en charge. 0 : Prise en charge.

Exemple

L'exemple suivant définit la qualité de rendu sur low :

```
status = fscommand2("SetQuality", "low");
```

SetSoftKeys

Disponibilité

Flash Lite 1.1.

Description

Remappe les touches programmables Gauche et Droite du périphérique, sous réserve qu'elles soient accessibles et remappables.

Après avoir exécuté cette commande, appuyer sur la touche gauche génère un événement de touche `PageUp` alors qu'appuyer sur la touche droite génère un événement de touche `PageDown`. Le code ActionScript associé aux événements de touche `PageUp` et `PageDown` est exécuté si l'utilisateur appuie sur la touche concernée.

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

Commande	Paramètres	Valeur renvoyée
`"SetSoftKeys"`	*`left`* Le texte à afficher pour la touche programmable Gauche. *`right`* Le texte à afficher pour la touche programmable Droite. Ces paramètres correspondent soit à des noms de variables, soit à des valeurs de chaîne constantes (par exemple, `"Previous"`).	-1 : Non prise en charge. 0 : Prise en charge.

Exemple

L'exemple suivant indique que la touche programmable Gauche doit être étiquetée Previous et la touche programmable Droite Next.

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

Voir aussi

`ResetSoftKeys`

StartVibrate

Disponibilité

Flash Lite 1.1.

Description

Lance la fonction vibration du téléphone. Si une vibration est déjà en cours, Flash Lite l'arrête avant de lancer la nouvelle. Les vibrations sont également arrêtées en cas d'arrêt ou de pause de la tête de lecture de l'application Flash, ou lorsque vous quittez le lecteur Flash Lite.

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

Exemple

L'exemple suivant tente de lancer une séquence de vibrations, avec activation de 2,5 secondes, désactivation de 1 seconde, le tout répété deux fois. Il affecte une valeur à la variable `status` indiquant le succès ou l'échec de l'opération.

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

Voir aussi

`StopVibrate`

StopVibrate

Disponibilité

Flash Lite 1.1.

Description

Arrête la vibration en cours, le cas échéant.

Commande	Paramètres	Valeur renvoyée
`"StopVibrate"`	Aucun.	-1 : Non prise en charge. 0 : La vibration est arrêtée.

Exemple

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

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

Voir aussi

`StartVibrate`

Unescape

Disponibilité

Flash Lite 1.1.

Description

Décode une chaîne quelconque qui a été codée pour garantir un transfert sécurisé sur le réseau pour revenir à son format normal non codé. Tous les caractères au format hexadécimal, à savoir, un caractère pourcentage (%) suivi de deux chiffres hexadécimaux, sont convertis en forme décodée.

Commande	Paramètres	Valeur renvoyée
`"Unescape"`	*`original`* Chaîne à décoder d'un format prévu pour les URL en une forme normale. *`decoded`* Chaîne décodée obtenue. (Ce paramètre peut être le nom d'une variable ou une valeur de chaîne contenant le nom d'une variable.)	0 : Echec. 1 : Succès.

Exemple

L'exemple suivant illustre le décodage d'une chaîne codée :

```
encoded_string = "Hello%2C%20how%20are%20you%3F";
status = fscommand2("unescape", encoded_string, "normal_string");
trace (normal_string);   // sortie : Hello, how are you?
```

Voir aussi

Escape

Index

Symboles

! (NOT logique) 96
++ (incrément), opérateur 94
+= (affectation d'addition), opérateur 85
, opérateur (virgule) 88
-= (affectation de soustraction), opérateur 113
. opérateur (point) 93
< opérateur (numérique inférieur à) 104
< opérateur (numérique inférieur ou égal à) 105
<> opérateur (inégalité numérique) 103
> opérateur (supérieur à) 102
> opérateur (supérieur ou égal à) 103
? (conditionnel) 90
_currentframe, propriété 54
_focusrect, propriété 54
_framesloaded, propriété 55
_height, propriété 55
_highquality, propriété 56
_level, propriété 57
_name, propriété 58
_rotation, propriété 59
_target, propriété 60
_visible, propriété 61
_width, propriété 62
_x, propriété 62
_xscale, propriété 63
_y, propriété 64
_yscale, propriété 65

A

addition numérique 101
affectation modulo 98
and 86

B

Boucle for 75
Boucle while 72

C

call 13
variable _cap4WayKeyAS 125
variable _capCompoundSound 118
variable _capEmail 119
variable _capLoadData 120
variable _capMFi 121
variable _capMMS 121
variable _capSMAF 123
variable _capSMS 123
variable _capStreamSound 124
chaîne inférieure ou égale à 111
chaîne supérieure ou égale à 109
commande fscommand() 126
commentaires
    block 87
    sur une ligne 89
concaténation 84
conditions 76
_currentframe, propriété 54

D

division 92

F

_focusrect, propriété 54
Fonction chr() 14
Fonction duplicateMovieClip() 15
Fonction eval() 16

Fonction getProperty() 17
Fonction getTimer() 18
Fonction getURL() 18
Fonction gotoAndPlay() 21
Fonction gotoAndStop() 22
Fonction ifFrameLoaded() 23
Fonction int() 24
Fonction length() 25
Fonction loadMovieNum() 27
Fonction loadVariablesNum() 30
Fonction mbchr() 31
Fonction mbsubstring() 33
Fonction nextFrame() 34
Fonction nextScene() 34
Fonction Number() 35
Fonction on() 36
Fonction ord() 37
Fonction play() 37
Fonction prevFrame() 38
Fonction prevScene() 39
Fonction random() 39
Fonction removeMovieClip() 40
Fonction set() 41
Fonction setProperty() 42
Fonction stop() 42
Fonction stopAllSounds() 43
Fonction String() 44
Fonction substring() 45
Fonction tellTarget() 45
Fonction toggleHighQuality() 46
Fonction trace() 47
Fonction unloadMovie() 48
Fonction unloadMovieNum() 49
fonctions
  chr() 14
  duplicateMovieClip() 15
  eval () 16
  fscommand() 126
  getProperty() 17
  getTimer() 18
  getURL() 18
  gotoAndPlay() 21
  gotoAndStop() 22
  ifFrameLoaded() 23
  int() 24
  length() 25
  loadMovie() 25
  loadMovieNum() 27
  loadVariables() 28
  loadVariablesNum() 30
  mbchr() 31
  mbsubstring() 33
  nextFrame() 34
  nextScene() 34
  Number() 35
  on() 36
  ord() 37
  play() 37
  prevFrame() 38
  prevScene() 39
  random() 39
  removeMovieClip() 40
  set() 41
  setProperty() 42
  stop() 42
  stopAllSounds() 43
  String() 44
  substring() 45
  tellTarget() 45
  toggleHighQuality() 46
  trace() 47
  unloadMovie() 48
  unloadMovieNum() 49
_framesloaded, propriété 55

H

_height, propriété 55
_highquality, propriété 56

I

identificateur du scénario racine 52
Instruction break 68
Instruction case 69
Instruction continue 70
Instruction do..while 72
Instruction else 73
Instruction else if 74
instruction for 75
Instruction if 76
Instruction while 78

L

_level, propriété 57
loadMovie(), fonction 25
loadVariables(), fonction 28

M

maxscroll, propriété 58
Messagerie MMS 121
multiplication 100

N

_name, propriété 58

O

opérateur " " (séparateur de chaîne) 106
opérateur % (modulo) 98
opérateur %= (affectation modulo) 98
|| (OR logique), opérateur 97
opérateur && (AND logique) 95
opérateur * (produit) 100
opérateur *= (affectation de multiplication) 99
opérateur + (addition numérique) 101
opérateur , (virgule) 88
opérateur / (division) 92
opérateur /* (bloc de commentaires) 87
opérateur // (commentaire) 89
opérateur /= (division) 92
opérateur = (affectation) 87
opérateur == (égalité numérique) 101
opérateur – (soustraction) 112
opérateur — (décrément) 91
Opérateur add (concaténation de chaîne) 84
Opérateur AND 86, 95
opérateur AND logique 95
opérateur conditionnel 90
Opérateur d'affectation 87
opérateur d'affectation de division 92
Opérateur d'affectation de l'addition 85
opérateur d'affectation de soustraction 113
opérateur d'égalité de chaîne 107
opérateur d'incrémentation 94
opérateur d'inégalité 103
opérateur de bloc de commentaires 87
opérateur de chaîne supérieur à 108
opérateur eq (égalité de chaîne) 107
opérateur ge (chaîne supérieure ou égale à) 109
opérateur gt (chaîne supérieur à) 108
opérateur inférieur à 104
opérateur inférieur ou égal à 105
opérateur le (chaîne inférieure ou égale à) 111
opérateur lt (chaîne inférieure à) 110
opérateur modulo 98
opérateur ne (inégalité de chaîne) 110
Opérateur NOT 96
opérateur NOT logique 96
Opérateur OR 97
opérateur OR logique 97
opérateur point 93
opérateur séparateur de chaîne 106
opérateur supérieur à 102
opérateur supérieur ou égal à 103
opérateur virgule 88
opérateurs
  addition numérique 101
  affectation 87
  affectation d'addition 85
  affectation de division 92
  affectation de soustraction 113
  affectation modulo 98
  AND logique 95
  bloc de commentaires 87
  chaîne inférieure à 110
  chaîne inférieure ou égale à 111
  chaîne supérieur à 108
  chaîne supérieure ou égale à 109
  comment 89
  concaténation de chaîne 84
  conditionnelles 90
  division 92
  égalité de chaîne 107
  égalité numérique 101
  incrémentation 94
  inégalité de chaîne 110
  inégalité numérique 103
  modulo 98
  NOT logique 96
  numérique inférieur à 104
  numérique inférieur ou égal à 105
  OR logique 97
  point 93
  produit 100
  séparateur de chaîne 106
  supérieur à 102
  supérieur ou égal à 103
  virgule 88

P

Propriété / (barre de fraction - scénario principal) 52
propriété _scroll 60
propriétés
  _alpha 53
  _currentframe 54
  _focusrect 54
  _framesloaded 55
  _height 55
  _highquality 56
  _level 57
  _name 58
  _rotation 59
  _scroll 60
  _target 60
  _visible 61
  _width 62
  _x 62
  _xscale 63
  _y 64
  _yscale 65
  barre de fraction 52
  maxscroll 58
  scroll 60

R

_rotation, propriété 59

S

scroll, propriété 60
son MFI 120
son MIDI 121
statements
  break 68
  case 69
  continue 70
  do..while 72
  else 73
  else if 74
  for 75
  if 76
  NOT logique 96
  switch 77
  while 78
switch, instruction 77

T

_target, propriété 60
_totalframes, propriété 61

V

variable $version 126
variable _alpha 53
variable _cap4WayKeyAS 125
variable _capCompoundSound 118
variable _capEmail 119
variable _capLoadData 120
variable _capMFi 120
variable _capMIDI 121
variable _capMMS 121
variable _capSMAF 123
variable _capSMS 123
variable _capStreamSound 124
variable de fonctionnalité d'envoi de messages électroniques 119
variables
  $version 126
  _alpha 53
  _cap4WayKeyAS 125
  _capCompoundSound 118
  _capEmail 119
  _capLoadData 120
  _capMFi 120
  _capMIDI 121
  _capMMS 121
  _capSMAF 123
  _capSMS 123
  _capStreamSound 124
  fonctionnalité d'envoi de messages électroniques 119
  fonctionnalité de chargement des données 120
  navigation avec les flèches 125
  numéro de version de Flash Lite 126
variables de messagerie 121, 123
variables de son 118, 120, 121, 123, 124
variables, messagerie
  _capMMS 121
  _capSMS 123
variables, son
  _capCompoundSound 118
  _capMFi 120
  _capMIDI 121
  _capSMAF 123
  _capStreamSound 124

_visible, propriété 61

W

propriété _width 62

X

_x, propriété 62
_xscale, propriété 63

Y

_y, propriété 64
_yscale, propriété 65