ADOBE® FLASH® LITE™ 1.x

Guide de référence du langage Adobe® ActionScript®

Adobe

Guide de référence du langage ActionScript™ Flash® Lite™ 1.x d'Adobe®

Updated Information/Additional Third Party Code Information available at http://www.adobe.com/go/thirdparty/.

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

Sommaire

Chapitre 1 : Présentation

Dossier Samples 1
Conventions typographiques 1

Chapitre 2 : Fonctions globales de Flash Lite

call() 3
chr() 4
duplicateMovieClip() 5
eval () 5
getProperty() 6
getTimer() 7
getURL() 7
gotoAndPlay() 10
gotoAndStop() 10
ifFrameLoaded() 11
int() 12
length() 12
loadMovie() 13
loadMovieNum() 14
loadVariables() 15
loadVariablesNum() 16
mbchr() 16
mblength() 17
mbord() 18
mbsubstring() 18
nextFrame() 19
nextScene() 19
Number() 20
on() 21
ord() 21
play() 22
prevFrame() 22
prevScene() 23
random() 24
removeMovieClip() 24
set() 25
setProperty() 26
stop() 26
stopAllSounds() 27
String() 27
substring() 28
tellTarget() 28

toggleHighQuality() 29
trace() 30
unloadMovie() 30
unloadMovieNum() 31

Chapitre 3 : Propriétés de Flash Lite
/ (barre de fraction) 33
_alpha 33
_currentframe 34
_focusrect 34
_framesloaded, ... 35
_height 35
_highquality 36
_level 36
maxscroll 37
_name 38
_rotation 38
scroll 38
_target 39
_totalframes 39
_visible 40
_width 40
_x 41
_xscale 41
_y 42
_yscale 42

Chapitre 4 : Instructions de Flash Lite
break 44
case 45
continue 46
do..while 47
else 48
else if 49
for 50
if 50
switch 51
while 52

Chapitre 5 : Opérateurs de Flash Lite
add (concaténation de chaîne) 56
+= (affectation d'addition) 56
and 57
= (affectation) 58
/* (bloc de commentaires) 58
, (virgule) 59
// (commentaire) 60

?: (conditionnel) 61
–– (décrément) 61
/ (division) 62
/= (affectation de division) 62
. (point) 63
++ (incrément) 64
&& (AND logique) 65
! (NOT logique) 65
|| (OR logique) 66
% (modulo) 67
%= (affectation modulo) 67
*= (affectation de multiplication) 68
* (produit) 69
+ (addition numérique) 69
== (égalité numérique) 70
> (numérique supérieur à) 70
>= (numérique supérieur ou égal à) 71
<> (inégalité numérique) 72
< (numérique inférieur à) 72
<= (numérique inférieur ou égal à) 73
() (parenthèses) 73
" " (séparateur de chaîne) 74
eq (égalité de chaîne) 75
gt (chaîne supérieure à) 75
ge (chaîne supérieure ou égale à) 76
ne (inégalité de chaîne) 77
lt (chaîne inférieure à) 77
le (chaîne inférieure ou égale à) 78
– (soustraction) 79
-= (affectation de soustraction) 80

Chapitre 6 : Eléments de langage spécifiques de Flash Lite

Fonctionnalités 83
_capCompoundSound 83
_capEmail 84
_capLoadData 84
_capMFi 85
_capMIDI 85
_capMMS 86
_capMP3 86
_capSMAF 87
_capSMS 87
_capStreamSound 88
_cap4WayKeyAS 88
$version 89
fscommand() 90

Launch 90
fscommand2() 91
Escape 92
FullScreen 92
GetBatteryLevel 93
GetDateDay 93
GetDateMonth 94
GetDateWeekday 94
GetDateYear 95
GetDevice 96
GetDeviceID 97
GetFreePlayerMemory 98
GetLanguage 98
GetLocaleLongDate 101
GetLocaleShortDate 102
GetLocaleTime 102
GetMaxBatteryLevel 103
GetMaxSignalLevel 103
GetMaxVolumeLevel 104
GetNetworkConnectStatus 104
GetNetworkName 105
GetNetworkRequestStatus 105
GetNetworkStatus 107
GetPlatform 108
GetPowerSource 109
GetSignalLevel 110
GetTimeHours 110
GetTimeMinutes 110
GetTimeSeconds 111
GetTimeZoneOffset 111
GetTotalPlayerMemory 112
GetVolumeLevel 112
Quit 113
ResetSoftKeys 113
SetInputTextType 114
SetQuality 115
SetSoftKeys 115
StartVibrate 116
StopVibrate 117
Unescape 117

Index 119

Chapitre 1 : Présentation

Ce manuel décrit la syntaxe et l'utilisation des éléments du code ActionScript tels que vous les utilisez en vue de développer des applications pour les logiciels Macromedia® Flash® Lite™ 1.0 et Macromedia® Flash® Lite™ 1.1 d'Adobe, dont l'ensemble porte le nom de Flash Lite 1.x. Le code ActionScript de Flash Lite 1.x est basé sur la version du code ActionScript utilisée dans Macromedia® Flash® 4 d'Adobe. Pour utiliser des exemples dans un script, copiez l'exemple de code figurant dans ce manuel et collez-le dans le panneau Script ou dans un fichier de script externe. Ce manuel répertorie tous les éléments du code ActionScript : opérateurs, mots-clés, instructions, commandes, propriétés, fonctions, classes et méthodes.

Dossier Samples

Pour obtenir des exemples de projets Flash Lite complets lors de l'utilisation du code ActionScript, consultez la page qui regroupe des didacticiels et des exemples pour Flash Lite à l'adresse www.adobe.com/go/learn_flt_samples_and_tutorials_fr. Repérez et téléchargez le fichier .zip correspondant à votre version d'ActionScript, puis décompressez-le. Recherchez ensuite le dossier Samples et ouvrez-le pour afficher l'exemple qui vous intéresse.

Conventions typographiques

Ce manuel utilise les conventions typographiques suivantes :

- La *police en italique* indique une valeur à remplacer (par exemple, dans le chemin d'un dossier).
- `La police de code indique le code ActionScript.`
- La *police de code en italique* signale un paramètre ActionScript.
- La **police en gras** désigne une entrée littérale.
- Les guillemets droits (`""`), dans les exemples de code, séparent les chaînes de caractères. Cependant, les programmeurs peuvent également employer des guillemets simples.

Chapitre 2 : Fonctions globales de Flash Lite

Cette section décrit la syntaxe et l'utilisation des fonctions globales ActionScript du logiciel Macromedia Flash Lite 1.1 d'Adobe. Elle comprend les rubriques suivantes :

Function	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 désigné.
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, à une URL donnée.
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 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.
ifFrameLoaded()	Vérifie si le contenu d'une image spécifique est disponible localement.
int()	Tronque un nombre décimal pour en faire un entier.
length()	Renvoie le nombre de caractères de la chaîne ou variable spécifiée.
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 Adobe ColdFusion®, CGI ASP, PHP ou Perl, et définit les valeurs des 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 ASP, PHP ou 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.

Function	Description
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 qui déclenche un autre 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.
« random() » à la page 24	Renvoie un entier aléatoire.
removeMovieClip()	Supprime le clip spécifié initialement créé à l'aide de duplicateMovieClip().
set()	Associe une valeur à une variable.
setProperty()	Modifie la valeur des propriétés d'un 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 désigné 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 des fichiers SWF.
trace()	Evalue l'expression et affiche les résultats dans le panneau Sortie en mode de test.
unloadMovie()	Supprime de Flash Lite le clip qui a été chargé par l'intermédiaire de la fonction loadMovie(), loadMovieNum() ou duplicateMovieClip().
unloadMovieNum()	Supprime d'un niveau de Flash Lite le clip qui a été chargé par l'intermédiaire de la fonction loadMovie(), loadMovieNum() ou duplicateMovieClip().

call()

Disponibilité
Flash Lite 1.0.

Utilisation
```
call(frame)

call(movieClipInstance:frame)
```

Opérandes
`bloc` Etiquette ou numéro d'une image dans le scénario.

`movieClipInstance` 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 prendre deux formes :

- La forme par défaut exécute le script sur l'image spécifiée du scénario pour lequel la fonction `call()` a été exécutée, sans déplacer la tête de lecture jusqu'à cette image.
- La forme à occurrence de clip spécifiée exécute le script sur l'image désignée de l'occurrence de clip, sans déplacer la tête de lecture jusqu'à cette image.

  Remarque : *La fonction `call()` agit de manière synchronisée ; tout code ActionScript suivant une fonction `call()` ne s'exécute que lorsque l'intégralité du code ActionScript est exécutée au niveau de l'image spécifiée.*

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

```
// to execute functions in frame with label "myScript"
thisFrame = "myScript";
trace ("Calling the script in frame: " add thisFrame);

// to execute functions in any other frame on the same timeline
call("myScript");
```

chr()

Disponibilité
Flash Lite 1.0.

Utilisation
`chr(number)`

Opérandes
`nombre` 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);// Output: A
```

duplicateMovieClip()

Disponibilité
Flash Lite 1.0.

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

Opérandes
`target` Chemin cible du clip à dupliquer.

`newname` Identificateur unique du clip dupliqué.

`depth` Niveau de profondeur unique pour le clip dupliqué. Le niveau de profondeur correspond à l'ordre d'empilement des clips dupliqués. Cet ordre d'empilement est très proche de 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 remplacer les clips figurant aux niveaux de profondeur utilisés.

Description
Fonction ; crée une occurrence de clip pendant la lecture du fichier SWF. Ne renvoie aucune valeur. 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()`. Référencez le nouveau clip en utilisant la chaîne transmise à l'aide de l'opérande *newname*.

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

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

L'exemple suivant utilise `duplicateMovieClip()` dans une boucle `for` pour créer plusieurs nouveaux clips à la fois. Une variable d'indice détermine la profondeur d'empilement occupée la plus élevée. Chaque nom de clip dupliqué contient un suffixe numérique qui correspond à sa 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` 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`.

Vous pouvez utiliser `eval()` pour simuler des tableaux ou définir de façon dynamique la valeur d'une variable et l'extraire.

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'affecte à `y` :

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

L'exemple suivant explique comment il est possible de simuler un tableau :

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

getProperty()

Disponibilité

Flash Lite 1.0.

Utilisation

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

Opérandes

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

`property` Propriété d'un clip.

Description

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

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

```
xPos = getProperty("person_mc", _x);
trace (xPos);               // output: -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);// Output: milliseconds of time movie has been playing
```

getURL()

Disponibilité
Flash Lite 1.0.

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

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

window Paramètre facultatif qui spécifie la fenêtre ou l'image HTML dans laquelle le document doit être chargé.

Remarque : *Le paramètre window n'est pas spécifié pour les applications Flash Lite parce que les navigateurs des téléphones portables ne prennent pas en charge les fenêtres multiples.*

Vous pouvez entrer une chaîne vide ou le nom d'une fenêtre spécifique, ou sélectionner l'un 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` Méthode `GET` ou `POST` pour l'envoi de 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.myserver.com), vous devez disposer d'une connexion réseau.

Flash Lite 1.0 reconnaît uniquement les protocoles HTTP, HTTPS, mailto et tel. Flash Lite 1.1 reconnaît ces protocoles, plus les protocoles de fichiers SMS et MMS.

Flash Lite transmet l'appel au système d'exploitation et celui-ci le gère avec l'application enregistrée par défaut pour le protocole spécifié.

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

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

Exemple

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

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

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

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

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

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

Vous pouvez affecter une fonction de bouton pour ouvrir une fenêtre de composition de message électronique avec les champs de texte `address`, `subject` et `body` déjà renseignés. Utilisez l'une des méthodes suivantes pour affecter une fonction de bouton : La méthode 1 permet d'encoder les caractères Shift-JIS ou anglais ; la méthode 2 permet d'encoder uniquement les caractères anglais.

Méthode 1 : Définissez les variables pour chacun des paramètres souhaités, comme dans cet exemple :

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

Méthode 2 : Définissez tous les paramètres de la fonction `getURL()`, comme dans cet exemple :

```
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 produit un encodage automatique des URL ; la méthode 2 préserve les espaces dans les chaînes. Par exemple, les chaînes produites avec la méthode 1 se présentent comme suit :

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

En revanche, la méthode 2 produit 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 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 MMS :

```
// mms example
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 :

```
// file protocol example
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` Chaîne facultative qui spécifie le nom de la séquence cible de la tête de lecture.

`bloc` Nombre représentant le numéro d'image ou 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 d'autres objets du document.

Exemple
Dans l'exemple suivant, lorsque l'utilisateur clique sur un bouton auquel la fonction `gotoAndPlay()` est affectée, la tête de lecture se place sur l'image 16 de la séquence actuelle et commence à lire le fichier SWF :

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

gotoAndStop()

Disponibilité
Flash 1.0.

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

Opérandes
`scene` Chaîne facultative qui spécifie le nom de la séquence cible de la tête de lecture.

`bloc` Nombre représentant le numéro d'image ou chaîne représentant l'étiquette de l'image cible de la tête de lecture.

Description
Fonction ; place la tête de lecture sur l'image spécifiée d'une séquence et l'arrête. 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 d'autres objets du document.

Exemple

Dans l'exemple suivant, lorsque l'utilisateur clique sur un bouton auquel la fonction `gotoAndStop()` est affectée, la tête de lecture se place sur l'image 5 de la séquence actuelle 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` Chaîne facultative qui spécifie le nom de la séquence à charger.

`bloc` Numéro ou étiquette d'image à charger avant l'exécution de l'instruction suivante.

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

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 contrôler 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 l'image est chargée, la commande `trace()` affiche le message « frame number 10 is loaded » dans le panneau Sortie. La variable de sortie est également définie avec une variable `frame loaded: 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
`valeur` Nombre ou chaîne à tronquer pour en faire un entier.

Description
Fonction ; tronque un nombre décimal pour en faire un entier.

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

```
distance = 6.04 - 3.96;
//trace ("distance = " add distance add " and rounded to:" add int(distance));
// Output: distance = 2.08 and rounded to: 2
myDistance = "3.8";
//trace ("myDistance = " add int(myDistance));
// Output: 3
```

length()

Disponibilité
Flash Lite 1.0.

Utilisation
`length(expression)`

`length(variable)`

Opérandes
`expression` Chaîne.

`variable` Nom d'une variable.

Description
Fonction de la chaîne ; renvoie le nombre de caractères de la chaîne ou variable spécifiée.

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 contient au moins six caractères :

```
email = "someone@example.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` 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` 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` Paramètre de chaîne facultatif qui spécifie une méthode HTTP d'envoi des variables. Ce paramètre doit correspondre à la chaîne `GET` ou `POST`. En l'absence de variable à envoyer, omettez ce paramètre. La méthode `GET` ajoute les variables à la fin de l'URL et est utilisée lorsque les variables sont peu nombreuses. La méthode `POST` place les variables dans un en-tête HTTP distinct et est utilisée pour des variables longues de type chaîne.

Description

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

Pour charger un fichier SWF à un niveau spécifique, utilisez la fonction `loadMovieNum()` à la place de `loadMovie()`.

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

Utilisez la fonction `unloadMovie()` pour 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");
// Equivalent statement: loadMovie("circle.swf", _level0.mySquare);
```

Voir aussi

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

loadMovieNum()

Disponibilité
Flash Lite 1,1.

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

Opérandes
`url` Chaîne qui spécifie l'URL absolue ou relative du fichier SWF à charger. Un chemin relatif doit être relatif par rapport au fichier SWF au 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 fichiers ne doivent pas inclure de spécifications de dossier ou de lecteur de disque.

`level` Entier qui spécifie le niveau de chargement du fichier SWF dans Flash Lite.

`method` Paramètre de chaîne facultatif qui spécifie une méthode HTTP d'envoi des variables. Il doit posséder la valeur `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 au lieu d'un niveau, utilisez la fonction `loadMovie()` à la place de `loadMovieNum()`.

Flash Lite empile les différents niveaux en commençant par le niveau 0. Ces niveaux sont comme 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 n'importe quel numéro de niveau. Vous pouvez charger un fichier SWF dans un niveau qui en contient déjà un ; le nouveau fichier SWF remplacera 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 la cadence d'images, la couleur d'arrière-plan et la taille d'image de tous les autres fichiers SWF chargés.

La fonction `unloadMovieNum()` permet de supprimer les fichiers SWF 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` 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` Chemin cible d'un clip devant recevoir les variables chargées.

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

Description

Fonction ; lit les données dans un fichier externe, tel qu'un fichier texte ou du texte généré par un script ColdFusion, CGI, ASP, PHP ou 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=Adobe&address=600+Townsend&city=San+Francisco&zip=94103
```

Pour charger des variables dans un niveau spécifique, utilisez la fonction `loadVariablesNum()` à la place de `loadVariables()`.

Exemple

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

```
// load variables from text file on local file system (Symbian Series 60)
on(release, keyPress "1") {
    filePath = "file://c:/documents/flash/myApp/myvariables.txt";
    loadVariables(filePath, _root);
}

// load variables (from server) into a movieclip
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` Chaîne représentant une URL absolue ou relative par rapport à l'emplacement des variables à charger. Si le fichier SWF qui émet cet appel s'exécute sur un navigateur Web, l'*url* doit appartenir au même domaine que le fichier SWF. Pour plus d'informations, consultez la section Description suivante.

`level` Entier qui spécifie le niveau de Flash Lite devant recevoir les variables.

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

Description
Fonction ; lit les données dans un fichier externe, tel qu'un fichier texte ou du texte généré par un script ColdFusion, CGI ASP, PHP ou 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. L'exemple de phrase suivant définit plusieurs variables :

```
company=Adobe&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, utilisez la fonction `loadVariables()` à la place de `loadVariablesNum()`.

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

mbchr()

Disponibilité
Flash Lite 1.0.

Utilisation
`mbchr(number)`

Opérandes

nombre 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 caractères multi-octets équivalents :

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

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

Voir aussi

mblength(),mbsubstring()

mblength()

Disponibilité

Flash Lite 1.0.

Utilisation

```
mblength(string)
```

Opérandes

chaîne 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));
// Output: string length = 2
```

Voir aussi

mbchr(),mbsubstring()

mbord()

Disponibilité
Flash Lite 1.0.

Utilisation
`mbord(character)`

Opérandes
`character` Caractère à convertir en nombre multi-octets.

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

Exemple
Les exemples suivants convertissent les caractères de la variable `myString` en nombres multi-octets :

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

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

Voir aussi
`mbchr()`, `mbsubstring()`

mbsubstring()

Disponibilité
Flash Lite 1.0.

Utilisation
`mbsubstring(value, index, count)`

Opérandes
`valeur` Chaîne multi-octets à partir de laquelle il convient d'extraire une nouvelle chaîne multi-octets.

`index` Numéro du premier caractère à extraire.

`count` 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 à partir 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));// Output: $1
```

Voir aussi
`mbchr()`

nextFrame()

Disponibilité
Flash Lite 1.0.

Utilisation
`nextFrame()`

Opérandes
Aucun.

Description
Fonction ; place la tête de lecture sur 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 la prochaine image 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 prochaine séquence :

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

Voir aussi
`prevScene()`

Number()

Disponibilité
Flash Lite 1.0.

Utilisation
`Number(expression)`

Opérandes
`expression` 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 dernière ligne affiche le résultat lorsque `Number()` est appelé sur une valeur booléenne.

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

trace (myResult);               // Output: 60

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

on()

Disponibilité
Flash Lite 1.0.

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

Opérandes
`statement(s)` Instructions à exécuter lorsqu'un événement *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 survole le bouton.
- `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 de la souris 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'événements ; spécifie l'événement de type utilisateur ou pression de touche qui déclenche une fonction. Certains 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` 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"));// Output: multibyte number = 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 saisit `Steve`, la fonction `play()` est appelée et la tête de lecture avance au sein du scénario. Si l'utilisateur saisit un élément autre que `Steve`, le fichier SWF n'est pas lu et un champ 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 ; 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.

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) {
    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 la séquence précédente :

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

Voir aussi
`nextScene()`

random()

Disponibilité
Flash Lite 1.0.

Utilisation
`random(value)`

Opérandes
`valeur` Entier.

Description
Fonction ; renvoie un entier aléatoire compris entre 0 et un entier inférieur au nombre spécifié dans le paramètre *value*.

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

```
//pick random number between 0 and 5
myNumber = random(5);
trace (myNumber);           // Output: could be 0,1,2,3,4

//pick random number between 5 and 10
myNumber = random(5) + 5;
trace (myNumber);           // Output: could be 5,6,7,8,9
```

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.

```
// select random name from list
myNames1 = "Mike";
myNames2 = "Debbie";
myNames3 = "Logan";

ran = random(3) + 1;
ranName = "myNames" add ran;
trace (eval(ranName));// Output: will be mike, debbie, or logan
```

removeMovieClip()

Disponibilité
Flash Lite 1.0.

Utilisation
`removeMovieClip(target)`

Opérandes
`target` Chemin cible d'une occurrence de clip créée à l'aide de `duplicateMovieClip()`.

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

Exemple
L'exemple suivant supprime le clip dupliqué appelé `second_mc` :

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

Voir aussi
`duplicateMovieClip()`

set()

Disponibilité
Flash Lite 1.0.

Utilisation
`set(variable, expression)`

Opérandes
`variable` Identificateur devant contenir la valeur du paramètre *expression*.

`expression` 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 `orig_x_pos` qui stocke la position d'origine de l'axe *x* du clip `ship` pour réinitialiser ultérieurement le bateau à sa position de départ 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` Chemin du nom d'occurrence du clip dont la propriété doit être définie.

`property` Propriété à définir.

`valeur` Nouvelle valeur littérale de la propriété.

`expression` Equation 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` 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, convertit la valeur 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
`chaîne` Chaîne à partir de laquelle il convient d'extraire la nouvelle chaîne.

`index` Numéro du premier caractère à extraire.

`count` 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 un tandis que les méthodes de la classe String sont de base zéro.

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

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

tellTarget()

Disponibilité
Flash Lite 1.0.

Utilisation

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

Opérandes

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

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

Description

Fonction ; applique les instructions spécifiées dans le paramètre *statement(s)* au scénario désigné 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 lire 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 lire les clips sur la scène ou obliger 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 elle est associée à une fonction `stop()` : elle n'est donc pas visible sur la scène. Lorsque l'utilisateur appuie 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 des fichiers 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` 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 les résultats dans le panneau Sortie en mode de 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 dans les paramètres de publication 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` Chemin cible d'un clip.

Description

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

Exemple

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

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

Voir aussi

`loadMovie()`

unloadMovieNum()

Disponibilité

Flash Lite 1.0.

Utilisation

```
unloadMovieNum(level)
```

Opérandes

`level` Niveau (`_level`*N*) d'une animation chargée.

Description

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

Normalement, Flash Lite affiche un fichier SWF, puis se ferme. La fonction `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 3 : Propriétés de Flash Lite

Cette section décrit les propriétés reconnues par le logiciel Macromedia Flash Lite 1.x d'Adobe. 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 texte ayant le focus.
`_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 `_level`*N*. 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 `_level`***N*** pour cibler un fichier SWF chargé au niveau affecté par *N*.
`maxscroll`	Indique le numéro de la première ligne de texte visible dans un champ texte défilant lorsque la dernière ligne du champ est également visible.
`_name`	Renvoie le nom d'occurrence d'un clip. S'applique uniquement aux clips et non au scénario principal.
`_rotation`	Renvoie la rotation du clip, en degrés, par rapport à son orientation d'origine.
`scroll`	Contrôle l'affichage des informations dans un champ texte associé à une variable. La propriété `scroll` définit l'emplacement à partir duquel le champ texte commence à afficher le contenu ; une fois l'emplacement défini, Flash Lite le met à jour lorsque l'utilisateur fait défiler le champ 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`	Définit le ***pourcentage*** de redimensionnement horizontal d'un clip tel qu'il est appliqué à partir de son point d'alignement.
`_y`	Contient un entier qui définit la coordonnée *y* d'un clip par rapport aux coordonnées locales du clip parent.
`_yscale`	Définit le ***pourcentage*** de redimensionnement vertical du clip tel qu'il est appliqué à partir de son point d'alignement.

/ (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 de la propriété `_root` dans Macromedia Flash 5.

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

Exemple 1 : Variable `car` du scénario principal :

```
/:car
```

Exemple 2 : Variable `car` de l'occurrence de clip mc1 du scénario principal :

```
/mc1/:car
```

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

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

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

```
mc2/:car
```

_alpha

Disponibilité
Flash Lite 1.0.

Utilisation
```
my_mc:_alpha
```

Description
Propriété ; 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 de bouton définit la propriété `_alpha` du clip `my_mc` sur 30 % lorsque l'utilisateur clique sur le bouton :

```
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 texte ayant le focus. La valeur par défaut `true` affiche un rectangle jaune autour du bouton ou champ texte ayant le focus lorsque l'utilisateur appuie sur la flèche Haut ou Bas de son téléphone ou périphérique portable pour naviguer entre 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) ; 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, est chargé et 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 `loader` 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) ; hauteur du clip, en pixels.

Exemple
L'exemple suivant de 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 la valeur 0 pour annuler 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 ; référence au scénario racine de `_level`*N*. 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 `_level`*N* 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);

// at least 1 frame later
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 la première ligne de texte visible dans un champ 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 texte. Cette propriété peut être récupérée mais pas modifiée.

Exemple

Le code suivant, qui fait défiler le champ texte `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++;
    }
}
```

Voir aussi

`scroll`

_name

Disponibilité
Flash Lite 1.0.

Utilisation
`my_mc:_name`

Description
Propriété ; nom d'occurrence du clip spécifié par *my_mc*. 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 hors de cette plage sont ajoutées ou soustraites de 360 pour obtenir une valeur comprise dans la plage. Par exemple, l'instruction `my_mc:_rotation = 450` est identique à `my_mc:_rotation = 90`.

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

Exemple
Le code suivant fait défiler le champ texte `myText` d'une ligne vers le haut chaque fois que l'utilisateur appuie 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 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 frames later in the main timeline
if (_level1._framesloaded >= _level1._totalframes) {
    tellTarget("_level1/") {
        gotoAndStop("myLabel");
    }
} else {
    // loop...
}
```

_visible

Disponibilité
Flash Lite 1.0.

Utilisation
`my_mc:_visible`

Description
Propriété ; 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 si cette action est explicitement définie.

Exemple
Le code suivant désactive le clip `my_mc` lorsque l'utilisateur appuie sur la touche 3, et l'active lorsqu'il 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 ; 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, il se trouve dans le système de coordonnées local de celui 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 identique. Les coordonnées du clip renvoient à la position du point d'alignement.

Exemple
L'exemple suivant redéfinit 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 ; définit le *pourcentage* de redimensionnement horizontal du clip tel qu'il est appliqué à partir de son point d'alignement. Le point d'alignement par défaut est (0, 0).

Le redimensionnement du système de coordonnées local affecte les paramètres des propriétés _x et _y, définis en pixels. 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 inférieur de moitié à celui qui serait appliqué si le clip était défini sur 100 %.

Exemple

L'exemple suivant redéfinit 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 ; 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, il se trouve dans le système de coordonnées local de celui 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 identique. Les coordonnées du clip renvoient à la position du point d'alignement.

Exemple

Le code suivant définit la coordonnée *y* du clip my_mc 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 ; définit le *pourcentage* de redimensionnement vertical du clip tel qu'il est appliqué à partir de son point d'alignement. Le point d'alignement par défaut est (0, 0).

Le redimensionnement du système de coordonnées local affecte les paramètres des propriétés `_x` et `_y`, définis en pixels. 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 inférieur de moitié 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 4 : Instructions de Flash Lite

Cette section décrit la syntaxe et l'utilisation des instructions ActionScript du logiciel Macromedia Flash Lite 1.x d'Adobe. Les instructions sont des éléments de langage qui effectuent ou spécifient une action. Les instructions sont présentées dans le tableau suivant :

Instruction	Description
`break`	Oblige Flash à ignorer le reste du corps de la boucle, arrête l'action de la boucle et exécute l'instruction qui suit l'instruction de bouclage.
`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 normalement à la fin de la boucle.
`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 le paramètre ***`condition`*** renvoie `true`, l'instruction ***`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 ignore les instructions entre accolades et exécute les instructions qui suivent ces accolades.
`switch`	Comme pour l'instruction `if`, l'instruction `switch` teste une condition et exécute des instructions si cette condition renvoie `true`.
`while`	Teste une expression et exécute une instruction ou une 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 précis au sein d'une instruction `switch`. L'instruction `break` oblige Flash Lite à ignorer le reste du corps de la boucle, arrête l'action de la boucle et exécute l'instruction qui suit l'instruction de bouclage. Lorsque vous utilisez l'instruction `break`, l'interprète d'ActionScript ignore le reste des instructions de ce bloc `case` et passe à la première instruction qui suit 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.

`instructions` 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()` qui suit `case 1` s'exécute ; si le paramètre `myNum` renvoie 2, l'instruction `trace()` qui suit `case 2` s'exécute, etc. Si aucune expression `case` ne correspond au paramètre `number`, l'instruction `trace()` qui suit 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 exécutée 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 normalement à la fin de la boucle. Elle n'a aucun effet en dehors d'une boucle.

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

- Dans une boucle `for`, l'instruction `continue` oblige l'interprète de Flash à ignorer le reste du corps de la boucle et à atteindre l'évaluation de la post-expression `for` de la boucle.

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`, 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 `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);                // output: 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);
}
```

Paramètres

`condition` Expression qui renvoie `true` ou `false`.

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

`f-statement(s)` Autre série d'instructions à exécuter si la condition renvoie `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. Dans un exemple réel, le code devrait 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` Expression qui renvoie `true` ou `false`.

`statement(s)` 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'interprète de 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 `elseif` pour créer des arborescences logiques dans vos scripts.

Exemple

L'exemple suivant utilise les instructions `else si` 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` Expression à évaluer avant d'amorcer la séquence de bouclage ; généralement une expression d'affectation.

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

`next` Expression à évaluer après chaque itération de boucle ; généralement une expression d'affectation utilisant l'opérateur d'incrémentation (++) ou de décrémentation (--).

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

Description

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

Certaines propriétés ne peuvent pas être énumérées par l'instruction `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` Expression qui renvoie `true` ou `false`.

`statement(s)` 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 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` 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` 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 `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()` qui suit `case 1` s'exécute ; si le paramètre `myNum` renvoie 2, l'instruction `trace()` qui suit `case 2` s'exécute, etc. Si aucune expression `case` ne correspond au paramètre `number`, l'instruction `trace()` qui suit 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);
}
```

Paramètres

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

`statement(s)` Instructions à exécuter lorsque la condition renvoie `true`.

Description

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

Avant d'exécuter le bloc d'instructions, la condition est testée ; si le test renvoie `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 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'a plus la valeur `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 le paramètre *condition* renvoie `true` ou une valeur convertie en valeur booléenne `true`, telle qu'un nombre autre que 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 5 : Opérateurs de Flash Lite

Cette section décrit la syntaxe et l'utilisation des opérateurs ActionScript du logiciel Macromedia Flash Lite 1 d'Adobe. Toutes les entrées sont classées par ordre alphabétique. Toutefois, certains opérateurs sont des symboles et sont classés par rapport à leur description.

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 de ***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 les balises ouvrante (/*) et fermante (*/) de commentaires est interprété en tant que commentaire et ignoré par l'interprète 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'interprète d'ActionScript.
`?: (conditionnel)`	Oblige Flash Lite à évaluer ***expression1***, et si la valeur de ***expression1*** est `true`, l'opérateur renvoie la valeur de ***expression2*** ; sinon, la valeur de ***expression3*** est renvoyée.
`-- (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 incorporés (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 renvoie `false`.

Opérateur	Description
`! (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*** divisé 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.
`+ (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, cet 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*** (renvoie `true`) ou si ***expression1*** est inférieure à ***expression2*** (renvoie `false`).
`<> (inégalité numérique)`	Teste l'égalité ; si ***expression1*** est égale à ***expression2***, renvoie `false`.
`< (numérique inférieur à)`	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`.
`<= (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` ; renvoie `false` dans le cas contraire.
`() (parenthèses)`	Regroupent un ou plusieurs paramètres, évaluent les expressions de façon séquentielle ou entourent un ou plusieurs paramètres et les transmettent 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 qu'ils doivent être traités en tant que ***chaîne*** et non 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 représentant ***expression2*** ; renvoie `false` dans le cas contraire.
`gt (chaîne supérieure à)`	Compare la chaîne représentant ***expression1*** à celle représentant ***expression2*** et renvoie `true` si ***expression1*** est supérieure à ***expression2*** ; renvoie `false` dans le cas contraire.

Opérateur	Description
ge (chaîne supérieure ou égale à)	Compare la chaîne représentant ***expression1*** à celle représentant ***expression2*** et renvoie une valeur `true` si ***expression1*** est supérieure ou égale à ***expression2***; renvoie `false` dans le cas contraire.
ne (inégalité de chaîne)	Compare la chaîne représentant ***expression1*** à celle représentant ***expression2*** et renvoie `true` si ***expression1*** n'est pas égale à ***expression2***; renvoie `false` dans le cas contraire.
lt (chaîne inférieure à)	Compare la chaîne représentant ***expression1*** à celle représentant ***expression2*** et renvoie une valeur `true` si ***expression1*** est inférieure à ***expression2***; renvoie `false` dans le cas contraire.
le (chaîne inférieure ou égale à)	Compare la chaîne représentant ***expression1*** à celle représentant ***expression2*** et renvoie une valeur `true` si ***expression1*** est inférieure ou égale à ***expression2***; renvoie `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 chaînes pour générer la chaîne *catalog*.

```
conStr = "cat" add "alog";
trace (conStr);// output: 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 d'addition (+=).

Exemple

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

```
x = 5;
y = 10;
x += y;
trace(x);// output: 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 gagné la partie. Les variables `turns` et `score` sont mises à jour lorsqu'un joueur prend un tour ou marque des points durant la partie. Le script 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!");
}
// output: You Win the Game!
```

Voir aussi

`&& (AND logique)`

= (affectation)

Disponibilité

Flash Lite 1.0.

Utilisation

`expression1 = expression2`

Opérandes

`expression1` Variable ou propriété.

`expression2` 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

`commentaire` Tout caractère.

Description

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

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

Ainsi, lorsque vous utilisez une balise ouvrante de commentaires (/*), la première balise fermante (*/) termine ce commentaire, quel que soit le nombre de balises ouvrantes (/*) 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 parenthèses `()` et montre 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); // output: 4
```

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

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

L'exemple suivant utilise l'opérateur virgule (,) sans l'opérateur parenthèses `()` et montre que l'opérateur virgule évalue de 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); // output: 4
trace(z); // output: 1
```

L'exemple suivant est identique à l'exemple précédent, à l'exception de l'ajout de l'opérateur parenthèses `()` ; il montre 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); // output: 6
trace(z); // output: 1
```

Voir aussi

for, () (parenthèses)

// (commentaire)

Disponibilité

Flash Lite 1.0

Utilisation

```
// comment
```

Opérandes

`commentaire` 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'interprète d'ActionScript.

Exemple

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

```
// Record the X position of the ball movie clip.
ballX = ball._x;
// Record the Y position of the ball movie clip.
ballY = ball._y;
// Record the X position of the bat movie clip.
batX = bat._x;
// Record the Y position of the bat movie clip.
batY = bat._y;
```

Voir aussi
`/* (bloc de commentaires)`

?: (conditionnel)

Disponibilité
Flash Lite 1.0.

Utilisation
`expression1 ? expression2 : expression3`

Opérandes
`expression1` 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`, renvoie la valeur de *expression2* ; sinon, renvoie la valeur de *expression3*.

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);// output: 5
```

–– (décrément)

Disponibilité
Flash Lite 1.0.

Utilisation
`--expression`

`expression--`

Opérandes
Aucun.

Description
Opérateur (arithmétique) ; 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;
// The bWidth value is equal to 2.
```

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

```
aWidth = 3;
bWidth = aWidth--;
// The bWidth value is equal to 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;
// The expression x now contains the value 5.
```

. (point)

Disponibilité

Flash Lite 1.0.

Utilisation

instancename.variable

instancename.childinstance.variable

Opérandes

instancename Nom d'occurrence d'un clip.

childinstance Occurrence de clip qui est un enfant d'un autre clip ou qui y est imbriquée.

variable 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) ; opérateur unaire de pré et post-décrémentation 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 ++ en tant qu'opérateur de pré-incrémentation :

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

Ce script affiche le résultat suivant dans le panneau Sortie :

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

L'exemple suivant utilise ++ en tant qu'opérateur de post-incrémentation :

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

Ce script affiche le résultat suivant dans le panneau Sortie :

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

&& (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 sa gauche) et renvoie `false` si cette expression renvoie `false`. Si *expression1* renvoie `true`, *expression2* (l'expression située à droite de l'opérateur) est évaluée. Si *expression2* renvoie `true`, le résultat final est `true` ; sinon renvoie `false`.

Exemple
L'exemple suivant utilise l'opérateur `&&` pour vérifier si un joueur a gagné la partie. Les variables `turns` et `score` sont mises à jour lorsqu'un joueur prend un tour ou marque des points durant la partie. Le script 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 imaginaire `x` 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)

Disponibilité
Flash Lite 1.0.

Utilisation
expression1 % expression2

Opérandes
expression1, expression2 Nombres ou expressions renvoyant un nombre.

Description
Opérateur (arithmétique) ; 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. 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);// output: 2
trace (4.3 % 2.1);// output: 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 de *expression1*`%` *expression2*. Par exemple, les deux expressions suivantes sont équivalentes :

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

Exemple

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

```
x = 14;
y = 5;
trace(x %= y);// output: 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);// output: 50
```

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

```
i = 5;
x = 4 - 6;
y = i + 2;
trace(x *= y);// output: -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 pas 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 nombres à virgule flottante 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");// output: 1(true)
trace (trees == bushes);// output: 1(true)
trace (trees == shrubs);// output: 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.

Description

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);// output: 1(true)
trace(1 > 2);// output: 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 ou égale à *expression2* (renvoie `true`) ou si *expression1* est inférieure à *expression2* (renvoie `false`).

Exemple

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

```
trace(3.14 >= 2);// output: 1(true)
trace(3.14 >= 4);// output: 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 égale à *expression2*, le résultat est `false`. Comme pour l'opérateur d'égalité (==), la définition de l'*égalité* dépend des types de données comparés :

- Les 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);// output: 1(true)
trace(trees <> B);// output: 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);// output: 1(true)

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

Voir aussi
`lt (chaîne inférieure à)`

<= (numérique inférieur ou égal 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 ou égale à *expression2*. Le cas échéant, l'opérateur renvoie `true` ; renvoie `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);// output: 1(true)
trace(2 <= 2);// output: 1(true)
trace (10 <= 3);// output: 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.

Opérandes

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

Description

Opérateur ; regroupent un ou plusieurs paramètres, évaluent les expressions de façon séquentielle ou entourent un ou plusieurs paramètres et les transmettent en tant que paramètres à une fonction en dehors des parenthèses.

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

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

Exemple

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

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

Utilisation 1 : 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)); // writes 45
trace(2 + (3 * (4 + 5))); // writes 29
trace(2 + (3 * 4) + 5); // writes 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 qu'ils doivent être traités en tant que *chaîne* et non 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 pas 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);// output: 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 représentant *expression2* ; renvoie `false` dans le cas contraire.

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

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

trace("persons" eq "people");// output: 0(false)
trace("persons" eq word);// output: 1(true)
trace(figure eq 50 + 5);// output: 1(true)
trace(55.0 eq 55);// output: 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 celle représentant *expression2* et renvoie `true` si *expression1* est supérieure à *expression2* ; renvoie `false` dans le cas contraire. Les chaînes sont comparées en fonction de l'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");// output: 1(true)
trace ("cats" gt "cattle");// output: 0(false)
trace (animals gt "cats");// output: 0(false)
trace (animals gt "Cats");// output: 1(true)
trace (breeds gt "5");      // output: 1(true)
trace (breeds gt 7);        // output: 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 celle représentant *expression2* et renvoie `true` si *expression1* est supérieure ou égale à *expression2* ; renvoie `false` dans le cas contraire. Les chaînes sont comparées en fonction de l'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");// output: 0(false)
trace (animals ge "cats");// output: 1(true)
trace ("persons" ge "people");// output: 1(true)
trace (animals ge "Cats");// output: 1(true)
trace (breeds ge "5");// output: 1(true)
trace (breeds ge 7);// output: 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 celle représentant *expression2* et renvoie `true` si *expression1* n'est pas égale à *expression2* ; renvoie `false` dans le cas contraire.

Exemple

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

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

trace ("persons" ne "people");      // output: 1(true)
trace ("persons" ne word);              // output: 0(false)
trace (figure ne 50 + 5);                   // output: 0(false)
trace (55.0 ne 55);                 // output: 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 celle représentant *expression2* et renvoie `true` si *expression1* est inférieure à *expression2* ; renvoie `false` dans le cas contraire. Les chaînes sont comparées en fonction de l'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 car la syntaxe d'ActionScript 1.0 tente de convertir le type de données entier en chaîne et renvoie `false`.

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

trace ("persons" lt "people");// output: 0(false)
trace ("cats" lt "cattle");// output: 1(true)
trace (animals lt "cats");// output: 0(false)
trace (animals lt "Cats");// output: 0(false)
trace (breeds lt "5");      // output: 0(false)
trace (breeds lt 7);        // output: 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 celle représentant *expression2* et renvoie `true` si *expression1* est inférieure ou égale à *expression2* ; renvoie `false` dans le cas contraire. Les chaînes sont comparées en fonction de l'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");// output: 0(false)
trace ("cats" le "cattle");// output: 1(true)
trace (animals le "cats");// output: 1(true)
trace (animals le "Cats");// output: 0(false)
trace (breeds le "5");      // output: 0(false)
trace (breeds le 7);        // output: 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));
// output: -5.
```

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

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

Le résultat est 3, qui correspond à un entier.

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

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

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

-= (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 retournée.

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);// output: -1
```

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

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

Chapitre 6 : Eléments de langage spécifiques de Flash Lite

Cette section présente les variables et fonctionnalités de la plate-forme reconnues par Macromedia Flash Lite 1.1 d'Adobe, 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 de 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 diffuser des sons au format 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 (MPEGAudio Layer 3).
`_capSMAF`	Indique si le périphérique peut diffuser des fichiers multimédias 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()`.
`_capStreamSound`	Indique si le périphérique peut diffuser des sons en flux continu (synchronisés).
`_cap4WayKeyAS`	Indique si Flash Lite exécute les expressions ActionScript liées aux gestionnaires d'événements clés associés aux touches Droite, Gauche, Haut et Bas.
`$version`	Contient le numéro de version de Flash Lite.
`fscommand()`	Fonction utilisée pour 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 l'environnement hôte, par exemple le système d'exploitation du téléphone ou du périphérique.
`fscommand2()`	Fonction utilisée pour exécuter les commandes de ce tableau, sauf `fscommand()`.
`Escape`	Code une chaîne arbitraire dans 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 actuel de la batterie.

Elément de langage	Description
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 servant à exécuter Flash Lite.
GetDeviceID	Définit un paramètre qui représente l'identificateur unique du périphérique (par exemple, le numéro de série).
GetFreePlayerMemory	Renvoie la quantité de mémoire heap, en kilo-octets, disponible actuellement pour Flash Lite.
GetLanguage	Définit un paramètre qui identifie la langue utilisée par le périphérique.
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 maximum de la batterie 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 qui indique l'état de la connexion réseau actuelle.
GetNetworkName	Définit un paramètre reprenant le nom du réseau actif.
GetNetworkRequestStatus	Renvoie une valeur indiquant l'état de la requête HTTP la plus récente.
GetNetworkStatus	Renvoie une valeur indiquant l'état 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, ce qui décrit de façon générale la classe du périphérique. Pour les périphériques disposant de systèmes d'exploitation ouverts, cet identificateur correspond généralement au nom et à la version du système d'exploitation.
GetPowerSource	Renvoie une valeur qui indique si l'alimentation vient de la batterie ou d'une source externe.
GetSignalLevel	Renvoie la force du signal actuel sous forme de valeur numérique.
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 la mémoire heap, en kilo-octets, affectée à Flash Lite.
GetVolumeLevel	Renvoie le niveau de volume actuel du périphérique sous forme de valeur numérique.
Quit	Entraîne l'arrêt du lecteur Flash Lite et ferme ce programme.
ResetSoftKeys	Rétablit les valeurs d'origine des touches programmables.
SetInputTextType	Spécifie le mode d'ouverture du champ 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	Active la fonctionnalité de vibration du téléphone.
StopVibrate	Arrête la vibration actuelle, si nécessaire.
Unescape	Décode une chaîne arbitraire qui a été codée pour garantir un transfert sécurisé sur le réseau afin de revenir à son format normal non codé.

Fonctionnalités

Cette section présente 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 les données sons composites. Dans l'affirmative, cette variable est définie et prend la valeur 1. Sinon, elle reste non définie.

Par exemple, un fichier Flash simple peut contenir le même son représenté aux formats MIDI et MFi. Le lecteur lit ensuite les données au format requis en fonction du format pris en charge par le périphérique. Cette variable définit si le lecteur Flash Lite prend en charge cette fonctionnalité sur le combiné en cours d'utilisation.

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

```
useCompoundSound = _capCompoundSound;

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

_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()`. Dans l'affirmative, cette variable est définie et prend la valeur 1. Sinon, elle reste non 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()`. Dans l'affirmative, cette variable est définie et prend la valeur 1. Sinon, elle reste non 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 MFi (Melody Format for i-mode). Dans l'affirmative, cette variable est définie et prend la valeur 1. Sinon, elle reste non définie.

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

```
canMFi = _capMFi;

if (canMFi == 1) {
    // send movieclip buttons to frame with buttons that trigger events sounds
    tellTarget("buttons") {
            gotoAndPlay(2);
        }
}
```

_capMIDI

Disponibilité
Flash Lite 1.1.

Utilisation
`_capMIDI`

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

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

```
canMIDI = _capMIDI;

if (canMIDI == 1) {
    // send movieclip buttons to frame with buttons that trigger events sounds
    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()`. Dans l'affirmative, cette variable est définie et prend la valeur 1. Sinon, elle reste non définie.

Exemple
L'exemple suivant définit `canMMS` sur 1 dans Flash Lite 1.1, mais la laisse non définie 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) {
        // send an MMS
        myMessage = "mms:4156095555?body=sample mms message";
    } else {
        // send an 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 MP3 (MPEG Audio Layer 3). Dans l'affirmative, cette variable est définie et prend la valeur 1. Sinon, elle reste non 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 diffuser des fichiers multimédias au format SMAF (Synthetic music Mobile Application Format). Dans l'affirmative, cette variable est définie et prend la valeur 1. Sinon, elle reste non définie.

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) {
    // send movieclip buttons to frame with buttons that trigger events sounds
    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()`. Dans l'affirmative, cette variable est définie et prend la valeur 1. Sinon, elle reste non 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) {
        // send an 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 diffuser des sons en flux continu (synchronisés). Dans l'affirmative, cette variable est définie et prend la valeur 1. Sinon, elle reste non définie.

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

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

_cap4WayKeyAS

Disponibilité
Flash Lite 1.1.

Utilisation

```
_cap4WayKeyAS
```

Description

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

Lorsque l'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 pas, l'application parcourt les contrôles Flash. Cependant, si un gestionnaire d'événements est trouvé, aucune action de navigation ne se produit pour cette touche. Par exemple, si un gestionnaire associé à la touche Bas est détecté, l'utilisateur ne peut pas procéder à la navigation.

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 les touches quadridirectionnelles, 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 String ; contient le numéro de version de Flash Lite. Cette variable indique les numéros majeur et secondaire et le numéro de création interne, qui correspond généralement à 0 dans les versions officielles.

Le numéro majeur signalé pour tous les produits Flash Lite 1.x est 5. Flash Lite 1.0 dispose d'un numéro mineur, 1. Flash Lite 1.1 a 2 pour numéro mineur.

Exemple

Dans le lecteur Flash Lite 1.1, le code 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"` Spécificateur de commandes. La commande `Launch` est la seule qui nécessite l'exécution de la fonction `fscommand()`.

"*application-path*, *arg1*, *arg2*,..., *argn*" Nom de l'application en cours de démarrage et ses paramètres, séparés par des virgules.

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

Voir aussi
`fscommand2()`

Launch

Disponibilité
Flash Lite 1.1.

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

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

"*application-path*, *arg1*, *arg2*,..., *argn*" Nom de l'application en cours de démarrage 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 en cours de démarrage et ses paramètres sont transmis sous la forme d'un argument unique.

Remarque : *Cette fonctionnalité dépend du système d'exploitation.*

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

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` Chaîne transmise à l'application hôte ou commande envoyée à Flash Lite.

`parameter1...parameterN` Liste de chaînes délimitées par des virgules transmises sous forme de paramètres à la commande spécifiée par *command*.

Description

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

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 `fscommand2()` immédiatement, alors que `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 les 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 dans 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* en cas de caractères multi-octets).

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 : échec. 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);                    // output: 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 correspondre à l'ensemble de l'écran ou non.

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

Commande	Paramètres	Valeur renvoyée
`"FullScreen"`	***size*** Peut correspondre à une variable définie ou une valeur constante de chaîne incluant l'une des valeurs suivantes : `true`(plein écran) ou `false` (n'occupe pas toute la surface de l'écran). Toute autre valeur est traitée comme la valeur `false`.	-1 : pas de 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 smallScreenMode :

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

GetBatteryLevel

Disponibilité

Flash Lite 1.1.

Description

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

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

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 : pas de prise en charge. 1 à 31 : 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 : pas de prise en charge. 1 à 12 : 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 : pas de 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

Disponibilité

Flash Lite 1.1.

Description

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

Commande	Paramètres	Valeur renvoyée
`"GetDateYear"`	Aucun.	-1 : pas de prise en charge. 0 à 9999 : année 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, GetDateMonth, GetDateWeekday

GetDevice

Disponibilité

Flash Lite 1.1.

Description

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

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

Exemple

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

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

`D506i` iTéléphone Mitsubishi 506

`DFOMA1` Téléphone Mitsubishi FOMA1.

`F506i` iTéléphone Fujitsu 506

`FFOMA1` Téléphone Fujitsu FOMA1.

`N506i` iTéléphone NEC 506

`NFOMA1` Téléphone NEC FOMA1.

`Nokia3650` Téléphone Nokia 3650.

`p506i` Téléphone Panasonic 506i.

`PFOMA1` Téléphone Panasonic FOMA1.

`SH506i` iTéléphone Sharp 506

`SHFOMA1` Téléphone Sharp FOMA1.

`SO506i` iTéléphone Sony 506

```
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

Disponibilité
Flash Lite 1.1.

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

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

Exemple

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

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

GetFreePlayerMemory

Disponibilité

Flash Lite 1.1.

Description

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

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

Exemple

L'exemple suivant définit l'état en fonction du montant de mémoire disponible :

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

Voir aussi

`GetTotalPlayerMemory`

GetLanguage

Disponibilité

Flash Lite 1.1.

Description

Définit un paramètre qui identifie la langue utilisée par le périphérique. Cette langue est renvoyée sous forme de chaîne dans une variable qui est transmise avec le 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 chaîne qui contient le nom d'une variable. Elle peut avoir l'une des valeurs suivantes : `cs` : Tchèque. `da` : Danois. `de` : Allemand. `en-UK` : Anglais britannique ou international. `en-US` : Anglais américain. `es` : Espagnol. `fi` : Finnois. `fr` : Français. `hu` : Hongrois. `it` : Italien. `ja` : 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 : pas de prise en charge. 0 : prise en charge.

Remarque : *Lorsque les téléphones japonais sont configurés pour un affichage en 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 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, par exemple `« Octobre 16, 2004 »` ou `« 16 octobre 2004 »`. Il peut s'agir du nom d'une variable ou d'une chaîne qui contient le nom d'une variable. La valeur renvoyée dans ***`longdate`*** est une chaîne à caractères multiples de longueur variable. Le formatage final dépend du périphérique et des paramètres régionaux sélectionnés.	-1 : pas de 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);                       // output: Tuesday, June 14, 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 long de la valeur de la date courante, par exemple « `10/16/2004` » ou « `16-10-2004` ». Il peut s'agir du nom d'une variable ou d'une chaîne qui contient le nom d'une variable. La valeur renvoyée dans ***shortdate*** est une chaîne à caractères multiples de longueur variable. Le formatage final dépend du périphérique et des paramètres régionaux sélectionnés.	-1 : pas de 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);                    // output: 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 courante, par exemple « `6:10:44 PM` » ou « `18:10:44` ». Il peut s'agir du nom d'une variable ou d'une chaîne qui contient le nom d'une variable. La valeur renvoyée dans ***time*** est une chaîne à caractères multiples de longueur variable. Le formatage final dépend du périphérique et des paramètres régionaux sélectionnés.	-1 : pas de 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);                                    // output: 14:30:21
```

Voir aussi

GetLocaleLongDate, GetLocaleShortDate

GetMaxBatteryLevel

Disponibilité

Flash Lite 1.1.

Description

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

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

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 : pas de prise en charge. Autres valeurs numériques : niveau maximum 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 : pas de prise en charge. Autres valeurs : niveau maximum du volume.

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

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

Voir aussi
GetVolumeLevel

GetNetworkConnectStatus

Disponibilité
Flash Lite 1.1.

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

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

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

```
connectstatus = fscommand2("GetNetworkConnectStatus");
switch (connectstatus) {
    case -1 :
        /: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 reprenant 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 chaîne qui contient le nom d'une variable. Si le réseau est enregistré et si son nom peut être déterminé, la variable ***networkname*** est définie sur le nom du réseau. Sinon, cette chaîne reste vide.	-1 : pas de prise en charge. 0 : aucun réseau n'est enregistré. 1 : le réseau est enregistré, mais le nom de réseau est inconnu. 2 : le réseau est enregistré et le nom de réseau est connu.

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

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 réseau du téléphone (indique si un réseau est enregistré et si le téléphone est en mode mobile).

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

Exemple

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

```
networkstatus = fscommand2("GetNetworkStatus");
switch(networkstatus) {
    case -1:
        /: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, ce qui décrit de façon générale la classe du périphérique. Pour les périphériques disposant de systèmes d'exploitation ouverts, cet identificateur correspond généralement au nom et à la version du système d'exploitation.

Commande	Paramètres	Valeur renvoyée
`"GetPlatform"`	***`platform`*** Chaîne devant recevoir l'identificateur de la plate-forme. Il peut s'agir du nom d'une variable ou d'une chaîne qui contient le nom d'une variable.	-1 : pas de prise en charge. 0 : prise en charge.

Exemple
L'exemple de code suivant affecte l'identificateur de la plate-forme à la variable `statusplatform` et met à jour un champ texte avec le nom générique de la plate-forme.

Les exemples suivants illustrent quelques résultats pour `myPlatform` et les classes de périphériques auxquels ils se rapportent :

`506i` iTéléphone 506

`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 vient de la batterie ou d'une source externe.

Commande	Paramètres	Valeur renvoyée
`"GetPowerSource"`	Aucun.	-1 : pas de prise en charge. 0 : le périphérique est alimenté par la batterie. 1 : le périphérique est alimenté par une source 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 la force du signal actuel sous forme de valeur numérique.

Commande	Paramètres	Valeur renvoyée
`"GetSignalLevel"`	Aucun.	-1 : pas de 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 : pas de prise en charge. 0 à 23 : heure courante.

Exemple
L'exemple suivant définit la variable `hour` sur la tranche horaire de l'heure courante ou sur -1 :

```
hour = fscommand2("GetTimeHours");
trace (hour);                              // output: 14
```

Voir aussi
`GetTimeMinutes`, `GetTimeSeconds`, `GetTimeZoneOffset`

GetTimeMinutes

Disponibilité
Flash Lite 1.1.

Description

Renvoie les minutes de l'heure courante. Il s'agit d'une valeur numérique (sans zéro de tête).

Commande	Paramètres	Valeur renvoyée
`"GetTimeMinutes"`	Aucun.	-1 : pas de prise en charge. 0 à 59 : minutes de l'heure courante.

Exemple

L'exemple suivant définit la variable `minutes` sur les minutes de l'heure courante ou sur -1 :

```
minutes = fscommand2("GetTimeMinutes");
trace (minutes);                          // output: 38
```

Voir aussi

GetTimeHours, GetTimeSeconds, GetTimeZoneOffset

GetTimeSeconds

Disponibilité

Flash Lite 1.1.

Description

Renvoie les secondes de l'heure courante. Il s'agit d'une valeur numérique (sans zéro de tête).

Commande	Paramètres	Valeur renvoyée
`"GetTimeSeconds"`	Aucun.	-1 : pas de prise en charge. 0 à 59 : secondes de l'heure courante.

Exemple

L'exemple suivant définit la variable `seconds` sur les secondes de l'heure courante ou sur -1 :

```
seconds = fscommand2("GetTimeSeconds");
trace (seconds);                          // output: 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 chaîne qui contient le nom d'une variable. Une valeur numérique positive ou négative est renvoyée, par exemple : 540: heure normale du Japon -420: heure d'été du Pacifique	-1 : pas de 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 cette dernière sur -1 :

```
status = fscommand2("GetTimeZoneOffset", "timezoneoffset");
trace (timezoneoffset);             // output: 300
```

Voir aussi

`GetTimeHours`, `GetTimeMinutes`, `GetTimeSeconds`

GetTotalPlayerMemory

Disponibilité

Flash Lite 1.1.

Description

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

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

Exemple

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

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

Voir aussi

`GetFreePlayerMemory`

GetVolumeLevel

Disponibilité

Flash Lite 1.1.

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

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

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

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

Quit

Disponibilité
Flash Lite 1.1.

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

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

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

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

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

ResetSoftKeys

Disponibilité
Flash Lite 1.1.

Description
Rétablit les valeurs d'origine des touches programmables.

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

Commande	Paramètres	Valeur renvoyée
`"ResetSoftKeys"`	Aucun.	-1 : pas de prise en charge. 0 : prise en charge.

Exemple

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

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

Voir aussi

`SetSoftKeys`

SetInputTextType

Disponibilité

Flash Lite 1.1.

Description

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

Commande	Paramètres	Valeur renvoyée
`"SetInputTextType"`	***variableName*** Nom du champ texte de saisie. Il peut s'agir du nom d'une variable ou d'une chaîne qui contient le nom d'une variable. ***type*** Une des valeurs `Numeric`, `Alpha`, `Alphanumeric`, `Latin`, `NonLatin` ou `NoRestriction`.	0 : échec. 1 : succès

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

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

Mode spécifié	Définit le processeur frontal sur l'un de ces modes, qui s'excluent mutuellement.	Si le mode retenu n'est pas pris en charge sur le périphérique actif, le processeur s'ouvre dans ce mode
Numeric	Nombres uniquement (0 à 9)	Alphanumérique
Alpha	Caractères alphabétiques uniquement (A à Z, a à z)	Alphanumérique
Alphanumérique	Caractères alphanumériques uniquement (0à 9, A à Z, a à z)	Latin

Mode spécifié	Définit le processeur frontal sur l'un de ces modes, qui s'excluent mutuellement.	Si le mode retenu n'est pas pris en charge sur le périphérique actif, le processeur s'ouvre dans ce mode
Latin	Caractères latins uniquement (alphanumérique et ponctuation)	NoRestriction
NonLatin	Caractères non latin uniquement (par exemple, Kanji et Kana)	NoRestriction
NoRestriction	Mode par défaut (ne définit par de restriction sur le processeur frontal)	

Remarque : *Tous les téléphones mobiles ne prennent pas en charge les types de champ texte de saisie suivants. Pour cette raison, vous devez valider les données du texte saisi.*

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`*** Qualité de rendu ; doit être `high`, `medium` ou `low`.	-1 : pas de 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.

Lorsque cette commande est exécutée, le fait d'appuyer sur la touche gauche génère un événement de pression de touche `PageUp` et le fait d'appuyer sur la touche droite génère un événement de pression de touche `PageDown`. Le code ActionScript associé aux événements de pression de touche `PageUp` et `PageDown` est exécuté si l'utilisateur appuie sur la touche correspondante.

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

Commande	Paramètres	Valeur renvoyée
`"SetSoftKeys"`	***left*** Texte à afficher pour la touche programmable Gauche. ***right*** 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 : pas de prise en charge. 0 : prise en charge.

Exemple

L'exemple suivant stipule que les touches programmables Gauche et Droite doivent être intitulées respectivement « Previous » et « Next » :

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

Voir aussi

`ResetSoftKeys`

StartVibrate

Disponibilité

Flash Lite 1.1.

Description

Active la fonctionnalité de vibration du téléphone. Si une vibration se produit déjà, Flash Lite arrête cette vibration avant de passer à la suivante. Les vibrations sont également interrompues en cas d'arrêt ou de pause de la lecture de l'application Flash et 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 : pas de prise en charge. 0 : la vibration a été activée. 1 : une erreur s'est produite et la vibration n'a pas pu être activée.

Exemple

L'exemple suivant tente de lancer une séquence de vibrations de 2,5 secondes à 1 seconde d'intervalle avec deux répétitions. Il affecte une valeur à la variable d'état 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 actuelle, si nécessaire.

Commande	Paramètres	Valeur renvoyée
"StopVibrate"	Aucun.	-1 : pas de prise en charge. 0 : la vibration s'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 arbitraire qui a été codée pour garantir un transfert sécurisé sur le réseau afin de 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 depuis un format prévu pour les URL vers un format normal. ***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 : échec. 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);                    // output: Hello, how are you?
```

Voir aussi

Escape

Index

Symboles

_alpha, variable 33
_cap4WayKeyAS, variable 88
_capCompoundSound, variable 83
_capEmail, variable 84
_capLoadData, variable 84
_capMFi, variable 85
_capMIDI, variable 85
_capMMS, variable 86
_capSMAF, variable 87
_capSMSx0d x0d, variable 87
_capStreamSoundx0d x0d, variable 88
_currentframe, propriété 34
_focusrect, propriété 34
_framesloaded, propriété 35
_height, propriété 35
_highquality, propriété 36
_level, propriété 36
_name, propriété 38
_Propriété scroll 38
_rotation, propriété 38
_target, propriété 39
_visible, propriété 40
_width, propriété 40
_x, propriété 41
_xscale, propriété 41
_y, propriété 42
_yscale, propriété 42
, (virgule), opérateur 59
, opérateur OR logique 66
! (NOT logique), opérateur 65
? (conditionnel), opérateur 61
. (point), opérateur 63
" " (séparateur de chaîne), opérateur 74
* (produit), opérateur 69
*= (affectation de multiplication), opérateur 68
*=, opérateur (affectation de 58
/ (barre de fraction - scénario racine), propriété 33
/ (division), opérateur 62
/* (bloc de commentaires), opérateur 58
// (commentaire), opérateur 60
/= (division), opérateur 62
\ 72, 73
\ (inégalité numérique), opérateur 72
&& (AND logique), opérateur 65
|| (OR logique), opérateur 66
% (modulo), opérateur 67
%= (affectation modulo), opérateur 67
+ (addition numérique), opérateur 69
++ (incrément), opérateur 64
+= (affectation d'addition), opérateur 56
-= (affectation de soustraction), opérateur 80
= (affectation), opérateur 58
== (égalité numérique), opérateur 70
> (supérieur à), opérateur 70
> (supérieur ou égal à), opérateur 71
$version, variable 89

A

add (concaténation de chaîne), opérateur 56
addition numérique 69
affectation d'addition, opérateur 56
affectation de division, opérateur 62
affectation de soustraction, opérateur 80
affectation modulo 67
_alpha, variable 33
AND, opérateur 65
and, opérateur 57

B

bloc de commentaires, opérateur 58
break, instruction 44

C

call 3
_cap4WayKeyAS, variable 88
_capCompoundSound, variable 83
_capEmail, variable 84
_capLoadData, variable 84
_capMFi, variable 85
_capMMS, variable 86
_capSMAF, variable 87
_capSMS, variable 87
_capStreamSoundx0d x0d, variable 88
case, instruction 45
chaîne inférieure ou égale à 78
chaîne supérieure à, opérateur 75
chaîne supérieure ou égale à 76
chr(),fonction 4
commentaires
  bloc 58
  une ligne 60
concaténation 56
conditions 50
continue, instruction 46
_currentframe, propriété 34

D

division 62
do..while, instruction 47
duplicateMovieClip(), fonction 5

E

égalité de chaîne, opérateur 75
else if, instruction 49
else, instruction 48
eq (égalité de chaîne), opérateur 75
eval(), fonction 5

F

_focusrect, propriété 34
fonctionnalité d'envoi de messages électroniques, variable 84
fonctions
  chr() 4
  duplicateMovieClip() 5
  eval() 5
  fscommand() 90
  getProperty() 6
  getTimer() 7
  getURL() 7
  gotoAndPlay() 10
  gotoAndStop() 10
  ifFrameLoaded() 11
  int() 12
  length() 12
  loadMovie() 13
  loadMovieNum() 14
  loadVariables() 15
  loadVariablesNum() 16
  mbchr() 16
  mbsubstring() 18
  nextFrame() 19
  nextScene() 19
  Number() 20

on() 21
ord() 21
play() 22
prevFrame() 22
prevScene() 23
random() 24
removeMovieClip() 24
set() 25
setProperty() 26
stop() 26
stopAllSounds() 27
String() 27
substring() 28
tellTarget() 28
toggleHighQuality() 29
trace() 29
unloadMovie() 30
unloadMovieNum() 31
for, boucle 50
for, instruction 50
_framesloaded, propriété 35
fscommand(), commande 90

G

ge (chaîne supérieure ou égale à), opérateur 76
getProperty(), fonction 6
getTimer(), fonction 7
getURL(), fonction 7
gotoAndPlay(), fonction 10
gotoAndStop(), fonction 10
gt (chaîne supérieure à), opérateur 75

H

_height, propriété 35
_highquality, propriété 36

I

if, instruction 50
ifFrameLoaded(), fonction 11
incrément, opérateur 64
inférieur à, opérateur 72
inférieur ou égal à, opérateur 73
instructions
break 44
case 45
continue 46
do..while 47
else 48
else if 49
for 50
if 50
NOT logique 65
switch 51
while 52
int(), fonction 12

L

le (chaîne inférieure ou égale à), opérateur 78
length(), fonction 12
_level, propriété 36
loadMovie(), fonction 13
loadMovieNum(), fonction 14
loadVariables(), fonction 15
loadVariablesNum(), fonction 16
lt (chaîne inférieure à), opérateur 77

M

maxscroll, propriété 37
mbchr(), fonction 16
mbsubstring(), fonction 18
MFI, son 85
MIDI, son 85
MMS, messagerie 86
modulo, opérateur 67
multiplication 69

N

_name, propriété 38
ne (inégalité de chaîne), opérateur 77
nextFrame(), fonction 19
nextScene(), fonction 19
NOT, opérateur 65
Number(), fonction 20

O

on(), fonction 21
Opérateur AND logique 65
opérateur conditionnel 61
opérateur d'inégalité 72
opérateur NOT logique 65
opérateur point 63
opérateur virgule 59
opérateurs
addition numérique 69
affectation 58
affectation d'addition 56
affectation de division 62
affectation de soustraction 80
affectation modulo 67
and 57
AND logique 65
bloc de commentaires 58
chaîne inférieure à 77
chaîne inférieure ou égale à 78
chaîne supérieure à 75
chaîne supérieure ou égale à 76
commentaire 60
concaténation de chaîne 56
conditionnel 61
division 62
égalité de chaîne 75
égalité numérique 70
incrément 64
inégalité de chaîne 77
inégalité numérique 72
modulo 67
NOT logique 65
numérique inférieur à 72
numérique inférieur ou égal à 73
OR logique 66
point 63
produit 69
séparateur de chaîne 74
supérieur à 70
supérieur ou égal à 71
virgule 59
OR, opérateur 66
ord(), fonction 21

P

play(), fonction 22
prevFrame(), fonction 22
prevScene(), fonction 23
propriétés
_alpha 33
_currentframe 34
_focusrect 34
_framesloaded 35
_height 35
_highquality 36
_level 36
_name 38
_rotation 38
_scroll 38
_target 39
_visible 40
_width 40
_x 41

_xscale 41
_y 42
_yscale 42
barre de fraction 33
maxscroll 37
scroll 38

R

random(), fonction 24
removeMovieClip(), fonction 24
_rotation, propriété 38

S

scénario racine, identificateur 33
scroll, propriété 38
séparateur de chaîne, opérateur 74
set(), fonction 25
setProperty(), fonction 26
stop(), fonction 26
stopAllSounds(), fonction 27
String(), fonction 27
substring(), fonction 28
supérieur à, opérateur 70
supérieur ou égal à, opérateur 71
switch, instruction 51

T

_target, propriété 39
tellTarget(), fonction 28
toggleHighQuality(), fonction 29
_totalframes, propriété 39
trace(), fonction 29

U

unloadMovie(), fonction 30
unloadMovieNum(), fonction 31

V

variables
_alpha 33
_cap4WayKeyAS 88
_capCompoundSound 83
_capEmail 84
_capLoadData 84
_capMFi 85
_capMIDI 85
_capMMS 86
_capSMAF 87
_capSMS 87
_capStreamSound 88
$version 89
fonctionnalité d'envoi de messages électroniques 84
fonctionnalité de chargement des données 84
navigation avec les touches fléchées 88
numéro de version de Flash Lite 89
variables de messagerie 86, 87
variables de son 83, 85, 87, 88
variables, messagerie
_capMMS 86
_capSMS 87
variables, son
_capCompoundSound 83
_capMFi 85
_capMIDI 85
_capSMAF 87
_capStreamSound 88
_visible, propriété 40

W

while, boucle 47
while, instruction 52
_widthx11, propriété 40

X

_x, propriété 41
xd0 (soustraction), opérateur 79
xd0 xd0 (décrément), opérateur 61
_xscale, propriété 41

Y

_y, propriété 42
_yscale, propriété 42