Apprendre le VBscript

Bref préambule

Après le Html (www.ccim.be/ccim328/html/index.htm) et le Javascript (www.ccim.be/ccim328/js/index.htm), il était logique de s'intéresser à cet autre langage de script qu'est le VBscript. Plutôt qu'un long tutorial, ce chapitre consacré au VBscript est surtout à considérer comme un tour d'horizon des possibilités de Vbscript. Bien que la tentation fut grande, j'ai évité de faire de ce qui suit, un match Javascript contre VBscript ou même Netscape contre Microsoft. D'autres s'en chargent...

1. Le VBscript

VBscript est un langage de script
qui incorporé aux balises Html
permet d'augmenter la présentation
et l'interactivité des pages Web.

- VBscript est donc une extension du code Html des pages Web. Les scripts sont en quelque sorte des ajoutes (ou ajouts) au code Html.
- Ces scripts vont être interprétés et exécutés par le navigateur (donc côté client) sans devoir faire appel aux ressources du serveur.
- VBscript (pour Microsoft Visual Basic Scripting Edition) a été développé par Microsoft qui a repris la logique et la syntaxe de son Visual Basic (d'où le VB). Bien que quelques fonctions lui soient propres, on peut considérer VBscript comme un sous-ensemble de Visual Basic.
- Pour l'instant, les seuls browsers qui reconnaissent le VBscript sont les navigateurs de la firme Microsoft, soit Internet Explorer 3 et Internet Explorer 4.
- VBscript ouvre la porte aux différents produits de Microsoft dédiés au Web et principalement aux ActiveX. Ce qui peut être particulièrement adapté dans le cas d'un réseau Intranet.
- Dans l'inévitable compétition avec cet autre langage de script qu'est Javascript, l'avenir de VBscript dépendra en grande partie des autres navigateurs qui adopteront les routines VBscript. Vous pensez du côté de Netscape ? Moi aussi !

2. VBscript ou Javascript

Pour le moins qu'on puisse dire, VBscript et Javascript sont très semblables quant à leur philosophie et leur mode de fonctionnement. Heureusement quelques différences, et non des moindres, en font deux outils totalement différents.

--- VBscript et Javascript ---

Code intégré aux balises Html
Code directement interprété par le browser à l'exécution
Permet d'accéder aux objets de la page et du browser
Confidentialité des scripts nulle car le code source est visible
Codes de programmation simples mais pour des applications limitées (sauf appel aux ActiveX pour VBscript)

VBscript	Javascript
Code d'inspiration Visual Basic	Code d'inspiration C et C++
Explorer 3 et 4 seulement !	Famille Netscape et Explorer
Sans appel aux ActiveX, Vbscript semble moins élaboré que Javascript	Apparaît plus autonome que VBscript

Même sans comprendre (à ce stade de cette introduction) les scripts, il peut être intéressant de comparer deux scripts identiques en VBscript et Javascript.

VBscript

```
<HTML>
<HEAD>
<SCRIPT language="VBscript">
<!--
Sub ok_OnClick
MsgBox "Le bouton est cliqué."
End Sub
-->
</SCRIPT>
</HEAD>
<BODY>
<FORM name="commande">
<INPUT name="ok" type="button"
value="Cliquez ici">
</FORM>
</BODY>
</HTML>
```

Javascript

```
<HTML>
<HEAD>
<SCRIPT language="Javascript">
<!--
function clickbut(){
alert("Le bouton est cliqué.")
}
//-->
</SCRIPT>
</HEAD>
<BODY>
<FORM name="commande">
<INPUT name="ok" type="button"
value="Cliquez ici" onClick="clickbut()">
</FORM>
</BODY>
</HTML>
```

VBscript et Javascript sont donc assez semblables. Ils diffèrent surtout par le langage dont ils sont dérivés, soit Visual Basic pour le VBscript et le langage C et C++ pour le Javascript.

Pour terminer, je voudrais ajouter qui si VBscript ressemble à du Javascript, il n'est en rien comparable à du Java. Cette confusion entre Javascript et Java est assez classique. En deux mots, Java est un langage du programmation à part entière, il forme un module (applet) distinct de la page Html et son code source est compilé avant son exécution.

3. VBscript et ActiveX

- Comme pour l'utilisateur le résultat final est identique, on serait tenté de dire que ActiveX est à VBscript ce que les applets Java sont à Javascript. Mais ceci est une approche trop simpliste car le concept d'ActiveX est une technologie sensiblement différente.

- "ActiveX est une plate-forme d'intégration ouverte qui fournit aux développeurs, aux utilisateurs et aux réalisateurs de Web le moyen le plus rapide et le plus facile de créer de nouvelles applications et de nouveaux contenus pour Internet et pour les Intranets."

  Microsoft présente donc son produit ActiveX comme un outil d'intégration de technologies permettant de créer des pages interactives sur le Web. L'idée de départ à été de profiter de l'expérience de Microsoft en matière de partage entre diverses applications (la technologie des Dll, Ole et autres VBX) pour la transposer sur le Web en recréant un tout un environnement dédié (au départ) à Windows.

- Les contrôles ActiveX peuvent être écrits dans différents langages de programmation comme le C, le C++, Pascal (Delphi), Visual Basic 5 ou Microsoft Visual J++. Ainsi, au contraire d'un langage de script comme VBscript, on évolue ici dans le monde de la programmation ce qui ne rend pas les choses particulièrement simples.

- L'appel à ses composants ActiveX est inséré dans une page Web par une combinaison de Html et de VBscript et spécialement par la balise <OBJECT> ... </OBJET> qui comprend de nombreux attributs. Nous y reviendrons plus loin dans cette introduction à VBscript sur l'incorporation des contrôles ActiveX dans vos pages Html.

- Lorsque Internet Explorer rencontre un appel à un contrôle ActiveX, celui-ci, situé initialement sur un serveur, explore votre machine [je n'aime pas trop ...] pour voir si le contrôle n'est pas déjà présent. Si le contrôle est absent ou d'une version plus ancienne, tous les fichiers nécessaires seront téléchargés et copiés sur votre disque dur [je n'aime pas trop ...]. Et à la différence des applets Java, les composants ActiveX resteront sur le disque dur de l'utilisateur et feront partie intégrante de son environnement Windows [je n'aime pas trop ...].

- Vous aurez deviné que ce fonctionnement peut poser des problèmes de sécurité. Microsoft utilise un système de source "sûre" ou "d'autorité certifiée" (Certificate Authority). Chaque composant est en quelque manière "signé" de manière à garantir son origine. De toute façon, Internet Explorer vous demande si vous acceptez les composants provenant de cette source.

- Dans la pratique, l'utilisateur moyen ou tout simplement prudent d'Internet rechignera à laisser librement installer des composants sur sa machine. En reprenant mot à mot un extrait d'une documentation de Microsoft "Avec Internet, en revanche, les utilisateurs reçoivent généralement votre composant comme un effet de l'exploration et ne feront fonctionner aucun programme d'installation". Tout ceci rejoint notre opinion que le tandem VBscript et ActiveX est, pour l'instant du moins, plutôt une solution pour Intranet que pour Internet.

- Terminons en signalant que si VBscript n'est pas accepté par Netscape, les contrôles ActiveX par contre peuvent être lus par Netscape par l'entremise d'un plug-in fourni par Microsoft.

4. Les outils pour VBscript

Pour apprendre et utiliser le VBscript, il vous faut :

1. un browser qui reconnaît le Vbscript
2. une solide connaissance du Html
3. un simple éditeur de texte

4.1 Un browser compatible VBscript

Le choix est limité aux seuls browsers de la firme Microsoft. Ce sera soit Microsoft Internet Explorer 3, soit Microsoft Internet Explorer 4 et aucun autre !

Pour les accros des versions :

- Microsoft Explorer 3.0 utilise VBscript Version 1.0
- Microsoft Explorer 4.0 utilise VBscript Version 3.0

4.2 Une solide connaissance du Html

VBscript ne remplace pas le langage Html. Au contraire, comme VBscript vient s'ajouter aux codes Html, une connaissance approfondie des balises et tags Html est souhaitable sinon indispensable. Les utilisateurs assidus des éditeurs Html "whsiwyg" risquent de devoir retourner à leurs chères études.
Je ne peux que vous recommander un tutorial du même auteur : "Apprendre le langage Html" à l'adresse : www.ccim.be/ccim328/html/index.htm

4.3 Un simple éditeur de texte

Une page Html n'est que du texte. Le code VBscript n'est lui aussi que du texte. Quoi de plus simple que d'utiliser un éditeur de ... texte. Le bloc-notes de Windows, fera parfaitement l'affaire (vous resterez ainsi dans la famille Microsoft).

Un éditeur Html de la première génération (un bon vieil éditeur qui n'est pas whsiwyg et qui fait apparaître les balises) fait également bien l'affaire.

5. Un peu de théorie objet

Les objets et leur hiérarchie

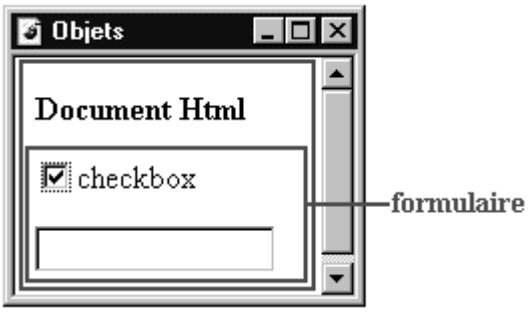
En bon internaute, vous voyez sur votre écran une page Web. VBscript va diviser cette page en objets et surtout va vous permettre d'accéder à ces objets, d'en retirer des informations et de les manipuler.

Voyons d'abord une illustration des différents objets qu'une page peut contenir.

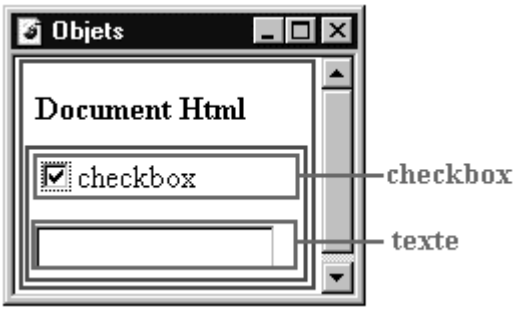
Vous avez chargé la page suivante :

Dans votre browser, il y a une page Web qui n'est rien d'autre qu'un document Html. C'est l'objet document.

Dans ce document, on trouve un formulaire au sens Html du terme. C'est l'objet formulaire. Autrement dit (et c'est là que l'on voit apparaître la notion de la hiérarchie des objets VBscript), l'objet document contient un objet formulaire.



Dans ce document, on trouve deux objets. Des boutons checkbox et une zone de texte. Ce sont respectivement l'objet checkbox et l'objet texte. Autrement dit l'objet document contient l'objet formulaire qui contient à son tour l'objet radio et l'objet document contient aussi l'objet formulaire qui contient à son tour l'objet texte.



La hiérarchie des objets de cet exemple est donc :

document	formulaire	Checkbox
document	formulaire	Texte

Pour accéder à un objet (vous l'avez peut-être déjà deviné), il faudra donner le chemin complet de l'objet (ou la généalogie complète de l'objet) en allant du contenant le plus extérieur à l'objet à l'objet référencé.

Soit par exemple pour le bouton checkbox : document.formulaire.checkbox
ou pour la zone de texte document.formulaire.texte.

6. Le VBscript minimum

6.1 La balise <SCRIPT>

Le langage Html utilise des balises pour "dire" au navigateur d'afficher du texte en gras, des images, des liens, etc. Dans la logique du Html, il faut donc signaler au browser par un tag que ce qui suit est un script et que c'est du VBscript (et non du Javascript).
C'est la balise <SCRIPT language="VBscript">.
De même, il faudra informer le browser de la fin du script.
C'est la balise </SCRIPT>.

6.2 Les commentaires

Il vous sera peut-être utile d'inclure de commentaires personnels dans vos codes VBscript.
VBscript utilise les conventions de Visual Basic, soit :
' commentaires
ou
Rem commentaires
Tout ce qui est écrit après l'apostrophe ou le mot clé Rem sera ignoré.

6.3 Masquer le script pour les autres browsers

Les browsers qui ignorent la balise <script>, vont afficher le code VBscript comme du texte. Pour éviter l'affichage de ce charabia, on utilise les balises de commentaire du Html soit <!-- et -->.

Votre premier VBscript ressemblera à ceci :

```
<SCRIPT language="VBscript">
<!--
...
code VBscript
...
-->
</SCRIPT>
```

6.4 Où inclure la balise de script

Le browser traite votre page Html, y compris vos ajoutes en VBscript de haut en bas. Toute instruction ne pourra être exécutée que si le browser possède à ce moment précis tous les éléments nécessaire à son exécution.

- Au moment de l'exécution. La balise de script apparaît alors à l'intérieur des tags <BODY> </BODY>.
- Pour s'assurer que le programme VBscript est bien chargé et prêt à fonctionner à toute intervention de votre visiteur, on prendra l'habitude de déclarer systématiquement (lorsque cela sera possible) un maximum d'éléments dans les balises d'en-tête soit entre <HEAD> et </HEAD> et avant la balise <BODY>.
  Ce sera le cas par exemple pour les procédures ou fonctions.
- Dans certains cas, la balise de script ne devra même pas être utilisée. Ce sera le cas pour les ajouts de script dans les contrôles de formulaire. Nous y reviendrons plus tard.

Rien n'interdit d'inclure plusieurs scripts dans la même page Html.

6.5 Attention

VBscript est case sensitive. Ainsi il faudra écrire write et non Write. Pour l'écriture des instructions VBscript, on utilisera l'alphabet ASCII classique (à 128 caractères) comme en Html. Les caractères accentués comme é ou à ne peuvent être employés que dans les chaînes de caractères c.-à-d. dans votre texte de notre exemple.

6.6 Votre première page Html avec du VBscript

```
<HTML>
<HEAD>
</HEAD>
<BODY>
... Html normal ...
<SCRIPT language="VBscript">
<!--
MsgBox "Mon premier VBscript!"
-->
</SCRIPT>
... Suite en Html ...
</BODY>
</HTML>
```

7. Les boites de message

7.1 La boite de message - MsgBox -

Petite boite d'un usage simple mais au combien utile. Cette boite affiche votre message et reste à l'écran aussi longtemps que le lecteur ne clique sur le bouton OK pour confirmer la bonne réception du message.

La syntaxe est de plus simple :

```
MsgBox "votre texte"
```

Voici un exemple :

```
MsgBox "Message reçu ?"
```

Cette boite de message sera souvent utilisée pour corriger les scripts présentant des erreurs. Ainsi, MsgBox variable, affichera le contenu de la variable à cet endroit du script ou MsgBox "jusqu'ici", vous permettra de voir jusqu'où se déroule le script erroné. Nous y reviendrons dans le chapitre intitulé "Les messages d'erreurs".

Vous pouvez aussi afficher du texte sur plusieurs lignes. Pour se faire, il suffit d'employer le mot-clé vbCrLf ou Chr (13) ou Chr (13) & Chr (10).
Ainsi MsgBox "Ligne 1" & vbCrLf & "Ligne 2" & Chr(13) & "Ligne 3", affichera dans la boite de message :

Ligne 1
Ligne 2
Ligne 3

7.2 La boite de message - Alert -

Assurément un emprunt au Javascript, la fonction Alert permet également d'afficher un boite de message. Tout comme MsgBox, Alert ouvre une petite fenêtre avec votre texte et un bouton OK. Aussi longtemps que le lecteur n'aura pas confirmé par un clic sur ce bouton, le processus sera bloqué.

La syntaxe est :

```
Alert "votre texte"
```

Un exemple ?

```
Alert "Message reçu ?
```

Votre oeil perspicace aura remarqué que la fenêtre s'intitule "Microsoft Internet Explorer" ou lieu de Visual Basic. A mon sens, plus joli.

7.3 La boite de saisie - InputBox -

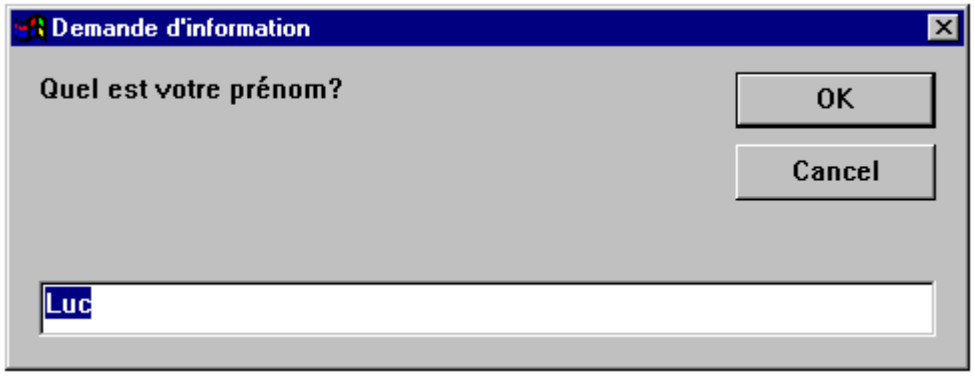
Avec la boite de saisie, vous pouvez accroître l'interactivité avec votre lecteur et obtenir de sa part d'autres réponses que oui, non, etc.

La syntaxe est simple :

```
InputBox "votre texte" , "le titre de la boite" , "la valeur par défaut"
```

Voici un exemple :

```
InputBox "Quel est votre prénom?", "Demande d'information", "Luc"
```



La boite de saisie ne comporte pas d'icônes et les seuls boutons sont OK et Cancel.
La valeur ainsi entrée par le lecteur pourra être récupérée dans une variable qui sera traitée dans d'autres lignes de code VBscript.

7.4 La boite de message - MsgBox - évoluée

On peut utiliser la boite MsgBox pour une plus grande interactivité qu'un simple (stupide?) bouton de confirmation. C'est le premier attribut (attribut-bouton), celui du choix des boutons.

Constante	Bouton(s) affiché(s)
vbOk	OK seulement (par défaut)
vbOkCancel	OK et Cancel
vbAbortRetryIgnore	Abort, Retry et Ignore
vbYesNoCancel	Yes, No et Cancel
vbYesNo	Yes et No
vbRetryCancel	Retry et Cancel

On peut modifier la petite icône qui accompagne la boite de message. C'est l'attribut-icône.

Constante	Icône affichée
vbCritical	un X (pour erreur fatale)
vbQuestion	un point d'interrogation (pour une question)
vbExclamation	un point d'exclamation (pour une remarque)
vbInformation	un I (pour une information)

Et si la mention "Visual Basic" de la fenêtre par défaut ne vous enchante guère, le titre de cette fenêtre peut aussi être modifié.

La syntaxe bien entendu, évolue :

MsgBox "votre texte" , attribut-bouton + attribut-icône, "titre de la fenêtre"

MsgBox renvoie une valeur de retour qui indique que le lecteur a cliqué sur tel ou tel bouton. Selon l'évaluation de ce code retour (voir Chapitre "Les conditions") telle ou telle action pourra être programmée. Ces codes sont 1 pour le bouton OK, 2 pour le bouton Cancel, 3 pour le bouton Abort, 4 pour le bouton Retry, 5 pour le bouton Ignore, 6 pour le bouton Yes et enfin 7 pour le bouton No.

On mettra ce code retour dans une variable (ici code). L'écriture prendra la forme :
Dim code
code = MsgBox("texte" , attribut-bouton)

8. Afficher du texte

8.1 Méthode de l'objet document

Rappelez-vous... Nous avions montré que ce qui apparaît sur votre écran, peut être "découpé" en objets et que VBscript allait vous donner la possibilité d'accéder à ces objets (Un peu de théorie objet).

La page Html qui s'affiche dans la fenêtre du browser est un objet de type document. A chaque objet VBscript, le concepteur du langage a prévu un ensemble de méthodes (ou fonctions dédiées à cet objet) qui lui sont propres. A l'objet document, VBscript a dédié la méthode "écrire dans le document", c'est la méthode write.

Pour appeler la méthode write du document, on notera

```
document.write
```

8.2 La méthode write

La syntaxe est assez simple soit

```
document.write "votre texte"
```

On peut aussi écrire une variable, soit la variable resultat,

```
document.write resultat
```

Pour associer du texte (chaînes de caractères) et des variables, on utilise l'instruction

```
document.write "Le résultat est " & resultat
```

On peut utiliser les balises Html pour agrémenter ce texte

```
document.write "<B>Le résultat est</B>" & resultat ou
document.write "<B>" & "Le résultat est "& "</B>" & resultat
```

où & est un opérateur de concaténation qui relie les "wagons" de ce que vous voulez écrire. Nous y reviendrons au chapitre consacré aux opérateurs.

8.3 Exemple (classique !)

On va écrire du texte en Html et en VBscript.

```
<HTML>
<BODY>
<H1>Ceci est du Html</H1>
<SCRIPT language="VBscript">
<!--
document.write "<H1>Et ceci du VBscript</H1>"
-->
</SCRIPT>
</BODY>
</HTML>
```

Ce qui donnera comme résultat :

Ceci est du Html
Et ceci du VBscript

9. Les variables

9.1 Les variables en VBscript

Les variables contiennent des données qui peuvent être modifiées lors de l'exécution d'un programme. On y fait référence par le nom de cette variable.

Les noms de variables :

- ne doivent pas dépasser 255 caractères.
- doivent commencer par une lettre (caractère alphabétique).
- ne peuvent contenir une virgule, un point ou un espace.
- ne peuvent reprendre des mots clés de VBscript.
- doivent être uniques à l'intérieur de leur portée (voir variables globales et locales).

Ajoutons pour nous les francophones, qu'il faut employer l'alphabet ASCII donc les lettres sans accents.

Pour rappel VBscript est sensible à la case. Attention donc aux majuscules et minuscules!

9.2 La déclaration de variable

Les variables peuvent se déclarer de deux façons :

- soit de façon explicite.
  On dit à VBscript que ceci est une variable.
  La commande qui permet de déclarer une variable est le mot clé Dim suivi du nom de la variable (et ce généralement en début de script). Par exemple :
  Dim Numero
  Dim x, y, z

- soit de façon implicite. On écrit directement dans le code VBscript, le nom de la variable suivi de la valeur que l'on lui attribue et VBscript s'en accommode. Par exemple :
  Numero = 1
  Prenom = "Luc"

Si ce mélange possible de variables implicites et explicites vous ennuie, vous pouvez exiger la méthode de déclaration explicite et empêcher les déclarations implicites. Cela se réalise par la commande :

```
Option Explicit
```

Cette commande se place dans la première ligne de code de votre VBscript :

```
<SCRIPT language="VBscript>
<!--
Option Explicit
... la suite du code ...
-->
</SCRIPT>
```

9.3 Le types de données sous VBscript

VBscript utilise un seul type de données nommée Variant (rappel du Visual Basic). Ce type Variant est véritablement un fourre-tout de différents types d'information. En voici quelques-uns :

Type	Description
Des nombres	Tout nombre entier ou avec virgule tel que 22 ou 3.1416
Des chaînes de caractères	Toute suite de caractères alphanumérique comprise entre guillemets telle que "suite de caractères". On emploiera aussi le terme "strings".

Des Booléens	Contient True (vrai) ou False (faux).
Empty	La variable n'a pas encore été initialisée. Sa valeur est égale à 0 pour les variables numériques et " " pour les strings.
Null	Contient (intentionnellement) des données incorrectes.
Error	Contient un numéro d'erreur. Utile pour corriger un script (voir chapitre les messages d'erreur).

9.4 Variables locales et variables globales

Les variables déclarées dans les procédures (voir plus loin) ont une portée dite locale c-à-d qu'elle ne sera valable que dans le cadre de cette seule procédure.

Une variable est dite globale lorsqu'elle pourra être partagée partout dans le code du script. Pour qu'une variable soit globale, elle doit être déclarée en dehors de toutes procédures. Pour ce faire, on les déclare tout au début du script.

Ainsi, la variable compteur définie comme suit sera globale :

```
<SCRIPT language="VBscript">
<!--
Dim compteur
Sub ....
compteur = 0
End Sub
Sub ...
compteur = compteur + 1
End Sub
-->
</SCRIPT>
```

10. Les opérateurs

Après les variables, abordons le chapitre, toujours follement passionnant (sic), des opérateurs.

10.1 Les opérateurs arithmétiques

Dans tous les exemples, y est égal à 11.

0pérateur	Signification	Exemple	Résultat
+	Addition	x = y + 5	x = 16
-	Soustraction	x = y - 5	x = 6
*	Multiplication	x = y * 5	x = 55
/	Division à virgule flottante	x = y / 5	x = 2.2
\	Division sans décimale	x = y \ 5	x = 2
^	Exposant	x = y ^2	x = 121
Mod	Modulo	x = y Mod 5	x = 1
-	Négation	x = -y	x = -11

Il faut noter que la division sans décimale renvoie la partie entière de la division et ne fonctionne en aucun cas comme un arrondi du nombre.

10.2 Opérateurs de comparaison

Ces opérateurs de comparaison seront surtout utilisés dans les tests de conditionnels. Nous y reviendrons dans le chapitre intitulé "Les conditions".

0pérateur	Signification	Exemple	Résultat
=	Egalité	a = b	a est égal à b
<>	Inégalité	a <> b	a différent à b
>	Plus grand	a > b	a supérieur à b
<	Plus petit	a < b	a inférieur à b
>=	Plus grand ou égal	a >= b	a plus grand ou égal à b
<=	Plus petit ou égal	a <= b	a inférieur à b
Is	Equivalence d'objets	objet.ref1 Is objet.ref2	

En VBscript, il n'y a pas de signe différent pour = valeur d'attribution et = comparaison. C'est le sens du script qui l'indique. Ce qui peut poser certains problèmes.
En Javascript, on a = pour la valeur d'attribution et == comparaison.

10.3 Opérateurs logiques

0pérateur	Signification	Exemple
And	Conjonction (et)	condition1 And condition2
Or	Disjonction (ou)	condition1 Or condition2
Not	Négation	Not expression
Xor	Exclusion	condition1 Xor condition2
Eqv	Equivalence logique	condition1 Eqv condition2
Imp	Implication	condition1 Imp condition2

10.4 Opérateur de concaténation

0pérateur	Signification	Exemple	Résultat
&	Concaténation de strings	"nom" & " " & "prénom"	nom prénom
+	Concaténation de strings"	"nom" + " " + "prénom"	nom prénom

Préférez cependant & à + car ce dernier est fait pour les valeurs numériques et l'interpréteur être troublé par la concaténation de nombres et de strings. La concaténation avec l'opérateur +, ne fournit pas toujours un résultat garanti.

11. Les procédures

11.1 Les procédures (généralités)

Une procédure (ou fonction) est un groupe de ligne(s) de code de programmation destiné à exécuter une tâche bien spécifique et que l'on pourra, si besoin est, utiliser à plusieurs reprises. De plus, l'usage des procédures améliorera grandement la lisibilité et la maintenance de votre script.

En VBscript, il existe trois types de fonctions ou de procédures :

- les fonctions propres à Javascript. On les appelle des "méthodes". Elles sont associées à un objet bien particulier comme c'était le cas de la méthode MsgBox avec l'objet document.
- les sous-programmes (Subroutine) écrits par vous-même pour les besoins de votre script et qui ne retournent pas de valeur.
- les fonctions (Function) proprement dites qui sont aussi écrites par vous-même mais qui peuvent retourner une valeur.

11.2 Déclaration d'un sous-programme

Pour déclarer ou définir un sous-programme (Subroutine), on utilise le mot (réservé) Sub. La syntaxe d'une déclaration de sous-routine est la suivante :

```
Sub nom-du-sous-programme(arguments)
... code des instructions ...
End Sub
```

Le nom du sous-programme suit les mêmes règles que celles qui régissent le nom de variables (nombre de caractères 255, commencer par une lettre, peuvent inclure des chiffres...). Pour rappel, VBscript est sensible à la case. Ainsi Calcul() ne sera pas égal à calcul(). En outre, Tous les noms des fonctions dans un script doivent être uniques.

La mention des arguments est facultative mais dans ce cas les parenthèses doivent rester. C'est d'ailleurs grâce à ces parenthèses que l'interpréteur VBscript distingue les variables des fonctions. Nous reviendrons plus en détail sur les arguments et autres paramètres.

Lorsqu'un sous-programme a été ouvert par un Sub, il doit impérativement, sous peine de message d'erreur, être refermé par un End Sub. Prenez la bonne habitude de fermer directement vos sous-routines et d'écrire votre code entre elles.

Le fait de définir une fonction n'entraîne pas l'exécution des commandes qui la composent. Ce n'est que lors de l'appel de la fonction que le code de programme est exécuté.

11.3 L'appel d'un sous-programme

L'appel d'un sous-programme se fait le plus simplement du monde par le nom du sous-programme (sans les parenthèses même s'il y a des arguments).

Soit par exemple :
nom-du-sous-programme
nom-du-sous-programme argument1, argument2

Vous pouvez aussi (mais ce n'est pas obligatoire) utiliser l'instruction call
Call nom-du_sous-programme
Call nom-du-sous-programme(argument1, argument2)
Si le sous-programme a des arguments, il faut ici mettre des parenthèses.

Il faudra veuillez en toute logique (car l'interpréteur lit votre script de haut vers le bas) que votre fonction soit bien définie avant d'être appelée.

11.4 Déclaration d'une fonction

Pour rappel, la fonction retourne une valeur. Pour le reste, les règles sont assez proche de celles vues pour les sous-programmes.

Pour déclarer ou définir une fonction, on utilise le mot clé Function . La syntaxe d'une déclaration de fonction est la suivante :

```
Function nom-de-la-fonction(arguments)
... code des instructions ...
End Function
```

Ce qui été dit pour les sous-programmes concernant le nom, l'usage des parenthèses, L'emploi de End Function reste valable pour les fonctions.

Le fait de définir une fonction n'entraîne pas l'exécution des commandes qui la composent. Ce n'est que lors de l'appel de la fonction que le code de programme est exécuté.

11.5 Retour de la valeur dans une fonction

Pour retourner une valeur, la fonction utilise un petit artifice. On affectera la valeur à une variable dans une des lignes de code de la fonction et cette variable portera le même nom que la fonction. Ainsi, la syntaxe serait :

Function nom-de-la-fonction(arguments)
... code des instructions ...
nom-de-la-fonction = expression
... code des instructions ...
End Function

Vbscript sait alors automatiquement que vous voulez renvoyer la valeur de la "variable" nom-de-la-fonction au programme qui l'a appelé. Cette façon de faire remplace le return d'autres langages de programmation ou de scripts.

Voici un exemple :

Dim cube
Function cube(nombre)
cube = nombre*nombre*nombre
End Function
La valeur retournée sera celle de "cube" soit nombre au cube.

11.6 Appel d'une fonction

Pour appeler une fonction, on fait apparaître le nom de la fonction à droite d'une affectation de variable souvent appelée variable de retour. La syntaxe est :

variable-retour = nom-de-la-fonction(arguments)

Dans le cadre de notre exemple, cela pourrait être :
resultat = cube(nombre)

Les parenthèses sont optionnelles même si on transmet des arguments à la fonction.

12. Les conditions

12.1 L'expression If

A un moment ou à un autre de la programmation, on aura besoin de tester une condition. Ce qui permettra d'exécuter ou non une série d'instructions.

Dans sa formulation la plus simple, l'expression if se présente comme suit

```
If condition=vraie Then
... suite d'instructions ...
End If
```

Ainsi, si la condition est vérifiée, les instructions s'exécutent. Si elle ne l'est pas, les instructions ne s'exécutent pas et le programme passe à la commande suivant le End If de fermeture.

De façon un peu plus évoluée, il y a l'expression If...Else

```
If condition=vraie Then
... instructions1 ...
Else
... instructions2 ...
End If
```

Si la condition est vérifiée (true), le bloc d'instructions 1 s'exécute. Si elle n'est pas vérifiée (false), le bloc d'instructions 2 s'exécute.

12.2 L'expression For ... Next

L'expression For ... Next permet d'exécuter un bloc d'instructions un certain nombre de fois.
Sa syntaxe est :

```
For compteur = début to fin
... instructions répétées ...
Next
```

où compteur est une simple variable

Prenons un exemple concret

```
For i = 0 to10
document.write i & "<BR>"
Next
```

Au premier passage, la variable i, étant initialisée à 0, vaut bien entendu 0. Le script affiche 0. Elle est alors incrémentée (par défaut) d'une unité. La variable vaut alors 1 et le script affiche 1. On incrémente de 1, la variable vaut alors 2 et le script affiche 2. Ainsi de suite jusqu'à 10 et le script affichera 10 (donc valeur de fin comprise). Ensuite la boucle est interrompue et le script passe à l'instruction suivant le Next.

On peut aussi déterminer la valeur d'incrémentation ou de décrémentation. For... Next devient :

```
For compteur = début to fin Step incrément
... instructions répétées ...
Next
```

Ainsi pour afficher que les nombres pairs dans notre petit exemple, celui-ci devient :

```
For i = 0 to10 Step 2
document.write i & "<BR>"
Next
```

Au premier passage, la variable i, étant initialisée à 0, vaut bien entendu 0. Le script affiche 0. Elle est alors incrémentée de la valeur de Step soit 2. La variable vaut alors 2 et le script affiche 2. On incrémente de 2 (valeur de Step), la variable vaut alors 4 et le script affiche 4. Ainsi de suite jusqu'à 10 et le script affichera 10 (donc valeur de fin comprise). Ensuite la boucle est interrompue et le script passe à l'instruction suivant le Next.

Les boucles sont toujours à manier avec précautions car elles risquent de boucler très (trop) longtemps ou indéfiniment. Un conseil est d'éviter Step 0.

12.3 Autres tests conditionnels

Il existe d'autres tests de condition, par exemple Do ... Loop et ses variantes avec While et Until, mais cela dépasse le cadre fixé d'un simple tour d'horizon.

13. Les événements

13.1 Généralités

Avec les événements et surtout leur gestion, nous abordons le côté "magique" des langages de scripts comme VBscript et Javascript. En Html classique, il y a un événement que vous connaissez bien. C'est le clic de la

souris sur un lien pour vous transporter sur une autre page Web. Les langages de scripts vont vous permettre d'appliquer et de gérer ces événements sur les objets de votre choix.

Les événements des langages de script et leur gestion, associés aux fonctions, aux méthodes et aux formulaires, ouvrent grand la porte pour une réelle interactivité de vos pages Web.

13.2 Les événements

Passons en revue quelques événements implémentés en VBscript.

Description	Evénement
Lorsque l'utilisateur clique sur un bouton, un lien ou tout autre élément.	Clik
Lorsqu'un élément de formulaire a le focus c-à-d devient la zone d'entrée active.	Focus
Lorsqu'un élément de formulaire perd le focus c-à-d que l'utilisateur clique hors du champs et que la zone d'entrée n'est plus active.	Blur
Lorsque l'utilisateur sélectionne un champ dans un élément de formulaire.	Select

13.3 Les procédures événementielles

Pour être efficace, il faut qu'à ces événements soient associés les actions prévues par vous. C'est le rôle des procédures événementielles.

La syntaxe est :

```
nom-de-l'objet_On-Evénement()
```

Ainsi pour une action déclenchée par le clic de l'utilisateur sur un bouton nommé "test", la procédure événementielle à créer par vous sera :

```
test_OnClick()
```

Comme il s'agit d'une procédure, on utilisera Sub et End :

```
Sub test_OnClick()
... action ...
End Sub
```

Comme ses procédures événementielles doivent être définies avant qu'elles soient appelées, il est recommandé de toujours les incorporer dans les balises <HEAD></HEAD>.

13.4 Exemple complet

Soit un bouton nommé test qui au click de l'utilisateur, provoque l'affichage d'une boite de message (Internet Explorer seulement)

```
<HTML>
<HEAD>
<SCRIPT language="VBscript">
Sub test_OnClick()
MsgBox "Test réussi!"
End Sub
</SCRIPT>
</HEAD>
<BODY>
<FORM>
<INPUT TYPE="button" NAME="test" VALUE="Pour un test">
</FORM>
```

```
</BODY>
</HTML>
```

13.5 Remarque finale

On verra dans le chapitre suivant que dans le cas particulier des formulaires (au sens Html du terme) cette déclaration des procédures événementielles peut être allégée.

14. Les contrôles de formulaire

14.1 Généralités

Avec les langages de script, les formulaires Html prennent une toute autre dimension. N'oublions pas qu'en VBscript (comme en Javascript), on peut accéder à chaque élément d'un formulaire pour, par exemple, y aller lire ou écrire une valeur, noter un choix auquel on pourra associer un gestionnaire d'événement... Tous ces éléments renforceront grandement les capacités interactives de vos pages.

Mettons au point le vocabulaire que nous utiliserons. Un formulaire est l'élément Html (à ne pas confondre avec le sens Visual Basic du terme) déclaré par les balises <FORM></FORM>. Un formulaire contient un ou plusieurs éléments que nous appellerons des contrôles (widgets). Ces contrôles sont notés par exemple par la balise <INPUT TYPE= ...>.

14.2 Procédure événementielle simplifiée (ou explicite)

Plutôt que de créer une procédure événementielle comme décrit dans le chapitre précédent (procédure d'événement implicite), on peut inclure l'événement comme un attribut de la balise <INPUT...> du formulaire. Cet attribut prendra la forme :

```
<INPUT ... On-Evénement="instruction">
```

Soit dans le cas de notre bouton "test" qui affiche une boite de message au clic de l'utilisateur :

```
<INPUT type="button" name="test" value="Pour un test" OnClick="MsgBox 'Test réussi!'">
```

Mais cette déclaration de procédure n'est pas encore complète. Il faut encore spécifier au browser à quel langage de script vous faites référence. Dans le cas présent du VBscript et non du Javascript (assumé par défaut).

```
<INPUT type="button" name="test" value="Pour un test" language="VBscript" OnClick="MsgBox
'Test réussi!'">
```

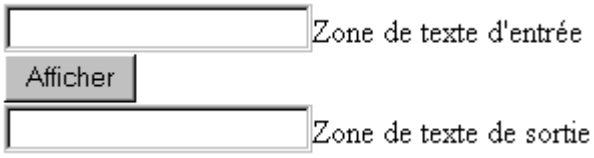
Ce genre de notation, assurément empruntée à Javascript, est fort pratique mais son usage intensif compliquera la relecture de vos scripts.

14.3 Le contrôle ligne de texte

Nous passons en ci-dessous quelques contrôles de formulaire avec une application en VBscript.

La ligne de texte est l'élément d'entrée/sortie par excellence de langage de script. La syntaxe Html est <INPUT TYPE="text" NAME="nom" SIZE=x MAXLENGTH=y> pour un champ de saisie d'une seule ligne, de longueur x et de longueur maximale de y.
L'objet text possède trois propriétés : name, defaultvalue, value.

Prenons un exemple tout simple. Entrez une valeur quelconque dans la zone de texte d'entrée. Appuyer sur le bouton pour afficher cette valeur dans la zone de texte de sortie.



Le code est :

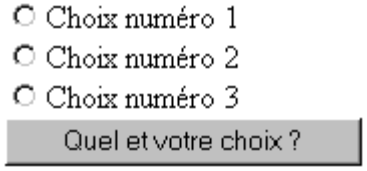
```
<HTML>
<HEAD>
<SCRIPT language="VBscript">
Function afficher(form2)
test =document. form2.input.value
document.form2.output.value=test
End Function
</SCRIPT>
</HEAD>
<BODY>
<FORM NAME="form2">
<INPUT TYPE="text" NAME="input" VALUE=""> Zone de texte d'entrée <BR>
<INPUT TYPE="button" NAME="bouton" VALUE="Afficher" onClick="afficher(form2)"><BR>
<INPUT TYPE="text" NAME="output" VALUE=""> Zone de texte de sortie
</FORM>
</BODY>
</HTML>
```

Lorsqu'on clique le bouton "Afficher" (onClick), VBscript appelle la fonction afficher() à laquelle on passe comme argument le formulaire dont le nom est form2 .
Cette fonction afficher() définie dans les balises <HEAD> prend sous la variable test, la valeur de la zone de texte d'entrée. Pour accéder à cette valeur, on note le chemin complet de celle-ci (voir le chapitre "Un peu de théorie objet"). Soit dans le document présent, il y a l'objet formulaire appelé form2 qui contient le contrôle de texte nommé input et qui a comme propriété l'élément de valeur value. Ce qui donne document.form2.input.value.A l'instruction suivante, on dit à VBscript que la valeur de la zone de texte output comprise dans le document.formulaire nommé form2 est celle de la variable test. A nouveau, on a utilisé le chemin complet pour arriver à la propriété valeur de l'objet souhaité soit en VBscript form2.output.value.

14.4 Les boutons radio

Les boutons radio sont utilisés pour noter un choix, et seulement un seul, parmi un ensemble de propositions. L'objet radio possède plusieurs propriétés : name, value, item(pour le rang du bouton en commençant par 0, checked et defaultchecked.

Prenons un exemple :




```
<HTML>
<HEAD>
<SCRIPT language="VBscript">
Function choix(form3)
If (form3.choix.item(0).checked) Then
alert("Choix " + form3.choix.item(0).value)
```

```
End If
If (form3.choix.item(1).checked) Then
alert("Choix " + form3.choix.item(1).value)
End If
If (form3.choix.item(2).checked) Then
alert("Choix " + form3.choix.item(2).value)
End If
End Function
</SCRIPT>
</HEAD>
<BODY> Entrez votre choix :
<FORM NAME="form3">
<INPUT TYPE="radio" NAME="choix" VALUE="1">Choix numéro 1<BR>
<INPUT TYPE="radio" NAME="choix" VALUE="2">Choix numéro 2<BR>
<INPUT TYPE="radio" NAME="choix" VALUE="3">Choix numéro 3<BR>
<INPUT TYPE="button"NAME="but" VALUE="Quel et votre choix ?" onClick="choix(form3)">
</FORM>
</BODY>
</HTML>
```

PS: Ce programme a été écrit avec un souci didactique. On pourrait l'écrire avec des codes plus compacts.

Dans le formulaire nommé form3, on déclare trois boutons radio. Notez que l'on utilise le même nom pour les trois boutons. Vient ensuite un bouton qui déclenche la fonction choix().
Cette fonction teste quel bouton radio est coché. On accède aux boutons sous forme d'indice par rapport à la propriété item du bouton radio nommé "choix" soit choix.item(0), choix.item(1) et choix.item(2). On teste la propriété checked du bouton par if(form3.choix.item(x).checked). Dans l'affirmative, une boite d'alerte s'affiche. Ce message reprend la valeur attachée à chaque bouton par le chemin form3.choix.item(x).value.

14.5 Liste de sélection

Le contrôle liste de sélection vous permet de proposer diverses options sous la forme d'une liste déroulante dans laquelle l'utilisateur peut cliquer pour faire son choix. Ce choix reste alors affiché.
La boite de la liste est crée par la balise <SELECT> et les éléments de la liste par un ou plusieurs tags <OPTION>. La balise </SELECT> termine la liste. Comme propriétés, on a name, length, selectedIndex (à partir de 0), defaultselected,

Un petit exemple comme d'habitude :

Entrez votre choix : Image

```
<HTML>
<HEAD>
<SCRIPT language="VBscript">
Function liste(form5)
MsgBox "Elément " & (form5.list.selectedIndex + 1)
End Function
</SCRIPT>
</HEAD>
<BODY> Entrez votre choix :
<FORM NAME="form5">
<SELECT NAME="list">
<OPTION VALUE="1">Elément 1
<OPTION VALUE="2">Elément 2
<OPTION VALUE="3">Elément 3
</SELECT>
<INPUT type="button" name="b" value="Quel est l'élément retenu?" onClick="liste(form5)">
</FORM>
```

```
</BODY>
</HTML>
```

Dans le formulaire nommé form5, on déclare une liste de sélection par la balise <SELECT> </SELECT>. Entre ses deux balises, on déclare les différents éléments de la liste par autant de tags <OPTION>. Vient ensuite un bouton qui déclenche la fonction liste().
Cette fonction teste quelle option a été sélectionnée. Le chemin complet de l'élément sélectionné est form5.nomdelaliste.selectedIndex. Comme l'index commence à 0, il ne faut pas oublier d'ajouter 1 pour retrouver le juste rang de l'élément.

Le même exemple avec la procédure événementielle simplifiée décrite plus haut, donnerait :

```
</SCRIPT>
</HEAD>
<BODY> Entrez votre choix :
<FORM NAME="form5">
<SELECT NAME="list">
<OPTION VALUE="1">Elément 1
<OPTION VALUE="2">Elément 2
<OPTION VALUE="3">Elément 3
</SELECT>
<INPUT TYPE="button"NAME="b" VALUE="Quel est l'élément retenu?"
language="VBscript"
onClick="MsgBox 'Elément ' & (form5.list.selectedIndex + 1)">
</FORM>
</BODY>
</HTML>
```

14.6 Pour terminer

Il existe d'autres contrôles de formulaires (textarea, submit, bouton checkbox...) que nous vous laissons découvrir lors de vos études ultérieures de VBscript.

15. A propos des tableaux

15.1 Les tableaux de longueur fixe à une dimension

Pour créer des tableaux en VBscript, on utilise la notion de variable (donc avec le mot clé Dim) mais qui sera indexée.

Pour faire un tableau, il faut procéder en deux étapes :

- d'abord construire la structure du tableau. A ce stade, les éléments du tableau sont vides.
- ensuite affecter des valeurs dans les cases ainsi définies.

On commence par définir le tableau :

```
Dim nom-du-tableau (x)
```
où x est le nombre d'éléments du tableau moins 1.

Ensuite, on va alimenter la structure ainsi définie :

```
nom-du-tableau(i) = "élément"
```
où i est un nombre compris entre 0 et x moins 1.

Exemple : un carnet d'adresse avec 3 personnes

construction du tableau :

Dim carnet(2)

ajout des données :
carnet(0)="Dupont"
carnet(1)="Médard"
carnet(2)="Van Lancker"

pour accéder un élément, on emploie :
document.write carnet(2)

On notera ici que les données sont bien visibles au lecteur un peu initié (view source).

Remarque :

Quand on en aura l'occasion, il sera pratique de charger le tableau avec une boucle. Supposons que l'on ait à charger 50 images. Soit on le fait manuellement, il faut charger 0.gif, 1.gif, 2.gif... Soit on utilise une boucle du style :

```
Dim gif(49)
For i = 0 to 49
gif(i) = i + ".gif"
Next
```

15.2 Tableau dynamique à une dimension

L'avantage des tableaux dynamiques est, bien entendu, qu'il ne faut pas connaître à l'avance le nombre d'éléments du tableau.

On peut utiliser à tout moment dans le script ReDim pour redimensionner le tableau. La syntaxe est :

```
ReDim nom-du-tableau (x)
où x est le nombre d'éléments du tableau moins 1 ou une variable.
```

Voici un exemple de tableau dont le nombre d'éléments est déterminé par une variable i :

```
Dim carnet()
Dim i
i = 3
ReDim carnet(i - 1)
et plus loin dans le script
ReDim carnet (i + 9)
```

Attention ! A chaque appel de ReDim, le contenu du tableau précédent est effacé.

Si cette option ne vous convient pas et que vous voulez "préserver" vos données, heureusement pour vous il existe le mot clé Preserve qui vous permet de conserver les données initiales.

```
ReDim Preserve carnet (i + 9)
```

15.3 Pour connaître la taille d'un tableau

Ce renseignement vous est accessible par la fonction VBscript UBound. La syntaxe n'a rien de fort compliqué :

```
variable = UBound(nom-du-tableau)
```

Appliqué à notre exemple, on peut imaginer :

```
MsgBox "Le nombre d'éléments est " & UBound(carnet)
Le résultat est bien entendu - Le nombre d'éléments est 2 -
```

15.4 Les tableaux à plusieurs dimensions

VBscript vous permet de construire aisément des tableaux à plusieurs dimensions (jusqu'à 60 dimensions).

La syntaxe de construction du tableau est :

```
Dim Grid(x, y)
```
où x est le nombre de lignes moins 1 et y le nombre de colonnes moins 1.

Pour alimenter la structure ainsi définie :

```
Grid(i,j) = "élément"
```
où i est un nombre compris entre 0 et x moins 1 et j un nombre compris entre 0 et y moins 1.

A titre d'exemple :

```
Dim Grid(3,3)
Grid(0,0)="Dupont"
Grid(1,0)="Médard"
Grid(2,0)="Van Lancker"
document.write Grid(2,0)
```

16. Incorporation des ActiveX

16.1 Généralités

Votre page Web se compose de balises Html et éventuellement du code VBscript. Vous pouvez "greffer" sur cette page des objets, extérieurs à celle-ci, qui vous permettront de réaliser des applications graphiques et/ou de programmation que le tandem Html/Script ne peut réaliser.

Les contrôles ActiveX (comme les objets Java) font partie de ses objets que l'on peut "greffer" sur une page Web.

16.2 La balise <OBJECT>

Pour incorporer un objet, on utilise la balise <OBJECT></OBJECT>

Cette balise nécessite une série d'attributs :

classid référence l'emplacement où le browser peut trouver le contrôle ActiveX d'abord sur le système de l'utilisateur et ensuite où il pourra être éventuellement téléchargé sur le Web. Ainsi, classid sera l'identificateur unique du composant. Celui-ci s'indique sur 128 bits sous forme d'un charabia illisible.

id désigne l'identificateur ou le nom de l'élément. C'est le nom que vous utiliserez chaque fois que vous ferez appel à l'ActiveX.

ensuite toute une série d'attributs (width, height, vspace, hspace, align) qui ont trait à la taille qu'occupera l'objet sur la page Web et à la représentation visuelle de celui-ci.

Ainsi une balise d'incorporation d'objet peut prendre la forme suivante :

```
<OBJECT
        classid="clsid:99B42120-6EC7-11CF-A6C7-00AA00A47DD2"
        id=lb1ActiveLb1
        width=250
        height=250
        align=left
```

```
hspace=20
vspace=0 >
```

16.3 La balise <PARAM>

Avec la balise <OBJECT>, vous avez importé dans votre page un petit objet. Ce petit programme doit maintenant fonctionner selon vos spécifications. Dans le cadre d'une présentation graphique de texte plus évoluée que celle permise en Html, on peut imaginer que ces paramètres soient votre texte, la police, la taille de la police, l'angle du texte, etc.

Ces balises <PARAM> pourraient prendre la forme suivante :

```
<PARAM NAME="Caption" VALUE="Mon texte">
<PARAM NAME="Fontname" VALUE="Arial, Helvetia">
<PARAM NAME="Fontsize" VALUE="20">
<PARAM NAME="Angle" VALUE="90">
</OBJECT>
```

Ces différents paramètres vous sont généralement fournis avec la documentation accompagnant le contrôle ActiveX.

16.4 ActiveX Control Pad

Microsoft met à votre disposition ActiveX Control Pad pour insérer aisément (?) des contrôles ActiveX dans vos pages Web en Html.

17. Les messages d'erreurs

On ne peut pas dire que les outils de débuggage offerts par VBscript soient des plus évolués. Pour un peu, on se croirait revenu aux temps préhistoriques du Basic. Ainsi, corriger vos erreurs en VBscript ressemble souvent au jeu des sept erreurs.

Pour ceux qui connaissent le Javascript, la gestion des erreurs de VBscript est du même niveau que celle de Javascript, donc faible et peu évoluée.

17.1 Les types d'erreurs.

Il y a 3 grandes catégories d'erreurs dans l'utilisation d'un programme VBscript :

- les erreurs au chargement.
- les erreurs à l'exécution
- les erreurs de logique.

17.2 Les erreurs au chargement

Au chargement du script par le browser, l'interpréteur VBscript passe en revue les différentes erreurs qui peuvent empêcher le bon déroulement de celui-ci.
Les erreurs au chargement, nombreuses lors de l'apprentissage de VBscript, sont souvent dues à des fautes de frappe et/ou des erreurs de syntaxe.
Pour vous aider à déterminer l'erreur, VBscript affiche sa fameuse boite de message d'erreur, vous indique le problème et le texte de l'erreur. Des exemples classiques d'erreurs au chargement sont des strings absents, des minuscules au lieu de majuscules, des variables mal orthographiées.

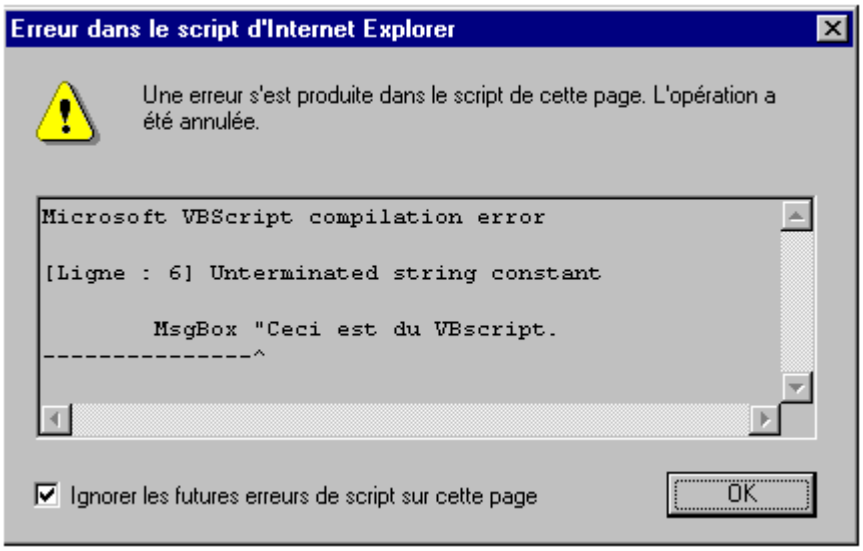
Le gestionnaire d'erreur de VBscript bloque le script et il ne vous reste plus qu'à corriger votre erreur. Voici un exemple de script avec une erreur :

```
<HTML>
<HEAD>
<SCRIPT LANGUAGE="vbscript">
```

```
<!--
Sub Bouton1_OnClick
MsgBox "Ceci est du VBscript.
End Sub
-->
</SCRIPT>
</HEAD>
<BODY> etc.
```

Vous aurez remarqué qu'il manque les guillemets de fin au string "Ceci est du VBscript".

VBscript affichera le message d'erreur suivant :



Le message d'erreur vous signale qu'à la ligne 6, il y a un string qui n'est pas terminé par des guillemets. Notez cependant que le message n'est pas toujours aussi précis et qu'il faut parfois remonter de plusieurs lignes pour trouver l'erreur effective.

17.3 Les erreurs à l'exécution

Ici votre script se charge sans problème, mais cette satanée boite de message d'erreurs apparaît lorsque l'exécution du script est demandée. Alors que les erreurs au chargement étaient surtout dues au mauvais usage de la syntaxe, les erreurs à l'exécution (runtime error) proviennent d'un mauvais usage des commandes ou des objets VBscript.
Un exemple d'erreur à l'exécution est un appel erroné à une variable ou une fonction inexistante (car il y a, par exemple, une erreur dans le nom de la variable ou de la fonction).

17.4 Les erreurs de logique

Ce sont les plus vicieuses car le "débuggeur" de VBscript ne signale bien entendu aucune erreur et votre script se déroule correctement. Hélas, à l'arrivée, le résultat ne correspond pas à celui espéré.
Il n'y a plus qu'à revoir la construction logique de votre script. Cent fois sur le métier remettez votre ouvrage...

De nombreuses erreurs de logique sont dues à des valeurs de variables incorrectes.
Voici quelques conseils :

- Dans le cas où l'utilisateur doit entrer une valeur, celle-ci était-t-elle au bon format? N'est-il pas utile de prévoir un petit script pour vérifier le format d'entrée ?
- On peut ajouter des points de contrôle de valeur de variable ou de passage avec l'instruction MsgBox variable ou MsgBox "Point de passage1"

17.5 Les grands classiques des erreurs.

On peut dresser une liste d'erreurs que tous les débutants (et même certains programmeurs confirmés) font ou feront tôt ou tard.

- Soyez vigilant au nom des variables (case sensitive). Mavariable et mavariable sont deux variables distinctes. Eviter d'utiliser des noms de variables trop rapprochants.
- Le nom de la fonction a-t-il bien la même orthographe dans la déclaration et dans l'appel. Le nom des fonctions est-il bien unique dans le script?
- N'oubliez pas les guillemets avant et après les chaînes de caractères.
- Avez-vous bien mis des virgules entre vos différents paramètres ou arguments?
- Avez-vous placé votre déclaration de fin (End ...)? Avez-vous placé la déclaration de fin au bon endroit dans le cas de blocs de commandes imbriquées?
- Assurez-vous que les noms des objets VBscript sont corrects. Le piège est que les objets ou fonctions VBscript commencent par une majuscule (MsgBox, Day, Isdate...) mais que les propriétés commencent par une minuscule (write).
- La confusion entre = opérateur d'affectation et = opérateur de comparaison.

17.6 On Error Resume Next

L'expression On Error Resume Next, propre à VBscript, permet de poursuivre l'exécution d'un programme après une erreur à l'exécution (runtime uniquement) et empêcher ainsi qu'il soit bloqué par une fenêtre de message d'erreur. VBscript ignore l'erreur et passe à la ligne suivante.

Exemple :

```
<HTML>
<BODY>
<SCRIPT language="VBS">
a = 100 / 0
MsgBox a
</SCRIPT>
</BODY>
</HTML>
```

VBscript affiche une fenêtre d'erreur (runtime error division par 0).

```
<HTML>
<BODY>
<SCRIPT language="VBS">
On Error Resume Next
a = 100 / 0
MsgBox a
</SCRIPT>
</BODY>
</HTML>
```

VBscript ignore l'erreur et affiche une boite de message vide (car a n'a pas été calculé).

Soulignons que cette astuce ne résout en rien l'erreur. Elle l'ignore simplement, ce qui désactive les messages d'erreur.

17.7 L'objet err

Avec l'objet err (objet interne de VBscript) et surtout ses propriétés, il est possible de se fabriquer un mini outil de déboggage.

err.number	Contient le code d'erreur numérique lorsqu'une erreur se produit. S'il n'y a pas d'erreur, la valeur est 0. Chaque type d'erreur a son propre code.
err.description	Contient une description de l'erreur correspondant au numéro d'erreur.

L'exemple précédent devient :

```
<HTML>
<BODY>
<SCRIPT language="VBS">
```

```
On Error Resume Next
a = 100 / 0
If err.number <> 0 then
MsgBox "Erreur : " & err.number & " " & err.description
End if
</SCRIPT>
</BODY>
</HTML>
```

On teste s'il y a une erreur (err.number différent de 0). Avec une boite de message, on affiche le numéro de code de l'erreur (err.number) et sa description (err.description).

La boite de message affiche dans le cas présent Erreur : 11 Division by zero

---

Apprendre le VBscript
www.ccim.be/ccim328/vb/index.htm

L'auteur :

Van Lancker Luc
Rue des Brasseurs, 22
7700 Mouscron
Belgium
Vanlancker.Luc@ccim.be

Un mot d'encouragement ou un compliment fait toujours plaisir.
Critiques et suggestions seront aussi examinées avec attention.

Du même auteur :

Apprendre le langage Html	www.ccim.be/ccim328/html/index.htm
Maîtriser le langage Html	www.ccim.be/ccim328/htmlplus/index.htm
Apprendre le Javascript	www.ccim.be/ccim328/js/index.htm
Apprendre à créer un site	www.ccim.be/ccim328/site/index.htm