1. Eléments du langage

Depuis plusieurs années et versions de Flash, il est conseillé de placer les scripts ActionScript (AS) sur les frames.

Commentaires

Les blocs de commentaire s'écrivent de deux façons :

- sur une seule ligne avec le début marqué par // et la fin marquée par la fin de la ligne,
- sur plusieurs lignes avec le début marqué par /* et la fin par */.

Variables

Une variable est un nom d'un élément qui permet de mémoriser une valeur. La création d'une variable s'effectue à l'aide de l'instruction var. Les valeurs numériques incluent trois types de données :

- Number toute valeur numérique,
- int un nombre entier,
- uint un nombre entier qui ne peut être négatif.

Les autres types les plus courants sont : String, Boolean, Object. Les variables du type * sont non typées (undefined).

Une variable de type int peut contenir un nombre oscillant entre -2147483648 et 2147483648. Une variable de type uint peut contenir un nombre entier oscillant entre 0 et 4294967295.

Exemples :

```
var maValeur:Number = 8;
var maValeur:String = "Marie";
var maValeur:Boolean = false;
var age:int = 22.2;
trace(age);                 // affiche : 22 (valeur arrondie à l'entier inférieur)
trace(int.MIN_VALUE);       // affiche : -2147483648
trace(int.MAX_VALUE);       // affiche : 2147483648
trace(uint.MIN_VALUE);      // affiche : 0
trace(uint.MAX_VALUE);      // affiche : 4294967295
```

Opérateurs

Les opérateurs symboliques sont des caractères qui spécifient comment combiner, comparer ou modifier les valeurs d'une expression.

	Arithmétique			Affectation composée arithmétique
+, -	addition, soustraction		+=, -=	
*, /	multiplication, soustraction		*=, /=	
++, --	incrémentation, décrémentation			Affection
%	modulo (reste de la division entière)		=	Affecte la valeur (à droite) à l'élément à gauche
	Logique			Comparaisons
&&, ||	et, ou		==, !=	égalité, inégalité
!	sauf		>, >=	supérieur à, supérieur ou égal à
	Autres		<, <=	inférieur à, inférieur ou égal à
new	constructeur (instancie une occurrence)			Sur les chaînes de caractères
delete	destructeur		+	concaténation
.	accès aux méthodes et aux variables		+=	affectation de concaténation
type	attribution d'un type		"	séparateur de chaîne

Boucles (for, for…in, for each… in, while…, do… while)

Une boucle permet d'exécuter un bloc de code de façon répétée :

- for permet de faire une itération sur une variable pour une plage de valeurs spécifique. Cf. exemple 1.
- for... in permet de faire une itération sur les propriétés d'un objet ou les éléments d'un tableau. Cf. exemple 2.
- for each...in permet de faire une itération sur les éléments d'une collection (balises dans un objet XML ou XMLList, valeurs des propriétés d'un objet ou éléments d'un tableau). Cf. exemples 3 et 4.
- while est semblable à une instruction if qui se répète tant qu'une condition est vérifiée. Cf. exemple 5.
- do...while est une boucle while qui garantit que le bloc de code est exécuté au moins une fois, car la condition est vérifiée une fois que le bloc de code est exécuté. Cf. exemple 6.

Exemple 1 :

```
var i:int;
for(i = 0; i < 8; i++){
      trace(i);
}                        // sortie : 01234567
```

Exemple 2 :

```
var monObjet:Object = {x:50, y:100};
for(var i:String in monObjet){
     trace(i + ": " + monObjet[i]);
}                    // sortie : x: 50    y: 100
```

Exemple 3 : sur les propriétés d'un objet

```
var monObjet:Object = {x:50, y:100};
for each (var num in monObjet){
      trace(num);
}                          // sortie :  50    100
```

Exemple 4 : sur les éléments d'un tableau

```
var monArray:Array = ["Marie", "Paul",
"Jacques"];
for each (var item in monArray){
     trace(item);
}        // sortie :   Marie    Paul    Jacques
```

Exemple 5 :

```
var i:int = 0;
while (i < 8){
     trace(i);
     i++;
}                       // sortie :  01234567
```

Exemple 6 :

```
var i:int = 8;
do{
     trace(i);
     i++;
} while (i < 8);              // sortie     8
```

Conditions (if… else, switch…)

L'instruction conditionnelle if…else permet de tester une condition, puis d'exécuter un bloc de code lorsque cette condition est remplie, d'en exécuter un autre dans le cas contraire. Cf. exemples 7 et 8.

L'instruction switch est utile s'il y a plusieurs chemins d'exécution sur la même expression de condition. Cf. exemple 9.

Exemple 7 : si… sinon…

```
if (maValeur == 0){
     trace("0");
}
else {
      trace("non défini");
}
```

Exemple 8 : pour tester plusieurs conditions

```
if (maValeur == 0){
     trace("0");
}
else {
     trace("non défini");
}
if (maValeur > 0){
     trace("positif");
}
else if (maValeur < 0) {
           trace("négatif");
     }
else {
     trace("nul");
}
```

Exemple 9 :

```
var maValeur:String = "3";
switch(maValeur){
     case "0" : trace("maValeur = 0");  break;
     case "1" : trace("maValeur = 1");  break ;
     case "2" : trace("maValeur = 2");  break;
     case "3" : trace("maValeur = 3");  break;
     default : trace("maValeur = <0 ou >4");   break;
}
```

Evénements, gestionnaires d'événement et fonctions

Les événements sont des faits qui surviennent, comme le mouvement de la souris par exemple, avec lesquels ActionScript peut interagir. La gestion des événements est la technique qui permet de spécifier les actions à exécuter en réponse à des événements particuliers. Un gestionnaire d'événement permet d'associer une fonction à un événement lié à clip ou à une fonction.

Exemple 10 : écouteurs pour capter un événement "souris" et générique

```
// on capte le clic de souris sur le clip monClip_mc et on exécute la fonction fonctionClick()
monClip_mc.addEventListener(MouseEvent.CLICK, fonctionClick);

// on capte le fait que le clip monClip_mc existe,
// si oui on exécute la fonction fonctionEnterframe()
monClip_mc.addEventListener(Event.ENTER_FRAME, fonctionEnterframe);
```

objet "sensible" — méthode pour rajouter un "écouteur" — choix de l'événement — nom de la fonction à associer

Exemple 11 : déclaration des fonctions de callback

```
function fonctionClick(event:MouseEvent):void  {
// code ActionScript…
}

function fonctionEnterframe(event:Event):void  {
// code ActionScript…
}
```

Exemple 12 : capter un événement sur un bouton

```
monBouton_btn.addEventListener(MouseEvent.MOUSE_DOWN, fonctionMouseDown);
function fonctionMouseDown(event:MouseEvent):void    {
// code ActionScript…
          trace("bouton enfoncé");
}
```

Exemple 13 : déclaration d'une fonction avec 2 paramètres (pour concaténer 2 chaînes de caractères)

```
// définition de la fonction
function metEnsemble(c1:String, c2:String):String
{   var total:String = c1+c2;
    return total;
}
// utilisation de la fonction
var ensemble:String = metEnsemble("bouton", " down");
```

Exemple 14 : déclaration d'une fonction avec n paramètres (pour calculer une moyenne)

```
// définition de la fonction
function calculMoyenne(...parametres ):Number
{    var lng:int = parametres.length;
     var total:Number = 0;
     for (var i:Number = 0; i< lng; i++) {
          total += parametres[i];
         }return total / lng;
}

// utilisation de la fonction
var moyenne:Number = calculMoyenne( 50, 48, 78, 20, 90 );
```

Erreurs et gestions des erreurs (avec try catch)

En AS 3, lorsque l'exécution du programme est interrompue de manière anormale, on dit qu'une erreur d'exécution est levée.

Exemple 15 (à ne pas reproduire) :

```
// définition d'une variable de type MovieClip
var monClip_mc:MovieClip;
// pour récupérer le nombre de frames du scénario de ce clip
var nbImages:int = monClip_mc.totalFrames;
trace(nbImages);
// La fenêtre de sortie affiche alors : TypeError: Error #1009:
// Il est impossible d'accéder à la propriété ou à la méthode d'une référence d'objet nul.
```

La valeur par défaut d'un clip créé est "null". Le nombre d'images de son scénario vaut "null". On dit qu'une erreur d'exécution est levée. Afin de gérer cette erreur, il est possible d'utiliser un bloc try catch. Cf. exmple 16.

Exemple 16 :

```
// définition d'une variable de type MovieClip (sa valeur par défaut est "null")
var monClip:MovieClip;
var nbImages:int;
// le bloc try catch permet de gérer l'erreur
try { nbImages = monClip.totalFrames;
    } catch(pErreur:Error) { trace("une erreur d'exécution a été levée !");
    }
```

Propriétés et méthodes (d'un objet)

Une propriété représente l'une des données réunies dans un objet. Cf. exemple 17.
Une méthode est une action qui peut être effectuée par un objet. Cf. exemple 18.

Exemple 17 :

```
monClip_mc.x = 50;       // le clip est placé horizontalement à 50 px
monClip_mc.rotation = 90;  // le clip est orienté à 90 degrés
monClip_mc.rotation += 5;  // le clip est tourné de 5 degrés en plus
                           // par rapport à son orientation actuelle
```

Exemple 18 :

```
// bloque puis débloque
// la timeline du clip
monClip_mc.play();
monClip_mc.stop();
```

Zones de texte dynamiques

Les zones de textes statiques sont des zones de textes qui ne peuvent pas être modifiées au cours de l'animation. A l'inverse, les zones de textes dynamique ou les zones de saisie peuvent être modifiées.

Exemple 19 :

```
monTexte_txt.text="ceci est un texte";
monTexte_txt.width = 300;
monTexte_txt.autoSize = TextFieldAutoSize.LEFT;
monTexte_txt.wordWrap = true;
```

La classe Timer

Avec ActionScript 3, la gestion du temps peut se faire en utilisant la classe Timer(). Cette classe prédéfinie permet de distribuer des événements dès qu'un intervalle de temps est atteint. Pour cela il faudra créer un nouvel objet Timer, et lui indiquer le nombre de millisecondes entre les événements.

Exemple 20 :

```
var monTimer:Timer = new Timer(1000);
monTimer.addEventListener(TimerEvent.TIMER, fonctionTimer);

function fonctionTimer(event:TimerEvent):void
{
       // Code ActionScript...
       trace("bing"); // ici, affichage dans la fenêtre de sortie toutes les 1000 millisecondes
}
monTimer.start();
```

2. Classes des événements

Liste des classes

Classe	Description
ActivityEvent	Flash® Player distribue un objet ActivityEvent chaque fois qu'une caméra ou un microphone signale qu'il est devenu actif ou inactif.
AsyncErrorEvent	Flash® Player distribue un événement AsyncErrorEvent lorsqu'une exception est renvoyée par le code asynchrone natif (LocalConnection, NetConnection, SharedObject ou NetStream).
ContextMenuEvent	Flash® Player distribue des objets ContextMenuEvent lorsqu'un utilisateur génère ou utilise le menu contextuel.
DataEvent	Flash® Player distribue des objets DataEvent lorsqu'il a terminé de charger des données brutes.
ErrorEvent	Flash® Player distribue des objets ErrorEvent lorsqu'une erreur entraîne l'échec d'une opération réseau.
Event	La classe Event est utilisée comme classe de base pour la création des objets Event, transmis aux écouteurs d'événement en tant que paramètres lorsqu'un événement se produit.
EventDispatcher	La classe EventDispatcher implémente l'interface IEventDispatcher et est la classe de base de la classe DisplayObject.
EventPhase	La classe EventPhase fournit des valeurs à la propriété eventPhase de la classe Event.
FocusEvent	Flash® Player distribue des objets FocusEvent lorsque l'utilisateur déplace le focus sur un autre objet dans la liste d'affichage.
FullScreenEvent	Flash® Player distribue un objet FullScreenEvent chaque fois que la scène passe en mode d'affichage plein écran ou quitte ce mode.
HTTPStatusEvent	Flash® Player distribue des objets HTTPStatusEvent lorsqu'une requête réseau renvoie un code d'état HTTP.
IMEEvent	Flash® Player distribue des objets IMEEvent lorsqu'un utilisateur entre du texte à l'aide d'un IME (éditeur de méthode d'entrée).
IOErrorEvent	Flash® Player distribue un objet IOErrorEvent lorsqu'une erreur entraîne l'échec d'une opération d'envoi ou de chargement.
KeyboardEvent	Flash® Player distribue des objets KeyboardEvent en réponse à une saisie utilisateur au clavier.
MouseEvent	Flash® Player distribue des objets MouseEvent dans le flux d'événements à chaque événement de souris.
NetStatusEvent	Flash® Player distribue des objets NetStatusEvent lorsqu'un objet NetConnection, NetStream ou SharedObject signale son état.
ProgressEvent	Flash® Player distribue les objets ProgressEvent lorsqu'une opération de chargement a commencé ou qu'un socket a reçu des données.
SecurityErrorEvent	Flash® Player distribue des objets SecurityErrorEvent pour signaler une erreur de sécurité.
StatusEvent	Flash® Player distribue des objets StatusEvent lorsqu'un périphérique, tel qu'une caméra ou un microphone, ou un objet de type LocalConnection publie son état.
SyncEvent	Flash® Player distribue des événements SyncEvent lorsqu'une occurrence de SharedObject distante a été mise à jour par le serveur.
TextEvent	Flash® Player distribue des objets TextEvent lorsqu'un utilisateur saisit du texte dans un champ de texte ou clique sur un lien hypertexte dans une zone de texte de type HTML.
TimerEvent	Flash® Player distribue des objets TimerEvent chaque fois qu'un objet Timer atteint l'intervalle spécifié par la propriété Timer.delay.

Classes les plus utiles

Event distribue les événements généraux liés à l'animation : ACTIVATE, ADDED, ADDED_TO_STAGE, CANCEL, CHANGE, CLOSE, COMPLETE, CONNECT, DEACTIVATE, DISPLAYING, ENTER_FRAME, FULLSCREEN, ID3, INIT, MOUSE_LEAVE, OPEN, REMOVED, REMOVED_FROM_STAGE, RENDER, RESIZE, SCROLL, SELECT, SOUND_COMPLETE, TAB_CHILDREN_CHANGE, TAB_ENABLED_CHANGE, TAB_INDEX_CHANGE, UNLOAD.
MouseEvent distribue les événements liés à la souris : CLICK, DOUBLE_CLICK, MOUSE_DOWN, MOUSE_MOVE, MOUSE_OUT, MOUSE_OVER, MOUSE_UP, MOUSE_WHEEL, ROLL_OUT, ROLL_OVER
ErrorEvent distribue les événements liés aux erreurs : ERROR
FocusEvent distribue les événements liés aux sélecttions : FOCUS_IN, FOCUS_OUT, KEY_FOCUS_CHANGE, MOUSE_FOCUS_CHANGE
StatusEvent distribue les événements liés aux status : STATUS
TextEvent distribue les événements spécifiques aux zones de texte : LINK, TEXT_INPUT
TimerEvent distribue les événements liés aux timer : TIMER, TIMER_COMPLETE
KeyboardEvent distribue les événements liés au clavier : KEY_DOWN, KEY_UP, KeyLocation

3. Méthodes utiles

```
trace("la valeur de ii est " + ii);     // permet d'afficher dans la fenêtre de sortie

pool_mc.stop()                          // permet de bloquer la tête de lecture
pool_mc.play()                          // permet d'avancer la tête de lecture
pool_mc.gotoAndPlay(10)                 // pour déplacer la tête de lecture du clip pool_mc
                                        // et continuer à partir de la frame 10
pool_mc.gotoAndStop(10)                 // pour déplacer la tête de lecture du clip pool_mc
                                        // et s'arrêter à la frame 10

// pour ouvrir une page Web (avec try catch)
var url:String = "http://www.uha.fr";
var request:URLRequest = new URLRequest(url);
try {
  navigateToURL(request, '_blank'); // the second argument is the target
} catch (e:Error) {
  trace("Error occurred!");
}

// pour ouvrir une page Web en cliquant sur un clip (avec try catch)
monClip_mc.addEventListener(MouseEvent.CLICK, callLink);
function callLink(event:MouseEvent):void {
  var url:String = "http://www.uha.fr";
  var request:URLRequest = new URLRequest(url);
  try {
    navigateToURL(request, '_blank');
  } catch (e:Error) {
    trace("Error occurred!");
  }
}

// pour charger dynamiquement un fichier externe (swf, gif, png, gif...) dans un clip
var request:URLRequest = new URLRequest("monFichier.png");
var monLoader:Loader = new Loader();
monLoader.load(request);
theClip_mc.addChild(monLoader);

// en plus court...
var my_ monLoader:Loader = new Loader();
my_ monLoader.load(new URLRequest("myPhoto.jpg"));
addChild(monLoader);

Mouse.hide();                           // cache le pointeur de la souris
Mouse.show();                           // affiche le pointeur de la souris

photo_mc.mask=masque_mc;                // autorise le masquage
photo_mc.mask=null;                     // annule le masquage

masque_mc.startDrag(true);              // autorise le "dragage"
masque_mc.stopDrag();                   // arrête le "dragage"

ObjetA_mc.hitTestObject(ObjetB_mc);     // détection une collision entre 2 clips
```

4. Principales propriétés des objets d'affichage (clip, bouton, sprite)

Propriété	Unité	max/min	Spécificités
x	pixels	pas de limite	coordonnée horizontale (par rapport au bord gauche de la scène)
y	pixels	pas de limite	coordonnée verticale (par rapport au bord haut de la scène)
alpha	%	entre 0 et 1	transparence d'un objet
rotation	degrés	pas de limite	angle de rotation d'un objet
visible	booléen	true ou false	détermine si un objet est visible ou masqué
height	pixels	pas de limite	hauteur d'un objet
width	pixels	pas de limite	largeur d'un objet
mouseX	pixels	en lecture seule	position de la souris par rapport au côté gauche de la scène
mouseY	pixels	en lecture seule	position de la souris par rapport au côté haut de la scène
scaleX	%	pas de limite	échelle horizontale d'un objet par rapport à sa taille d'origine (au moment où il a été placé dans la scène la première fois)
scaleY	%	pas de limite	échelle verticale d'un objet par rapport à sa taille d'origine (au moment où il a été placé dans la scène la première fois)
name	chaîne de caractères	Pas de caract. spéciaux, pas de majuscule au début	permet de définir ou de lire le nom d'occurrence d'un objet
stage	n/a	en lecture seule	tous les objets d'affichage possèdent la propriété commune stage
parent	n/a	en lecture seule	fait référence à l'objet d'affichage qui contient l'objet en question
root	n/a	en lecture seule	fait référence à l'objet d'affichage racine
currentFrame	entiers >0	en lecture seule	numéro de la frame courante (celui où se trouve la tête de lecture)
totalFrames	entiers >0	en lecture seule	nombre total de frames d'un clip

5. Liste des suffixes préconisés pour les types les plus courants

L'utilisation des suffixes permet de bénéficier des conseils de code automatiques (avec CTRL + espace) :

Type d'objet	Suffixe de variable	Descriptif
Array	_array	tableau
Button	_btn	bouton
Camera	_cam	objet pour capturer de la vidéo
Color	_color	objet pour gérer la couleur
ContextMenu	_cm	permet de contrôler à l'exécution les éléments du menu contextuel de Flash Player
ContextMenuItem	_cmi	permet de créer des éléments de menus personnalisés afin qu'ils s'affichent dans le menu contextuel de Flash Player
Date	_date	objet pour gérer le format des dates
Error	_err	objet contenant des informations sur une erreur qui s'est produite dans un script
LoadVars	_lv	objet pour contrôler le chargement des données externes
LocalConnection	_lc	pour développer des fichiers SWF qui peuvent échanger des instructions entre eux
Microphone	_mic	pour capturer des données audio avec un micro

MovieClip	_mc	clip
MovieClipLoader	_mcl	permet d'implémenter des rappels d'écouteur qui fournissent des informations d'état lors du chargement des fichiers SWF, JPEG, GIF et PNG dans les clips
PrintJob	_pj	permet de créer un contenu et de l'imprimer
NetConnection	_nc	permet de lire des fichiers FLV en flux continu à partir d'un lecteur local ou d'une adresse HTTP
NetStream	_ns	fournit des méthodes et des propriétés permettant de lire des fichiers Flash Video (FLV)
SharedObject	_so	permet de lire et stocker des quantités limitées de données sur l'ordinateur d'un utilisateur
Sound	_sound	objet pour gérer des données audio
String	_str	chaîne de caractères
TextField	_txt	champs de texte
TextFormat	_fmt	pour gérer les formats des textes
Video	_video	permet d'afficher un contenu vidéo Internet en direct sans l'intégrer dans le fichier SWF
XML	_xml	objet pour gérer les flux et fichiers XML
XMLNode	_xmlnode	pour gérer des nœuds XML
XMLSocket	_xmlsocket	pour échanger des données entre objet XML

6. Liste des instructions, mots clés et directives

Espace de nom	
AS3	Définit les méthodes et les propriétés des classes ActionScript qui sont des propriétés fixes et non des propriétés de prototype.
flash_proxy	Définit les méthodes de la classe Proxy.
object_proxy	Définit les méthodes de la classe ObjectProxy.
Directive	
default xml namespace	La directive `default xml namespace` définit l'espace de nom par défaut à utiliser pour les objets XML.
import	Rend les classes et les packages définis en externe disponibles pour votre code.
include	Inclut le contenu du fichier spécifié, comme si les commandes du fichier faisaient partie du script d'appel.
use namespace	Suite à cette opération, les espaces de nom spécifiés sont ajoutés à l'ensemble d'espaces de nom ouverts.
instruction	
break	Apparaît dans une boucle (`for`, `for..in`, `for each..in`, `do..while` ou `while`) ou dans un bloc d'instructions associées à un cas particulier dans une instruction `switch`.
case	Définit un hyperlien cible pour l'instruction `switch`.
continue	Ignore toutes les instructions restantes dans la boucle imbriquée de plus bas niveau et passe à l'itération suivante, comme si le contrôle avait été transmis à la fin de la boucle normalement.
default	Définit le cas par défaut d'une instruction `switch`.
do..while	Semblable à une boucle `while`, à la différence que les instructions sont exécutées une fois avant l'évaluation initiale de la condition.
else	Spécifie les instructions à exécuter si la condition de l'instruction `if` renvoie `false`.
for	Evalue l'expression `init` (initialiser) une fois, puis amorce une séquence de bouclage.
for..in	Répète en boucle les propriétés dynamiques d'un objet ou d'éléments de tableau, puis exécute l'instruction `statement` pour chaque propriété ou élément.
for each..in	Procède à une itération sur les éléments d'une collection et exécute `statement` pour chaque élément.
if	Evalue une condition pour déterminer la prochaine instruction à exécuter.
label	Associe une instruction à un identificateur qui peut être référencé par `break` ou `continue`.
return	Transfère immédiatement l'exécution à la fonction appelante.
super	Invoque la version super-classe ou parent d'une méthode ou d'un constructeur.
switch	Transfère le contrôle à une instruction en fonction de la valeur d'une expression.

throw	Génère ou *renvoie* une erreur qui peut être traitée ou *interceptée* par un bloc de code `catch`.
try..catch..finally	Entoure un bloc de code dans lequel une erreur peut se produire et être traitée.
while	Evalue une condition. Si elle renvoie `true`, exécute une ou plusieurs instructions avant de remonter la boucle et de réévaluer la condition.
with	Etablit un objet à utiliser par défaut pour l'exécution d'une ou plusieurs instructions, ce qui permet de réduire le nombre de lignes de code à écrire.
mot-clé d'attribut	
dynamic	Spécifie que les occurrences d'une classe peuvent posséder des propriétés dynamiques qui sont ajoutées lors de l'exécution.
final	Permet de spécifier qu'une méthode ne peut pas être remplacée ou qu'une classe ne peut pas être étendue.
internal	Permet de spécifier si une classe, une variable, une constante ou une fonction est disponible pour tous les appels au sein du même package.
native	Permet de spécifier si une fonction ou méthode doit être implémentée par Flash Player en code natif.
override	Spécifie une méthode qui remplace une méthode héritée.
private	Spécifie qu'une variable, une constante, méthode ou espace de nom est disponible uniquement pour la classe qui la définit.
protected	Spécifie qu'une variable, une constante, méthode ou espace de nom est disponible uniquement pour la classe qui la définit et pour toutes les sous-classes de cette classe.
public	Permet de spécifier si une classe, une variable, une constante ou une méthode est disponible pour tous les appels.
static	Permet de spécifier qu'une variable, constante ou méthode appartient à la classe, et non pas aux occurrences de la classe.
mot clé de définition	
... (rest) parameter	Permet de spécifier si une fonction peut accepter un nombre illimité d'arguments séparés par des virgules.
class	Définit une classe, ce qui permet de créer des occurrences d'objets qui partagent les méthodes et les propriétés que vous définissez.
const	Spécifie une constante, variable qui ne peut recevoir de valeur qu'une seule fois.
extends	Définit une classe qui est une sous-classe d'une autre classe.
function	Comprend un ensemble d'instructions que vous définissez pour effectuer une certaine tâche.
get	Définit une méthode de lecture qui peut être lue comme une propriété.
implements	Spécifie qu'une classe implémente une ou plusieurs interfaces.
interface	Définit une interface.
namespace	Permet de contrôler la visibilité des définitions.
package	Permet de répartir votre code dans des groupes de petite taille qui peuvent être importés par d'autres scripts.
set	Définit une méthode de définition, méthode qui s'affiche dans l'interface publique en tant que propriété.
var	Spécifie une variable.
mot clé d'expression primaire	
false	Valeur booléenne false.
null	Valeur spéciale qui peut être affectée à des variables ou renvoyée par une fonction en l'absence de données.
this	Référence à l'objet contenant une méthode.
true	Valeur booléenne true.

7. Webographie

http://wiki.mediabox.fr/tutoriaux/flash
http://www.bases-as3.fr/index.php/plan-site
http://livedocs.adobe.com/flash/9.0/ActionScriptLangRefV3/
http://www.adobe.com/devnet/actionscript/articles/actionscript3_overview.html
http://www.flashandmath.com/