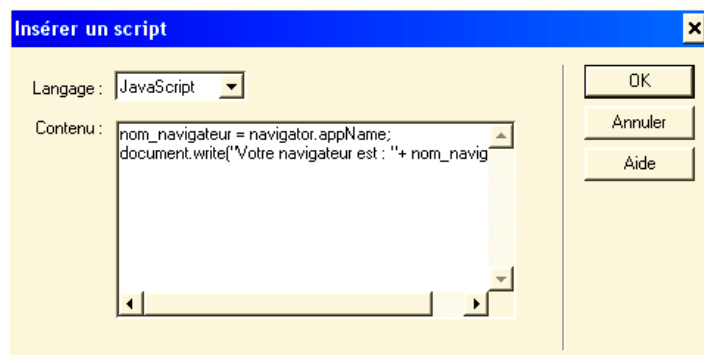


JavaScript - Présentation

Qu'est-ce que JavaScript ?

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

- Les instructions JavaScript se placent donc à l'intérieur du code HTML
- Les scripts seront donc exécutés (interprétés) par le navigateur de l'utilisateur. Ceci sans faire appel à des ressources du côté du serveur et sans avoir à attendre de téléchargement supplémentaire.
- JavaScript n'est pas Java. Bien que ressemblants, ces deux noms désignent des technologies très différentes.
- JavaScript est un langage objet et événementiel. Le développeur peut créer des objets sur la page, avec des propriétés et des méthodes et leur associer des actions en fonction d'événements déclenchés par le visiteur (passage de souris, clic, saisie clavier, etc...).
- Un éditeur Web comme DreamWeaver intègre la possibilité d'insérer du code JavaScript :



```
<script language="JavaScript">
nom_navigateur = navigator.appName;
document.write("Votre navigateur est : "+ nom_navigateur);
</script>
```



Il y a plusieurs endroits dans une page web où il est possible d'intégrer du code JavaScript :

- dans le corps de la page
- dans l'entête de page
- dans un événement d'un objet de la page
- dans un fichier externe inclus lors de la lecture de la page

La structure classique d'une page HTML contenant du JavaScript déclaré **dans l'entête** est la suivante :

```
<HTML>
<HEAD>
<TITLE>Titre de page</TITLE>
<SCRIPT language="JavaScript">
<!-- // Cache ce qui suit aux navigateurs qui ne supportent pas
JavaScript
```

Le code JavaScript est écrit ici
//--> // Fin de la partie cachée

```
</SCRIPT>
</HEAD>
```

```
<BODY>
```

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

Événementuellement appelé là

Fonction JavaScript appelée dans un lien :

```
<A HREF="javascript:ma_fonction()">Cliquez ici</A>
```

Fonction JavaScript appelée lors d'un événement (ici sur un lien) :

```
<A HREF="www.lien.htm" onMouseOver="ma_fonction()">lien</A>
```

JavaScript dans un fichier externe :

```
<script language=javascript1.2
src="loto.js"></script>
```

JavaScript - Présentation

Un premier script

```
<HTML>
<HEAD>
<TITLE>Mon premier Javascript</TITLE>
</HEAD>
<BODY>
<H1>Ceci est du Html</H1>
<SCRIPT LANGUAGE="Javascript">
<!--
alert("votre texte");
document.write("<H1>Et ceci du Javascript</H1>");
//-->
</SCRIPT>
<H1>Ceci est encore du Html</H1>
</BODY>
</HTML>
```

Javascript est sensible à la casse. 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 ne peuvent être employés que dans les chaînes de caractères comme dans le texte de notre exemple.

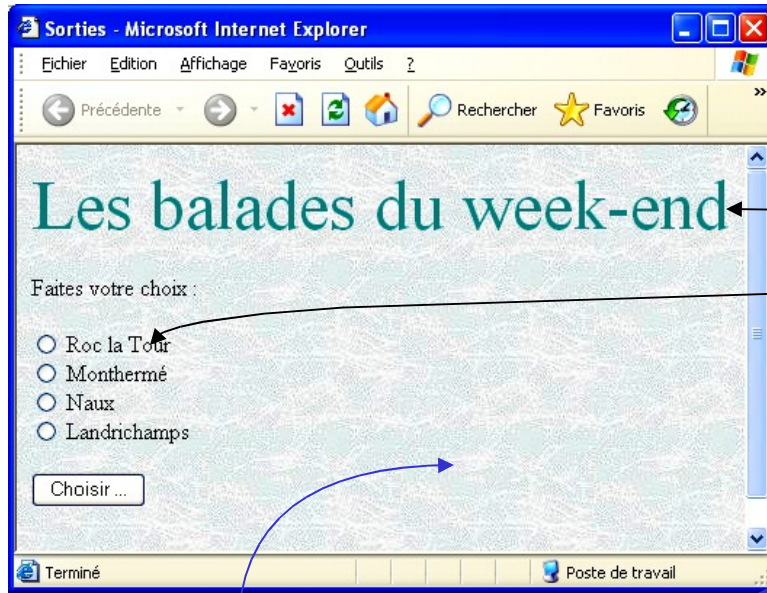
Les guillemets " et l'apostrophe ' font partie intégrante du langage Javascript. On peut utiliser l'une ou l'autre forme à condition de ne pas les mélanger. Ainsi `alert("blabla")` donnera un message d'erreur. Si vous souhaitez utiliser des guillemets dans vos chaînes de caractères, tapez `\"` ou `\'`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

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 scripts 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>
```

JavaScript - Présentation

Objets JavaScript



JavaScript nous permet d'accéder au contenu d'une page Web en hiérarchisant ses composants.

Voici un exemple de page telle qu'affichée par votre navigateur. Celle-ci comporte un titre et un formulaire.

JavaScript va donc diviser cette page en objets et surtout va vous permettre d'accéder à ces objets, d'en retirer des informations et de les manipuler.

L'objet le plus haut dans la hiérarchie est la fenêtre, dans laquelle s'affiche la page : [window](#)

L'objet [document](#) est la page HTML s'affichant à l'intérieur de la fenêtre. Il contient des informations concernant le document courant (par le biais de tableaux et de propriétés). C'est l'un des objets les plus utilisés (avec [window](#)) en JavaScript et permet, par exemple, d'écrire dans le document html, on appelle cela : écrire dans le flux d'un document.

Ce document contient un formulaire que JavaScript nomme [form](#). Ce formulaire contenant lui-même des boutons radio et un bouton d'envoi.

Pour accéder à un objet, il faudra donner le chemin complet de l'objet en allant du contenant le plus extérieur à l'objet à l'objet référencé. Il est toutefois optionnel de mentionner l'objet [window](#). On utilise le point pour passer d'un contenant à un contenu.

Exemple de notation pour atteindre le bouton "Choisir" du formulaire ci-dessus :

[window.document.form.button](#)

optionnel

JavaScript - Présentation

Propriétés et méthodes des objets

Un objet JavaScript est constitué de ses propriétés (**sa description, ses caractéristiques**) et de méthodes (**ce qu'il sait faire**). Les propriétés et méthodes d'un objet peuvent être appelées, que l'on souhaite lire ou modifier l'une de ses propriétés ou bien exécuter l'une de ses méthodes, par la même notation à point :

objet.propriété
objet.méthode (argument)

❑ Exemples de propriétés :

document.backgroundColor : la couleur de fond de l'objet document courant

navigator.appName : le nom du navigateur Internet sur la machine du visiteur

```
<script language="javascript">
document.backgroundColor="#A0A0A0";
var nom_navigateur = navigator.appName
</script>
```

❑ Exemples de méthodes :

history.back() : demande à l'objet history le retour à la page vue précédemment

document.write() : écrit dans l'objet document, soit du texte, soit des balises HTML

```
<script language="javascript">
<a href="javascript:history.back();"><b>Retournez d'où vous
venez</b></a>
</script>
```

JavaScript - Présentation

document

L'objet **document** se réfère au contenu affiché dans la fenêtre du navigateur.
Quelques propriétés de l'objet document :

lastModified : la date de dernière modification du document

linkcolor : la couleur des liens

location : l'adresse de la page en cours (l'URL)

referrer : l'adresse d'où l'on vient

title : le titre du document

Quelques méthodes de l'objet document :

write() : écrit du texte ou de l'HTML dans le document

writeln() : idem, mais suivi d'un retour-chariot


close() : fermeture du document en cours



❑ Exemple :

```
<script language="javascript">
document.write("L'URL courante est : " + document.location + "<br> et vous venez de : " +
document.referrer+"")
</script>
```

JavaScript - Présentation

 if ... then ... else

JavaScript permet de faire dépendre l'exécution du code d'une condition. Ceci s'exprime à l'aide des mots-clés **if** et **else**, et la structure d'une condition est la suivante :

```
if ( condition )  
    { faire ceci; }  
else  
    { faire cela; }
```

Pour demander si :

- Opérateurs de comparaison :
- deux valeurs sont égales ==
 - deux valeurs sont différentes !=
 - si une valeur est plus grande ou égale qu'une autre >=
 - si une valeur est plus grande dans tous les cas qu'une autre >
 - une valeur est plus petite ou égale qu'une autre <=
 - une valeur est plus petite dans tous les cas qu'une autre <

Opérateurs logiques :

Avec **&&** vous reliez deux ou plusieurs conditions par "et"
Avec **||** vous reliez deux ou plusieurs conditions par le "ou" inclusif

❑ Exemple : test du contenu de **referrer** avant de l'afficher :

```
<script language="Javascript">  
if (document.referrer != "") {document.write ("Vous venez de : " + document.referrer)}  
else {document.write ("Vous avez trouvé le chemin tout seul !")}  
</script>
```

JavaScript - Présentation

Variables

Les variables sont des cases mémoire dans lesquels vous pouvez ranger les données dont vous avez besoin au cours de votre programme.

Vous pouvez à tout moment modifier la valeur d'une variable. Pour pouvoir travailler avec des variables, vous devez d'abord les définir.

Le nom de la variable pourra ensuite être employé pour manipuler la valeur qu'elle contient.

Exemples :

☐ `var nombre = 42;`

Ne confondez, sous peine d'erreurs, pas l'opérateur d'affectation `=` avec l'opérateur de comparaison `==`

Notez qu'il n'y a pas de phase de déclaration de la variable, et le mot "nombre" désigne maintenant un emplacement mémoire auquel on accèdera par ce même nom.

☐ `var carre = nombre * nombre;`

Le résultat de l'opération est rangé dans une nouvelle variable

☐ `var phrase = "Le carré de ";`

Alors que les variables nombre et carre sont numériques, phrase contient une chaîne de caractères

☐ `document.write (phrase + nombre + " est " + carre);`

La méthode write accepte des arguments de types différents

- ☐ Un nom de variable doit commencer par une lettre (majuscule ou minuscule) ou un "_"
- ☐ Un nom de variable peut comporter des lettres, des chiffres et les caractères _ et & (les espaces ne sont pas autorisés!)
- ☐ Les noms de variables ne peuvent pas être un des noms réservés par JavaScript (if, then, var, ...)
- ☐ Les noms de variables sont sensibles à la casse (**Total** et **total** sont deux variables différentes)

JavaScript - Présentation

Fonctions

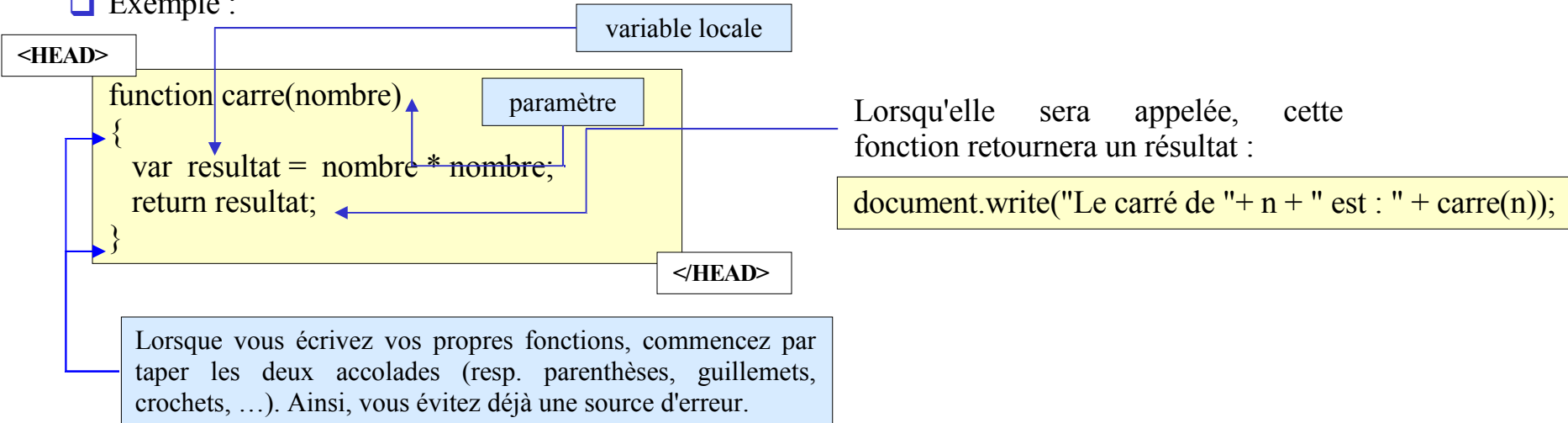
Une fonction est un ensemble d'instructions que l'on peut appeler à l'aide de son nom. Notez que le code JavaScript qui ne se trouve pas dans une fonction sera exécuté lors de la lecture du fichier HTML tandis qu'une fonction ne s'exécutera pas tant que l'on n'y aura pas fait appel quelque part dans la page !

L'utilité d'une fonction : la structuration du code, la réutilisation, l'économie.

Une fonction accepte des paramètres (définis au moment de l'appel) et peut, ce n'est pas obligatoire, renvoyer un résultat.

Le navigateur interprète la page de haut en bas. Aussi, on déclare généralement les fonctions dans l'en-tête (<HEAD> ... </HEAD>) afin qu'elles puissent être appelées de n'importe où dans la page.

Exemple :



- ❑ Les noms de fonctions suivent les mêmes règles que les noms de variables
- ❑ L'appel d'une fonction doit toujours comporter les parenthèses, même si la fonction ne prend pas d'argument

JavaScript - Présentation

Variables globales ou locales à une fonction

```
<SCRIPT language="Javascript">
```

```
<!--
```

```
var a = 12;
```

```
var b = 4;
```

```
function Double(b)
```

```
{
```

```
  var a = b * 2;
```

```
  return a;
```

```
}
```

```
document.write  
("Le double de " + b + " est " + Double(b));
```

```
document.write  
("La valeur de a est ",a);
```

```
// -->
```

```
</SCRIPT>
```

- ❑ Une variable déclarée au début du script, c'est-à-dire avant toute fonction sera globale, on pourra alors l'utiliser à partir de n'importe quel endroit dans le script
- ❑ Une variable déclarée par le mot clé *var* dans une fonction aura une portée limitée à cette seule fonction, c'est-à-dire qu'elle est inutilisable ailleurs, on parle alors de variable locale

Durée de vie des variables a et b déclarées avec *var* dans la fonction

Durée de vie des variables a et b globales, toutefois cachées dans la fonction par les variables locales homonymes

- ❑ Prenez la précaution d'utiliser systématiquement le mot *var*, car sans cela, une variable déclarée à l'intérieur d'une fonction devient globale, source d'erreur possible ultérieurement difficile à déceler

JavaScript - Présentation

Évènements

En Html classique, il y a un événement bien connu. C'est le clic de la souris sur un lien. JavaScript ajoute d'autres événements et permet de les intercepter.

Comme, par exemple :

Lorsque l'utilisateur clique sur un bouton, un lien ou tout autre élément

Lorsque la page est chargée par le browser ou le navigateur.

Lorsque l'utilisateur quitte la page.

Lorsque l'utilisateur place le pointeur de la souris sur un lien ou tout autre élément.

Lorsque le pointeur de la souris quitte un lien ou tout autre élément.

Attention : Javascript 1.1 (donc pas sous MSIE 3.0 et Netscape 2)

Exemple :

```
<HTML>
<HEAD>
<SCRIPT LANGUAGE='Javascript'>
function bienvenue() {
  alert("Bienvenue à cette page");
}
function au_revoir() {
  alert("Au revoir");
}
</SCRIPT>
</HEAD>
<BODY onLoad='bienvenue()' onUnload='au_revoir()'>
</BODY>
</HTML>
```

Un gestionnaire d'évènement est une fonction qui est prévenue d'un évènement donné et qui appelle alors le code écrit pour se dérouler à ce moment-là, en réponse à l'évènement.

onClick

onLoad

onUnload

onMouseOver

onMouseOut

on : au moment où ...

Les événements onLoad et onUnload sont utilisés sous forme d'attributs de la balise <BODY>

"alert" est une méthode de l'objet **window**

JavaScript - Présentation

onmouseover et onmouseout

Le code du gestionnaire d'événement onmouseover s'ajoute aux balises de lien :

```
<A HREF="#" onmouseover="action()">lien</A>
```

La fonction `action()` est appelée lorsque l'utilisateur passe avec sa souris sur le lien. L'attribut `HREF` est indispensable. Il peut contenir l'adresse d'une page Web si vous souhaitez que le lien soit actif ou simplement `"#"` si aucun lien actif n'est prévu..

❑ Exemple :

```
<HTML>
<HEAD>
<SCRIPT language="Javascript">
function message(){
alert("Coucou")
}
</SCRIPT>
</HEAD>
<BODY>
<A HREF="" onmouseover="message()">message important</A>
</BODY>
</HTML>
```

Code du gestionnaire de l'évènement dans une fonction

Code du gestionnaire de l'évènement dans l'évènement

La syntaxe du gestionnaire onmouseout est la suivante :

```
<A HREF="#" onmouseout="action()">message important</A>
```

Exemple :

Les deux en un :

```
<A HREF="" onmouseover="actionOver()"
onmouseout="actionOut()">message important</A>
```

```
<html>
<head>
<title>Test</title>
</head>
<body>
<a href="nouvelles.htm"
onMouseout="alert('Vous devriez aller voir ce qu'il y a de
neuf')"><b>Quoi de neuf?</b></a>
</body>
</html>
```

JavaScript - Présentation

window

C'est l'objet du Javascript (implicite) le plus élevé dans la hiérarchie. Il peut être défini comme une véritable fenêtre, où l'objet document est considéré comme le contenu de cette même fenêtre. L'utilisation la plus courante est la création d'une fenêtre volante (popup).

Quelques propriétés de l'objet window :

defaultstatus : message par défaut dans la barre d'état du navigateur

status : lit ou écrit dans la barre d'état du navigateur

document : le contenu de la fenêtre (objet lui aussi)

history : l'historique de la navigation pour cette fenêtre (objet aussi)

Quelques méthodes de l'objet window :

alert('message') : affiche un avertissement dans une boîte de dialogue

back() : va à l'adresse précédente de l'historique

forward() : va à l'adresse précédente de l'historique

close() : fermeture de la fenêtre

open(arguments) : ouverture d'une nouvelle fenêtre

Open est la méthode qui permettra de créer une nouvelle fenêtre, par exemple en répondant au clic sur un lien, un bouton, ou pour répondre à un événement donné. Les arguments de la méthode open sont : le nom donné à la fenêtre, l'URL à afficher dans la fenêtre, et tous les paramètres de configuration de ladite fenêtre.



JavaScript - Présentation

Création d'une nouvelle fenêtre

Syntaxe : `window.open('URL','nom_fenêtre',options)`

Le site ou le fichier à afficher dans la nouvelle fenêtre

Un nom pour que la nouvelle fenêtre puisse éventuellement servir de cible pour d'autres liens

Il est possible d'utiliser cette méthode avec n'importe quel gestionnaire d'événement, directement dans le code à exécuter ou bien dans une fonction.

Les options doivent être saisies les unes après les autres, séparées par des virgules, sans espace.

L'ensemble des options doit être encadré par les guillemets.

option	description
<code>directories = yes/no</code>	Affiche ou non les boutons de navigation
<code>location = yes/no</code>	Affiche ou non la barre d'adresse
<code>menubar = yes/no</code>	Affiche ou non la barre de menu (fichier, édition, ...)
<code>resizable = yes/no</code>	Définit si la taille de la fenêtre est modifiable ou non
<code>scrollbars = yes/no</code>	Affiche ou non les ascenseurs (barres de défilement)
<code>status = yes/no</code>	Affiche ou non la barre d'état
<code>toolbar = yes/no</code>	Affiche ou non la barre d'outils
<code>width = largeur (en pixels)</code>	Définit la largeur
<code>height = hauteur (en pixels)</code>	Définit la hauteur

Ces options correspondent aussi à des propriétés de l'objet window.

Exemple :

Syntaxe du gestionnaire d'évènement onClick

```
<A HREF="#" onclick="window.open('http://www.bnf.fr','FenBNF','width=800,height=600,left=0,top=0')">La grande bibliothèque</A>
```

Création, dans le onClick, d'une nouvelle fenêtre avec options

- L'adresse à afficher dans la nouvelle fenêtre : 'http://www.bnf.fr'
- Un nom pour que la nouvelle fenêtre puisse éventuellement recevoir d'autres liens, comme par exemple :

```
<a href='http://expositions.bnf.fr/savoirs/index.htm' target='FenBNF'>
```

- Les options, entre guillemets, séparées par des virgules : 'width=800,height=600,left=0,top=0'

JavaScript - Présentation

Exemples de commandes de fenêtres

● Méthode moveTo() : déplacement de la fenêtre

Ici, le script déclare une variable de type window ("Fen"), afin de pouvoir la manipuler ensuite dans la fonction "bouge"

L'objet **screen** est l'écran de l'ordinateur de l'utilisateur, grâce à quoi vous pouvez connaître la résolution disponible, le nombre de couleurs, ...

```
<html><head>
<script type="text/javascript">
Fen = window.open("fichier.htm","secondefenetre","width=200,height=200");
function bouge() {
  Fen.moveTo(screen.width-200,screen.height-200); Fen.focus();}

</script>
</head><body>
<a href="javascript : bouge()">Ranger la fenêtre</a>
</body></html>
```

● Méthode close() : un lien qui ferme la fenêtre

```
<A HREF="#" ONCLICK="window.close()">FERMER</A>
```

● Propriété defaultstatus : sauvegarde le contenu de la barre d'état qui est affiché aussi longtemps qu'aucun événement particulier n'intervient (par exemple passer sur un lien avec la souris).

```
window.defaultStatus = "Page d'accueil de Gabriel";
```

JavaScript - Présentation

Boucle for

La boucle for comporte son propre compteur.

Sa syntaxe se compose de trois parties :

for (initialisation du compteur ; condition d'entrée ; évolution du compteur)

❑ Exemple :

```
var xsum = 0;  
var x;
```

```
for ( x = 1; x <= 10; x++ )
```

```
{  
  xsum += x;  
}
```

Initialisation de la "variable de boucle"

Condition pour entrer dans la boucle

Evolution de la variable de boucle à chaque tour

Les parties Initialisation et Evolution d'une boucle for peuvent contenir plusieurs instructions :

```
var xsum = 0;  
for ( var x = 1, lcnt = 0 ; lcnt < 100 ; x++, lcnt++ )  
{  
  ...  
}
```


JavaScript - Présentation

Boucle while

La boucle while est indiquée quand le nombre de boucles à exécuter est inconnu (peut-être zéro ...).

Syntaxe :

while (condition d'entrée)

```
var saisie = "";  
var compteur = 3;
```

```
while(saisie != "1515" && compteur>0)  
{  
    saisie = window.prompt("Mot de passe ?", "");  
    compteur--;  
}
```

La condition est évaluée avant d'entrer dans la boucle.
Elle doit donc être initialisée (signifiante).

```
    saisie = window.prompt("Mot de passe ?", "");  
    compteur--;
```

La ou les conditions de sortie doivent évoluer dans la boucle, faute de quoi celle-ci bloquera la fenêtre du navigateur dans une boucle infinie.

```
if(saisie == "1515")  
    document.write("<b>OK ...</b>");  
else  
    document.write("<b>Accès refusé ...</b>");
```

Puisque l'exécution de la boucle était soumise à plusieurs conditions, il faut tester ici pour savoir pour quelle raison on est sorti de la boucle ...

Il existe aussi une boucle do while. La différence entre les deux réside en ce que, pour la boucle while normale la condition de la boucle est vérifiée **avant** l'exécution du code tandis que pour la boucle do while le code est **d'abord** exécuté et qu'après seulement la condition de la boucle est vérifiée. De cette façon il vous est possible d'imposer que les instructions de la boucle soient exécutées dans tous les cas au moins une fois, même quand la condition de la boucle s'avère fausse dès le départ.

JavaScript - Présentation

Encore une manipulation de fenêtre

```
<html><head><title>Test</title>
<script type="text/javascript">
Fen = window.open("", "FenCompteur", "width=200,height=200,left=0,top=0");
focus();
function MontreCompteur(n) {
  Fen.document.open();
  if (n==0)
    Fen.document.write("<B>Terminé.</B>");
  else
    Fen.document.write("<B>Plus que " + n + " fois</B>");
  Fen.document.close();
}
</script>
</head>

<body>
<script type="javascript">
var saisie = "";
var compteur = 3;
while(saisie != "1515" && compteur>0) {
  MontreCompteur(compteur);
  saisie = window.prompt("Mot de passe ?", "");
  compteur--;
}
if(saisie == "1515")
  document.write("<b>OK ...</b>");
else
  document.write("<b>Accès refusé ...</big>");
Fen.close();
</script>
</body></html>
```

Le document dans la fenêtre est libéré pour l'écriture

Un nouveau contenu est écrit ici ...

Puis la modification est entérinée avec close()

JavaScript - Présentation

Les commentaires

La mise au point de code source (quel que soit l'environnement de développement Delphi, C++, JavaScript, ...), et surtout la maintenance et le réemploi du code, nécessitent que ce code soit abondamment commenté.

Les lignes de commentaire ne jouent aucun rôle lors de l'exécution du code, elles sont simplement ignorées. D'ailleurs une utilisation marginale des commentaires peut consister à désactiver provisoirement une ou plusieurs instructions. Ou encore à mentionner des droits dans votre code JavaScript ...

Commentaires à la C++ (commentaire de fin de ligne) //

```
focus(); // Rend le focus à la fenêtre principale
```

Ne confondez pas les commentaires Javascript et les commentaires Html :

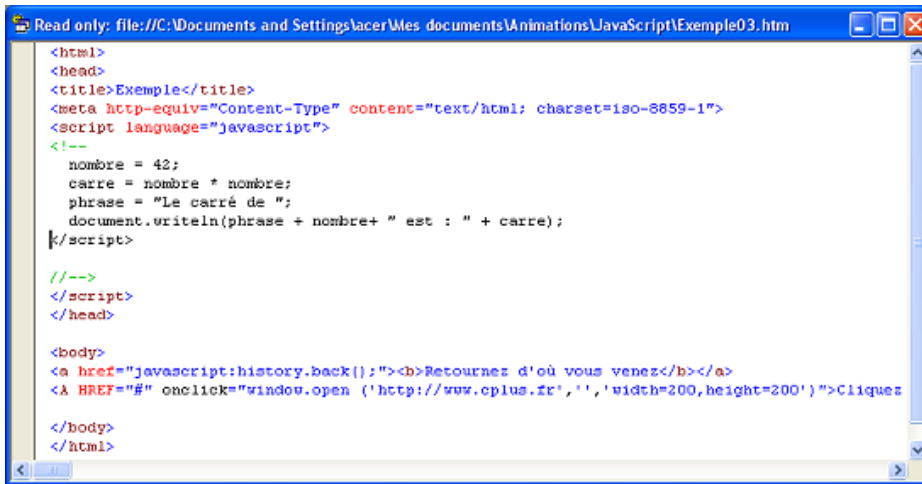
Pour rappel :

```
<!--  
-->
```

Commentaires à la C (plusieurs lignes possibles) /* */

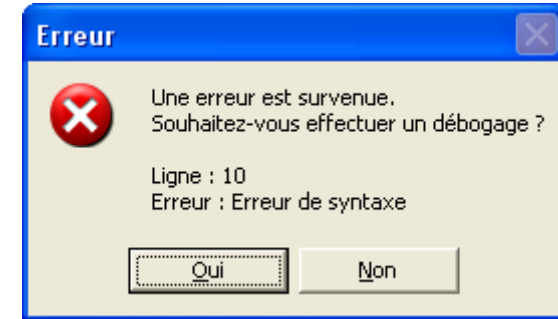
```
Fen = window.open("", "FenCompteur", "width=200,height=200,left=0,top=0");  
/* il est important de rendre le focus à la fenêtre principale  
faute de quoi le curseur reste dans la petite fenêtre (Fen) et  
l'utilisateur ne comprendra pas pourquoi sa réponse n'apparaît pas  
dans la fenêtre de saisie ... */  
focus();
```

Pour debugger les scripts, plusieurs solutions existent. Comme par exemple, la solution proposée par DreamWeaver et qui passe par la machine virtuelle Java. La méthode la plus répandue semble être le debugger de scripts de Microsoft, téléchargeable sur <http://www.microsoft.com>. Cette solution s'intègre en effet à Internet Explorer :

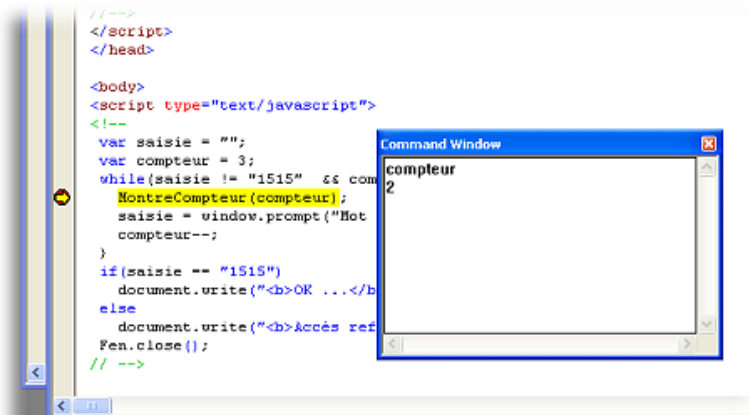


```
<html>
<head>
<title>Exemple</title>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
<script language="javascript">
<!--
nombre = 42;
carre = nombre * nombre;
phrase = "Le carré de ";
document.writeln(phrase + nombre + " est : " + carre);
//-->
</script>
</head>

<body>
<a href="javascript:history.back();"><b>Retournez d'où vous venez</b></a>
<A HREF="#" onclick="window.open ('http://www.eplus.fr','','width=200,height=200')">Cliquez
</body>
</html>
```



Le debugger de scripts permet de poser des points d'arrêt dans le code, d'exécuter le code pas à pas, de suivre ou de modifier la valeur des variables du script.



JavaScript - Présentation

Quelques adresses

<http://fr.selfhtml.org/>

Une référence en matière d'autoformation à JavaScript, mais aussi à HTML, DHTML, CGI, ...

<http://www.toutjavascript.com/>

Tutoriels et téléchargement de scripts ...

<http://www.allhtml.com/javascript/index.php>

Nombreux scripts et liens ...

<http://www.dynamicdrive.com/>

Scripts très intéressants et pédagogiques but in English