

Javascript_Part2

1. JavaScript

JavaScript est un langage de scripts qui, incorporé aux balises Html, permet d'améliorer la présentation et **l'interactivité** des pages Web.

JavaScript est donc une extension du code Html des pages Web. Les scripts, *qui s'ajoutent ici aux balises Html*, peuvent en quelque sorte être comparés aux macros d'un traitement de texte.

Ces scripts vont être gérés et exécutés par le browser lui-même sans devoir faire appel aux ressources du serveur. Ces instructions seront donc traitées en direct et surtout sans retard par le navigateur.

JavaScript a été initialement développé par Netscape et s'appelait alors LiveScript. Adopté à la fin de l'année 1995, par la firme Sun (qui a aussi développé Java), il prit alors son nom de Javascript.

Javascript n'est donc pas propre aux navigateurs de Netscape (bien que cette firme en soit un fervent défenseur). Microsoft l'a d'ailleurs aussi adopté à partir de son Internet Explorer 3. On le retrouve, de façon améliorée, dans Explorer 4.

Les versions de Javascript se sont succédées avec les différentes versions de Netscape : Javascript pour Netscape 2, JavaScript 1.1 pour Netscape 3 et Javascript 1.2 pour Netscape 4. Ce qui n'est pas sans poser certains problèmes de compatibilité, selon le browser utilisé, des pages comportant du code Javascript. Mais consolons nous en constatant qu'avec MSIE 3.0 ou 4.0 et la famille Netscape, une très large majorité d'internautes pourra lire les pages comprenant du Javascript.

L'avenir de Javascript est entre les mains des deux grands navigateurs du Web et en partie lié à la guerre que se livrent Microsoft et Netscape. On s'accorde à prédire un avenir prometteur à ce langage surtout de par son indépendance vis à vis des ressources du serveur.

2. JavaScript n'est pas Java

Il importe de savoir que JavaScript est totalement différent de Java. Bien que les deux soient utilisés pour créer des pages Web évoluées, bien que les deux reprennent le terme Java (café en américain), nous avons là deux outils informatiques bien différents.

JavaScript	Java
<ol style="list-style-type: none"> 1. Code intégré dans la page Html 2. Code interprété par le browser au moment de l'exécution 3. Codes de programmation simple mais pour des applications limitées 4. Permet d'accéder aux objets du navigateur 5. Confidentialité des codes nulle (code source visible) 	<ol style="list-style-type: none"> 1. Module (applet) distinct de la page Html 2. Code source compilé avant son exécution 3. Langage de programmation beaucoup plus complexe mais plus performant 4. N'accède pas aux objets du navigateur 5. Sécurité (code source compilé)

3. Vos outils pour le JavaScript

Pour apprendre et exploiter le JavaScript, il vous faut :

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

3.1. Un browser compatible Javascript

Uniquement Netscape et Microsoft vous proposent des navigateurs Javascript "enabled". Pour Microsoft à partir de MSIE Explorer 3.0 et Netscape à partir de Netscape Navigator 2.0.

Par contre, il faut être attentif aux versions de Javascript exploitées par ces browsers.

Netscape 2.0	Javascript (baptisé à posteriori 1.0)
Netscape 3.0	Javascript 1.1
Netscape 4.0	Javascript 1.2

Explorer 3.0	Quelque chose qui ressemble à du Javascript 1.0
Explorer 4.0	Javascript 1.2

Il faut bien admettre que Javascript est plutôt l'affaire de Netscape et que vous courez au devant d'une collection d'ennuis en utilisant Explorer 3 pour le Javascript.

3.2. Un solide bagage en Html

Comme le code du Javascript vient s'ajouter au "code" du langage Html, une connaissance approfondie des balises ou tags Html est souhaitable sinon indispensable. Ainsi les utilisateurs d'éditeurs Html "whsiwyg" ou autres "publishers" Html risquent de devoir retourner à leurs chères études.

Je ne peux que vous recommander un tutorial du langage Html du même auteur. "Apprendre le langage Html" à l'adresse www.ccim.be/ccim328/html/index.htm

3.3. Un bon éditeur de texte

Une page Html n'est que du texte. Le code Javascript n'est lui aussi que du texte. Quoi de plus simple qu'un éditeur de ... texte comme le Notepad de Windows pour inclure votre Javascript dans votre page Html. Un éditeur Html de la première génération (un bon vieil éditeur qui fait encore apparaître les balises), comme HTML Notepad, fait également bien l'affaire.

De plus en plus d'éditeurs Html whsiwyg proposent une fenêtre Javascript. Attention ! Si certains semblent bien faits comme WebExpert 2 (en français) avec d'autres, il arrive que le code Javascript introduit soit modifié par l'éditeur comme FrontPage ou Netscape Gold. A vos expériences...

Ajoutons que l'on commence à voir des programmes "Visual Javascript" mais ils me semblent très lourds à gérer pour n'ajouter finalement que quelques lignes.

4. Le JavaScript minimum

4.1 La balise <SCRIPT>

De ce qui précède, vous savez déjà que votre script vient s'ajouter à votre page Web.

Le langage Html utilise des tags ou balises pour "dire" au browser d'afficher une portion de texte en gras, en italique, etc.

Dans la logique du langage Html, il faut donc signaler au browser par une balise, que ce qui suit est un script et que c'est du Javascript (et non du VBScript). C'est la balise `<SCRIPT LANGUAGE="Javascript">`.

De même, il faudra informer le browser de la fin du script.

C'est la balise `</SCRIPT>`.

4.2. Les commentaires

Il vous sera peut-être utile d'inclure des commentaires personnels dans vos codes Javascript. C'est même vivement recommandé comme pour tous les langages de programmation (mais qui le fait vraiment ?).

Javascript utilise les conventions utilisées en C et C++ soit

```
// commentaire
```

Tout ce qui est écrit entre le // et la fin de la ligne sera ignoré.

Il sera aussi possible d'inclure des commentaires sur plusieurs lignes avec le code
/* commentaire sur
plusieurs lignes */

Ne confondez pas les commentaires Javascript et les commentaires Html
(pour rappel <!-- ...-->).

4.3 Masquer le script pour les anciens browsers

Les browsers qui ne comprennent pas le Javascript (et il y en a encore) ignorent la balise <script> et vont essayer d'afficher le code du script sans pouvoir l'exécuter. Pour éviter l'affichage peu esthétique de ses inscriptions cabalistiques, on utilisera les balises de commentaire du langage Html <!-- ... -->.

Votre premier Javascript ressemblera à ceci :

```
<SCRIPT LANGUAGE="javascript">  
<!-- Masquer le script pour les anciens browsers  
...  
programme Javascript  
...  
// Cesser de masquer le script -->  
</SCRIPT>
```

4.4. Où inclure le code en Javascript ?

Le principe est simple. Il suffit de respecter les deux principes suivants :

4.4.1. n'importe où.

4.4.2. mais là où il le faut.

Le browser traite votre page Html de haut en bas (y compris vos ajoutes en Javascript). Par conséquent, toute instruction ne pourra être exécutée que si le browser possède à ce moment précis tous les éléments nécessaires à son exécution. Ceux-ci doivent donc être déclarés avant ou au plus tard lors de l'instruction.

Pour s'assurer que le programme script est chargé dans la page et prêt à fonctionner à toute intervention de votre visiteur (il y a des impatientes) 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 fonctions.

Rien n'interdit de mettre plusieurs scripts dans une même page Html.

Il faut noter que l'usage de la balise script n'est pas toujours obligatoire. Ce sera le cas des événements Javascript (par exemple onClick) où il faut simplement insérer le code à l'intérieur de la commande Html comme un attribut de celle-ci. L'événement fera appel à la fonction Javascript lorsque la commande Html sera activée. Javascript fonctionne alors en quelque sorte comme une extension du langage Html.

4.5. Une première instruction Javascript

Sans vraiment entrer dans les détails, voyons une première instruction Javascript (en fait une méthode de l'objet window) soit l'instruction alert().

```
alert("votre texte");
```

Cette instruction affiche un message (dans le cas présent votre texte entre les guillemets) dans une boîte de dialogue pourvue d'un bouton OK. Pour continuer dans la page, le lecteur devra cliquer ce bouton.

Vous remarquerez des points-virgules à la fin de chaque instruction Javascript (ce qui n'est pas sans rappeler le C et le C++). Le Javascript, bon enfant, est moins strict que ces autres langages et ne signale généralement pas de message d'erreur s'ils venaient à manquer. On peut considérer que le point-virgule est optionnel et qu'il n'est obligatoire que lorsque vous écrivez plusieurs instructions sur une même ligne. On recommande quand même vivement dans la littérature d'en mettre de façon systématique.

Javascript est "bon enfant" car il n'est pas toujours trop strict sur la syntaxe et passe au-dessus de certaines libertés prises avec celle-ci. Très bien! Mais ce caractère "bon enfant" est à double tranchant car parfois, pour une raison indéterminée, il devient dans certaines situations plus rigoureux et alors bonne chance pour déboguer votre script.

4.6. Votre première page Html avec du Javascript

```
<HTML>
<HEAD>
<TITLE>Mon premier Javascript</TITLE>
</HEAD>
<BODY>
Bla-bla en Html
<SCRIPT LANGUAGE="Javascript">           ← Début du script
<!--
alert("votre texte");
//-->
</SCRIPT>                                ← Masquer le script
Suite bla-bla en Html
</BODY>
</HTML>
```

4.7. Remarques

Javascript est case sensitive. Ainsi il faudra écrire alert() et non Alert(). Pour l'écriture des instructions Javascript, 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.

Les guillemets " et l'apostrophe ' font partie intégrante du langage Javascript. On peut utiliser l'une ou l'autre forme à condition de ne pas les mélanger. Ainsi alert("...") donnera un message d'erreur. Si vous souhaitez utiliser des guillemets dans vos chaînes de caractères, tapez \" ou \' pour les différencier vis à vis du compilateur.

4.8. Versions du langage Javascript

Avec les différentes versions déjà existantes (Javascript 1.0, Javascript 1.1 et Javascript 1.2), on peut imaginer des scripts adaptés aux différentes versions mais surtout aux différents navigateurs ;

```
<SCRIPT LANGUAGE="Javascript">
// programme pour Netscape 2 et Explorer 3
var version="1.0";
</SCRIPT>
```

```
<SCRIPT LANGUAGE="Javascript1.1">
// programme pour Netscape 3 et Explorer 4
var version=1.1;
</SCRIPT>
```

```
<SCRIPT LANGUAGE="Javascript1.2">
// programme pour Netscape 4
var version=1.2;
</SCRIPT>
```

```
<SCRIPT LANGUAGE="Javascript">
```

```
document.write('Votre browser supporte le Javascript ' + version);  
</SCRIPT>
```

4.9. Extension .js pour scripts externes

Il est possible d'utiliser des fichiers externes pour les programmes Javascript. On peut ainsi stocker les scripts dans des fichiers distincts (avec l'extension .js) et les appeler à partir d'un fichier Html. Le concepteur peut de cette manière se constituer une bibliothèque de script et les appeler à la manière des #include du C ou C++. La balise devient

```
<SCRIPT LANGUAGE='javascript' SRC='http://site.com/javascript.js'></SCRIPT>
```

4.10 Toujours des commentaires

Outre les annotations personnelles, les commentaires peuvent vous être d'une utilité certaine en phase de débogage d'un script pour isoler (sans effacer) une ligne suspecte.

Pour les esprits compliqués, notons que les commentaires ne peuvent être imbriqués sous peine de message d'erreur. La formulation suivante est donc à éviter :

```
/* script réalisé ce jour /* jour mois */  
et testé par nos soins*/
```

4.11 Alert() ... rouge

Joujou des débutants en Javascript, cette petite fenêtre est à utiliser avec parcimonie pour attirer l'attention du lecteur pour des choses vraiment importantes. Et puis, elles ne sont vraiment pas destinées à raconter sa vie. Javascript met à votre disposition la possibilité de créer de nouvelles fenêtres de la dimension de votre choix qui apparaissent un peu comme les popup des fichiers d'aide. Nous les étudierons plus loin dans l'objet Window.

Alert() est une méthode de l'objet Window. Pour se conformer à la notation classique nom_de_l'objet.nom_de_la_propriété, on aurait pu noter window.alert(). Window venant en tête des objets Javascript, celui-ci est repris par défaut par l'interpréteur et devient en quelque sorte facultatif.

Si vous souhaitez que votre texte de la fenêtre alert() s'inscrive sur plusieurs lignes, il faudra utiliser le caractère spécial /n pour créer une nouvelle ligne.

5. Les Formulaires

5.1 Généralités

Avec Javascript, les formulaires Html prennent une toute autre dimension. N'oublions pas qu'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 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= ...>.
```

5.2 Déclaration d'un formulaire

La déclaration d'un formulaire se fait par les balises <FORM> et </FORM>. Il faut noter qu'en Javascript, l'attribut NAME="nom_du_formulaire" a toute son importance pour désigner le chemin complet des éléments. En outre, les attributs ACTION et METHOD sont facultatifs pour autant que vous ne faites pas appel au serveur.

Une erreur classique en Javascript est, emporté par son élan, d'oublier de déclarer la fin du formulaire

</FORM> après avoir incorporé un contrôle.

5.3 Le contrôle ligne de texte

La zone de texte est l'élément d'entrée/sortie par excellence de Javascript. 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.

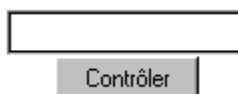
L'objet text possède trois propriétés :

Propriété	Description
name	indique le nom du contrôle par lequel on pourra accéder.
Defaultvalue	indique la valeur par défaut qui sera affichée dans la zone de texte.
Value	indique la valeur en cours de la zone de texte. Soit celle tapée par l'utilisateur ou si celui-ci n'a rien tapé, la valeur par défaut.

5.3.1 Lire une valeur dans une zone de texte

Voici un exemple que nous détaillerons :

Voici une zone de texte. Entrez une valeur et appuyer sur le bouton pour contrôler celle-ci.



The image shows a visual representation of a web form. It consists of a rectangular text input field at the top, and a button labeled "Contrôler" centered below it. The button has a grey background and a black border.

Le script complet est celui-ci :

```
<HTML>
<HEAD>
<SCRIPT LANGUAGE="javascript">
function controle(form1) {
var test = document.form1.input.value;
alert("Vous avez tapé : " + test);
}
</SCRIPT>
</HEAD>
<BODY>
<FORM NAME="form1">
<INPUT TYPE="text" NAME="input" VALUE=""><BR>
<INPUT TYPE="button" NAME="bouton" VALUE="Contrôler" onClick="controle(form1)">
</FORM>
</BODY>
</HTML>
```

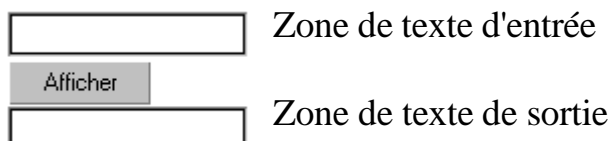
Lorsqu'on clique le bouton "contrôler", Javascript appelle la fonction controle() à laquelle on passe le formulaire dont le nom est form1 comme argument.

Cette fonction controle() définie dans les balises <HEAD> prend sous la variable test, la valeur de la zone de texte. Pour accéder à cette valeur, on note le chemin complet de celle-ci. Soit dans le document présent, il y a

l'objet formulaire appelé form1 qui contient le contrôle de texte nommé input et qui a comme propriété l'élément de valeur value. Ce qui donne document.form1.input.value.

5.3.2 Ecrire une valeur dans une zone de texte

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.



The diagram shows a simple web form layout. At the top is a rectangular input field. Below it is a button with the text 'Afficher'. At the bottom is another rectangular field, which is the output area. Labels 'Zone de texte d'entrée' and 'Zone de texte de sortie' are placed to the right of their respective fields.

Voici le code :

```
<HTML>
<HEAD>
<SCRIPT LANGUAGE="javascript">
function afficher(form2) {
var testin =document. form2.input.value;
document.form2.output.value=testin
}
</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", Javascript appelle la fonction afficher() à laquelle on passe le formulaire dont le nom est cette fois form2 comme argument.

Cette fonction afficher() définie dans les balises <HEAD> prend sous la variable testin, la valeur de la zone de texte d'entrée. A l'instruction suivante, on dit à Javascript que la valeur de la zone de texte output comprise dans le formulaire nommé form2 est celle de la variable testin. A nouveau, on a utilisé le chemin complet pour arriver à la propriété valeur de l'objet souhaité soit en Javascript document.form2.output.value.

5.4. Les boutons radio

Les boutons radio sont utilisés pour noter un choix, et seulement un seul, parmi un ensemble de propositions.

Propriété	Description
name	indique le nom du contrôle. Tous les boutons portent le même nom.
index	l'index ou le rang du bouton radio en commençant par 0.
checked	indique l'état en cours de l'élément radio
defaultchecked	indique l'état du bouton sélectionné par défaut.
value	indique la valeur de l'élément radio.

Prenons un exemple :

```
<HTML>
<HEAD>
<SCRIPT language="javascript">
```



```

function choixprop(form3) {
if (form3.choix[0].checked) { alert("Vous avez choisi la proposition " + form3.choix[0].value) };
if (form3.choix[1].checked) { alert("Vous avez choisi la proposition " + form3.choix[1].value) };
if (form3.choix[2].checked) { alert("Vous avez choisi la proposition " + form3.choix[2].value) };
}
</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="choixprop(form3)">
</FORM>
</BODY>
</HTML>

```

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

Entrez votre choix :

Choix numéro 1
 Choix numéro 2
 Choix numéro 3

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 choixprop().

Cette fonction teste quel bouton radio est coché. On accède aux boutons sous forme d'indice par rapport au nom des boutons radio soit choix[0], choix[1], choix[2]. On teste la propriété checked du bouton par if(form3.choix[x].checked). Dans l'affirmative, une boîte d'alerte s'affiche. Ce message reprend la valeur attachée à chaque bouton par le chemin form3.choix[x].value.

5.5. Les boutons case à cocher (checkbox)

Les boutons case à cocher sont utilisés pour noter un ou plusieurs choix (pour rappel avec les boutons radio un seul choix) parmi un ensemble de propositions. A part cela, sa syntaxe et son usage est tout à fait semblable aux boutons radio sauf en ce qui concerne l'attribut name.

Propriété	Description
name	indique le nom du contrôle. Toutes les cases à cocher portent un nom différent.
checked	indique l'état en cours de l'élément case à cocher.
defaultchecked	indique l'état du bouton sélectionné par défaut.
value	indique la valeur de l'élément case à cocher.

Entrez votre choix :

Il faut sélectionner les numéros 1,2 et 4 pour avoir la bonne réponse.

Choix numéro 1
 Choix numéro 2
 Choix numéro 3
 Choix numéro 4

<HTML>

```

<HEAD>
<script language="javascript">
function reponse(form4) {
if ( (form4.check1.checked) == true && (form4.check2.checked) == true && (form4.check3.checked) == false
&& (form4.check4.checked) == true)
{ alert("C'est la bonne réponse! ") }
else
{alert("Désolé, continuez à chercher.")}
}
</SCRIPT>
</HEAD>
<BODY>
Entrez votre choix :
<FORM NAME="form4">
<INPUT TYPE="CHECKBOX" NAME="check1" VALUE="1">Choix numéro 1<BR>
<INPUT TYPE="CHECKBOX" NAME="check2" VALUE="2">Choix numéro 2<BR>
<INPUT TYPE="CHECKBOX" NAME="check3" VALUE="3">Choix numéro 3<BR>
<INPUT TYPE="CHECKBOX" NAME="check4" VALUE="4">Choix numéro 4<BR>
<INPUT TYPE="button"NAME="but" VALUE="Corriger" onClick="reponse(form4)">
</FORM>
</BODY>
</HTML>

```

Dans le formulaire nommé form4, on déclare quatre cases à cocher. Notez que l'on utilise un nom différent pour les quatre boutons. Vient ensuite un bouton qui déclenche la fonction reponse(). Cette fonction teste quelles cases à cocher sont sélectionnées. Pour avoir la bonne réponse, il faut que les cases 1, 2 et 4 soient cochées. On accède aux cases en utilisant chaque fois leur nom. On teste la propriété checked du bouton par (form4.nom_de_la_case.checked). Dans l'affirmative (&& pour et logique), une boîte d'alerte s'affiche pour la bonne réponse. Dans la négative, une autre boîte d'alerte vous invite à recommencer.

5.6. 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 boîte de la liste est créée par la balise <SELECT> et les éléments de la liste par un ou plusieurs tags <OPTION>. La balise </SELECT> termine la liste.

Propriété	Description
name	indique le nom de la liste déroulante.
length	indique le nombre d'éléments de la liste. S'il est indiqué dans le tag <SELECT>, tous les éléments de la liste seront affichés. Si vous ne l'indiquez pas un seul apparaîtra dans la boîte de la liste déroulante.
selectedIndex	indique le rang à partir de 0 de l'élément de la liste qui a été sélectionné par l'utilisateur.
defaultselected	indique l'élément de la liste sélectionné par défaut. C'est lui qui apparaît alors dans la petite boîte.

Un petit exemple comme d'habitude :

Entrez votre choix :

```

<HTML>
<HEAD>
<script language="javascript"> function liste(form5) {
alert("L'élément " + (form5.list.selectedIndex + 1)); }
</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.

5.7 Le contrôle textarea (pour les bavards)

L'objet textarea est une zone de texte de plusieurs lignes.

La syntaxe Html est :

```

<FORM>
<TEXTAREA NAME="nom" ROWS=x COLS=y>
texte par défaut
</TEXTAREA>
</FORM>

```

où ROWS=x représente le nombre de lignes et COLS=y représente le nombre de colonnes.

L'objet textarea possède plusieurs propriétés :

Propriété	Description
name	indique le nom du contrôle par lequel on pourra accéder.
defaultvalue	indique la valeur par défaut qui sera affichée dans la zone de texte.
value	indique la valeur en cours de la zone de texte. Soit celle tapée par l'utilisateur ou si celui-ci n'a rien tapé, la valeur par défaut.

A ces propriétés, il faut ajouter onFocus(), onBlur(), onSelect() et onChange().

En Javascript, on utilisera \r\n pour passer à la ligne.

Comme par exemple dans l'expression document.Form.Text.value = 'Check\r\nthis\r\nout'.

5.8 Le contrôle Reset

Le contrôle Reset permet d'annuler les modifications apportées aux contrôles d'un formulaire et de restaurer les valeurs par défaut.

la syntaxe Html est :

```

<INPUT TYPE="reset" NAME="nom" VALUE "texte">
où VALUE donne le texte du bouton.

```

Une seule méthode est associée au contrôle Reset, c'est la méthode onClick(). Ce qui peut servir, par exemple,

pour faire afficher une autre valeur que celle par défaut.

5.9 Le contrôle Submit

Le contrôle a la tâche spécifique de transmettre toutes les informations contenues dans le formulaire à l'URL désignée dans l'attribut ACTION du tag <FORM>.

la syntaxe Html est :

```
<INPUT TYPE="submit" NAME="nom" VALUE "texte">
```

où VALUE donne le texte du bouton.

Une seule méthode est associée au contrôle Submit, c'est la méthode onClick().

5.10 Le contrôle Hidden (caché)

Le contrôle Hidden permet d'entrer dans le script des éléments (généralement des données) qui n'apparaîtront pas à l'écran. Ces éléments sont donc cachés. D'où son nom.

la syntaxe Html est :

```
<INPUT TYPE="hidden" NAME="nom" VALUE "les données cachées">
```

5.11 L'envoi d'un formulaire par Email.

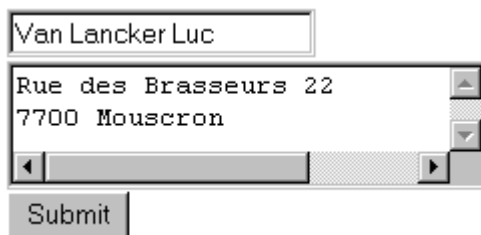
Uniquement Netscape !!!

A force de jouer avec des formulaires, il peut vous prendre l'envie de garder cette source d'information. Mais comment faire? Javascript, et à fortiori le Html, ne permet pas d'écrire dans un fichier. Ensuite, le contrôle Submit est surtout destiné à des CGI ce qui entraîne (encore) un codage spécial à maîtriser. D'autant que pour nous simples et présumés incompetents internautes, la plupart des providers ne permettra pas d'héberger une CGI faite par un amateur pour des raisons (tout à fait compréhensibles) de sécurité. Il ne reste plus que l'unique solution de l'envoi d'un formulaire via le courrier électronique.

La syntaxe est :

```
<FORM METHOD="post" ACTION="mailto:votre_adresse_Email">
<INPUT TYPE=text NAME="nom">
<TEXTAREA NAME="adresse" ROWS=2 COLS=35>
</TEXTAREA>
<INPUT TYPE=submit VALUE="Submit">
</FORM>
```

Ce qui donne :



The image shows a screenshot of a web form. At the top, there is a single-line text input field containing the text "Van Lancker Luc". Below it is a multi-line text area containing two lines of text: "Rue des Brasseurs 22" and "7700 Mouscron". At the bottom of the form is a button labeled "Submit".

Vous recevrez dans notre boîte de réception, un truc bizarre du genre :
nom=Van+Lancker+Luc&adresse=Rue+des+Brasseurs+2217OD%OA7700+Mouscron.
où on retrouve les champs nom= et adresse=, où les champs sont séparés par le signe &, où les espaces sont remplacés par le signe + et 17%OD%OA correspond à un passage à la ligne.

Attention ! Ceci ne marche que sous Netscape et pas sous Microsoft Explorer 3.0 ...
Avec Explorer, le mailto ouvre le programme de Mail mais n'envoie rien du tout.

6. Un peu de tout

6.1. Les boîtes de dialogue ou de message

Javascript met à votre disposition 3 boîtes de message :

- alert()
- prompt()
- confirm()

Ce sont toutes trois des méthodes de l'objet window.

6.2. La méthode alert()

La méthode alert() doit, à ce stade de votre étude, vous être familière car nous l'avons déjà souvent utilisée.

La méthode alert() affiche une boîte de dialogue dans laquelle est reproduite la valeur (variable et/ou chaîne de caractères) de l'argument qui lui a été fourni. Cette boîte bloque le programme en cours tant que l'utilisateur n'aura pas cliqué sur "OK".

Alert() sera aussi très utile pour vous aider à débbuger les scripts.

Sa syntaxe est :

```
alert(variable);  
alert("chaîne de caractères");  
alert(variable + "chaîne de caractères");
```

Si vous souhaitez écrire sur plusieurs lignes, il faudra utiliser le signe \n.

6.3. La méthode prompt()

Dans le même style que la méthode alert(), Javascript vous propose une autre boîte de dialogue, dans le cas présent appelée boîte d'invite, qui est composée d'un champ comportant une entrée à compléter par l'utilisateur. Cette entrée possède aussi une valeur par défaut.

La syntaxe est :

```
prompt("texte de la boîte d'invite", "valeur par défaut");
```

En cliquant sur OK, la méthode renvoie la valeur tapée par l'utilisateur ou la réponse proposée par défaut. Si l'utilisateur clique sur Annuler ou Cancel, la valeur null est alors renvoyée.

Prompt() est parfois utilisé pour saisir des données fournies par l'utilisateur. Selon certaines sources, le texte ne doit cependant pas dépasser 45 caractères sous Netscape et 38 sous Explorer 3.0.

6.4. La méthode confirm()

Cette méthode affiche 2 boutons "OK" et "Annuler". En cliquant sur OK, continue() renvoie la valeur true et bien entendu false si on a cliqué sur Annuler. Ce qui peut permettre, par exemple, de choisir une option dans un programme.

La syntaxe de l'exemple est :

```
confirm("Voulez-vous continuer ?")
```

6.5 Une minuterie

Javascript met à votre disposition une minuterie (ou plus précisément un compteur à rebours) qui permettra de

déclencher une fonction après un laps de temps déterminé.

La syntaxe de mise en route du temporisateur est :

```
nom_du_compteur = setTimeout("fonction_appelée()", temps en milliseconde)
```

Ainsi, `setTimeout("demarrer()",5000)` va lancer la fonction `demarrer()` après 5 secondes.

Pour arrêter le temporisateur avant l'expiration du délai fixé, il y a :

```
clearTimeout(nom_du_compteur)
```

6.6 L'emploi de this

Pour désigner l'objet en cours, Javascript met à votre disposition le mot-clé `this`. Cette écriture raccourcie est souvent utilisée (sans risque de confusion) en remplacement du chemin complet de l'objet dans un formulaire. Un exemple vous éclairera mieux qu'un long discours.

Soit un script avec un formulaire :

```
<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="choixprop(form3)">
</FORM>
```

Au lieu d'employer `choixprop(form3)`, on aurait pu utiliser `choixprop(this.form)` et éviter ainsi toute confusion avec les autres noms de formulaires. Dans cet exemple, `this.form` désigne le formulaire `form3` complet. Par contre, `choixprop(this)` n'aurait désigné que l'élément de type bouton du formulaire `form3`.

Pour être complet, `this` est utilisé aussi pour créer une ou plusieurs propriétés d'un objet. Ainsi, pour créer un objet livre avec les propriétés auteur, éditeur et prix cette opération peut être effectuée à l'aide de la fonction :

```
function livre(auteur, editeur, prix) {
this.auteur = auteur;
this.editeur = editeur;
this. prix = prix;
}
```