

# Le langage Javascript

Les documents web structurés et présentés sont insuffisants aujourd'hui pour rendre vivant les contenus. Il faut souvent les rendre dynamiques grâce au langage PHP et adapter leur contenu aux utilisateurs. Les traitements effectués par le langage PHP permettent d'interagir avec les bases de données et nécessitent souvent divers contrôles et enrichissements que peut effectuer localement (à partir du navigateur) le client. C'est le rôle du langage Javascript qui, rattaché au navigateur, peut réaliser certains traitements et contrôles ainsi qu'assurer des présentations enrichies.

## Sommaire

A.	Les bases du langage Javascript.....	3
A.1.	Déclaration et implémentation des scripts .....	3
A.1.1.	Implémentation interne d'un script.....	3
A.1.2.	Script inscrit dans un événement.....	4
A.1.3.	Script externe .....	4
A.2.	Débogage de Javascript dans le navigateur.....	5
A.3.	Interaction du Javascript dans les balises.....	6
A.3.1.	Contenus avec document.write .....	6
A.3.2.	Contenus avec innerHTML .....	7
B.	Programmation en Javascript .....	9
B.1.	Entrées / sorties, utilisation de variables .....	9
B.1.1.	Affectation .....	9
B.1.2.	Calculs .....	10
B.1.3.	Déclarations des variables.....	11
B.2.	Structures algorithmiques usuelles .....	12
B.2.1.	Les conditions.....	12
B.2.2.	Les structures itératives .....	15
B.3.	La gestion d'événements.....	16
B.3.1.	Les événements .....	16
B.3.2.	Insertion d'un bouton d'action avec un événement.....	17
B.3.3.	Événements à l'ouverture de la page.....	17
B.3.4.	Lien hypertexte .....	18
B.3.5.	Événements liés à la souris .....	18
B.3.6.	Fonctions avec passage de paramètre .....	19
B.3.7.	Commandes JavaScript « inline ».....	20
C.	Utilisation des objets fenêtres avec JavaScript.....	21
C.1.	Les fenêtres pop-up.....	21
C.2.	Créer une fenêtre à la volée .....	23
D.	Fonctions et objets usuels de Javascript .....	25
D.1.	Déviations de pages .....	25
D.2.	Afficher ou utiliser les paramètres liés au navigateur.....	25
D.3.	Affichage dynamique de la date de la machine cliente .....	26
D.4.	Navigation et menus avec JavaScript .....	27
D.4.1.	Menus déroulant .....	27

D.4.2.	Images sensibles .....	28
E.	Les formulaires.....	30
E.1.	L'accès aux formulaires .....	30
E.1.1.	Principe de l'accès aux éléments de formulaire .....	30
E.1.2.	Accéder au formulaire .....	30
E.1.3.	Accéder à un élément .....	31
E.1.4.	Manipuler les propriétés d'un élément.....	31
E.1.5.	Appeler une méthode sur un élément .....	31
E.1.6.	Modifier un élément sur un événement .....	32
E.1.7.	Désigner l'objet par « this » .....	32
E.2.	L'accès aux éléments de formulaire .....	33
E.2.1.	L'accès aux éléments de type INPUT .....	33
E.2.2.	Exemple complet .....	38
F.	Le DHTML .....	41
F.1.	Qu'est-ce que le DHTML ? .....	41
F.2.	Les cookies .....	41
F.3.	Gestion dynamique de la présentation des sites .....	44
F.3.1.	Les objets du DHTML .....	44
F.3.2.	Manipuler les calques avec JavaScript.....	44
F.3.3.	Manipulation sur le style .....	45
F.3.4.	Exemple DHTML sur un formulaire .....	45

## A. Les bases du langage Javascript

Il est possible d'augmenter les performances de XHTML, en ajoutant des fonctionnalités de présentation, d'interactivité et de traitement directement à partir du navigateur. Ceci en utilisant au sein des documents XHTML des langages spécifiques tels que JavaScript (JS), ou VBScript (VBS).

### A.1. Déclaration et implémentation des scripts

L'utilisation d'un script dans un fichier XHTML peut se faire de trois manières.

#### A.1.1. Implémentation interne d'un script

Le code Javascript est inscrit dans l'entête **<head></head>** du document. Cela permet de centraliser tous les éléments dans un seul support. Le risque est cependant d'alourdir un document et de ne pas bénéficier de toutes les fonctionnalités d'un environnement de développement où la partie HTML et Javascript seraient traités indépendamment.

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
  <title>Super document</title>
  <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
  <link rel="stylesheet" type="text/css" media="screen" href="css/style.css" />
  <script type="text/javascript">
    document.write('<p>Bonjour à tous, je suis un script <b>Javascript</b></p>');
  </script>
</head>
<body>
  <p>Nous allons continuer ici...</p>
</body>
</html>
```

NB : La déclaration sera text/vbscript pour du VBScript.

Bonjour à tous, je suis un script **Javascript**

Nous allons continuer ici..

L'intérêt de cette manipulation peut sembler obscure, et même décevant. Mais on vient de montrer qu'un programme écrit en JavaScript peut interférer dans une page web. Or, les pages HTML ne sont pas des programmes, il n'y a pas de réactivité et le HTML ne peut afficher rien d'autre que ce qui est inscrit entre les balises. L'ajout de commandes va donc permettre de multiplier les possibilités et d'intégrer des instructions de programmation : déclaration de variables, structures conditionnelles, structures de contrôle, etc.

### A.1.2. Script inscrit dans un événement

Un script Javascript peut être associé à des événements inscrits dans les attributs d'une balise. Ces événements peuvent être le chargement de la page, le clic de la souris, etc. ils seront plus amplement traités par la suite.

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
  <title>Super document</title>
  <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
  <link rel="stylesheet" type="text/css" media="screen" href="css/style.css" />
</head>
<body onload="document.write('<p>Bonjour à tous, je suis un script <b>Javascript</b></p>');">
  <p>Nous allons continuer ici...</p>
</body>
</html>
```

Bonjour à tous, je suis un script **Javascript**

Remarque : le script Javascript s'exécute sans discontinuer et ne laisse pas apparaître le texte HTML situé dans les balises `<p></p>`. Il aurait fallu compléter le code par l'instruction « `break;` » qui aurait immédiatement cessé le script et donc... on aurait pas eu le temps de voir la phrase. Donc, cette façon de procéder est adaptée pour certains cas liés à des événements que nous verrons ultérieurement.

### A.1.3. Script externe

La normalisation XHTML suggère l'insertion de scripts externes placés dans un fichier à part (portant l'extension .js) et déclarés dans l'en-tête.

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
  <title>Super document</title>
  <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
  <link rel="stylesheet" type="text/css" media="screen" href="css/style.css" />
  <script type="text/javascript" src="scripts/script.js"></script>
</head>
<body>
  <p>Nous allons continuer ici...</p>
</body>
</html>
```

Le script Javascript « `script.js` » situé dans le dossier « `scripts` » :

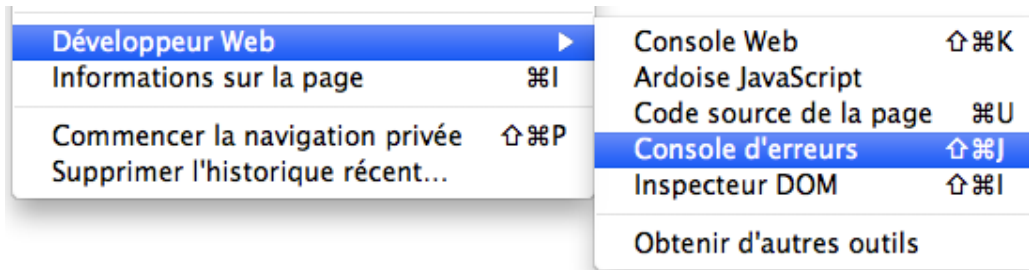
```
document.write('<p>Bonjour à tous, je suis un script <b>Javascript</b></p>');
```

Bonjour à tous, je suis un script **Javascript**

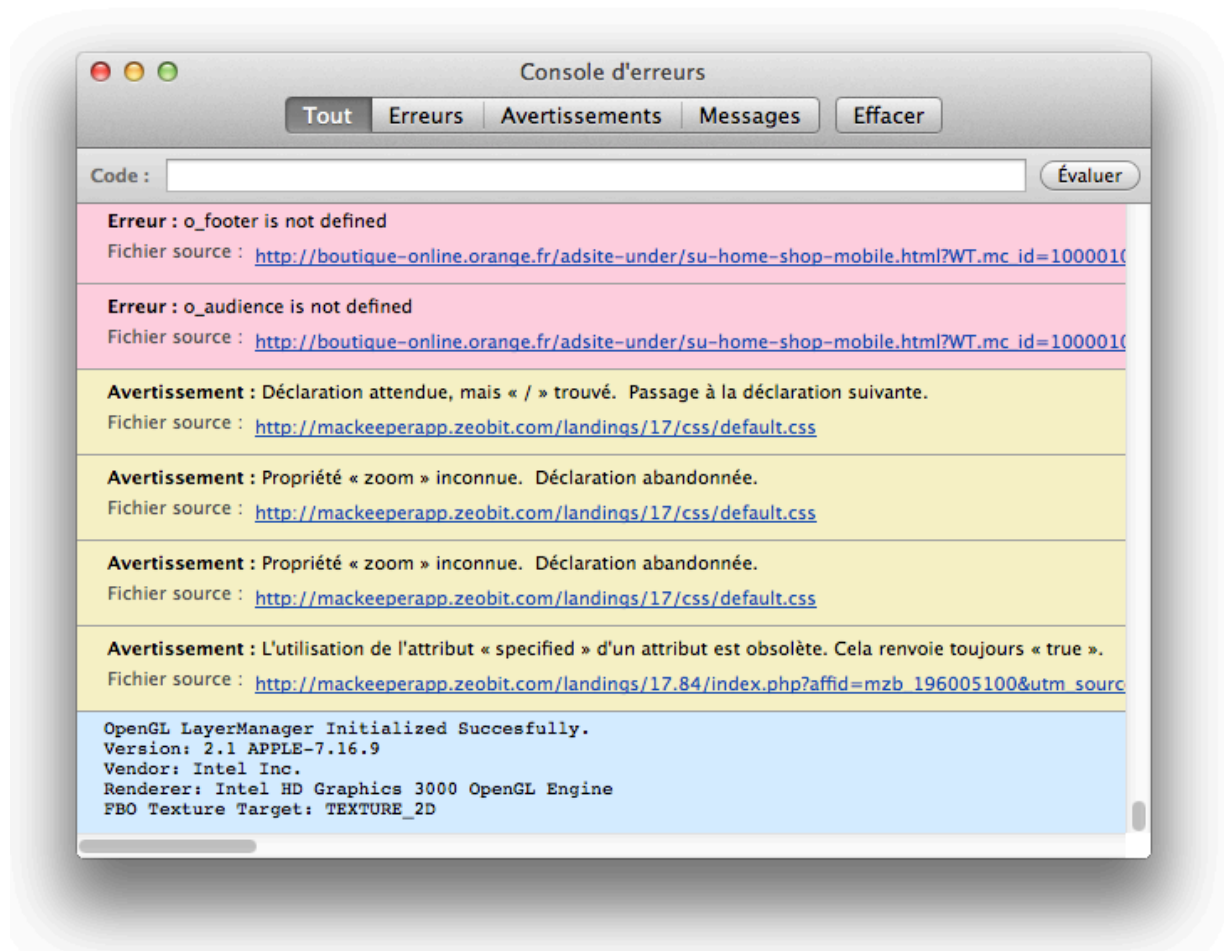
Nous allons continuer ici...

## A.2. Débogage de Javascript dans le navigateur

De nombreux navigateurs, comme Firefox, proposent des outils pour développeur.



La console d'erreurs de Firefox est capable d'indiquer les avertissements ou erreurs liées au Javascript intégré à vos pages ainsi qu'aux erreurs dans les styles CSS.



## A.3. Interaction du Javascript dans les balises

Nous l'avons vu, le Javascript interagit directement avec le document web auquel il se rattache. Il peut donc en écrire totalement le contenu ou une partie ciblée.

### A.3.1. Contenus avec document.write

C'est l'objet **document** et sa méthode **write** qui permettent de le faire. Cela permet d'écrire n'importe quelle instruction HTML et même de faire appel aux styles déclarés dans la feuille de style.

```
<head>
  <title>Super document</title>
  <meta http-equiv="Content-Type" content="text/html; charset=UTF-8"/>
  <link rel="stylesheet" type="text/css" media="screen" href="css/style.css" />
  <script type="text/javascript" src="scripts/script.js"></script>
</head>
<body>
  <p>
    Nous allons continuer ici...
  </p>
</body>
</html>
```

```
document.write('<p>Tout s\'écrit ici en <b>Javascript</b></p>');
document.write('<p class="beau">Et même avec du CSS</b></p>');
```

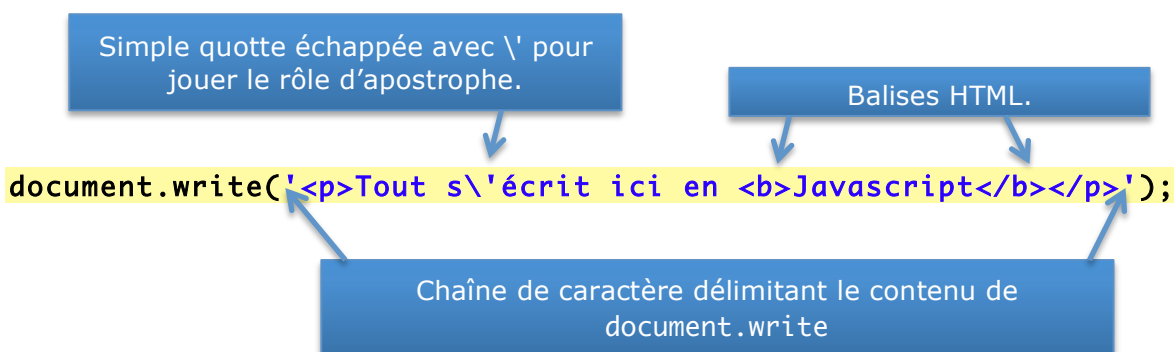
```
p.beau {
  color: #000099;
  font-family: verdana;
}
```

Tout s'écrit ici en **Javascript**

Et même avec du CSS

Nous allons continuer ici...

Remarque : le texte du script est encadré par de simples quotes alors que celui d'HTML est encadré par des guillemets. Nous avons donc dû « échapper » l'apostrophe de « s'écrit » avec `\'` pour éviter que l'interpréteur Javascript ne croit que la chaîne de caractère s'arrête.



## A.3.2. Contenus avec innerHTML

### A.3.2.1. Affectation d'un contenu statique

Avec le langage Javascript on peut affecter n'importe quelle balise avec un contenu adapté. Il faut que les balises soient identifiées par un attribut (unique) *id* ou *name* (pour les champs de formulaire).

```
<body>
  <p id="insertion"></p>
  <p>
    Vous pouvez voir le paragraphe mystère en cliquant <a
href="javascript:ecriture()">ici</a>.
  </p>
</body>
```

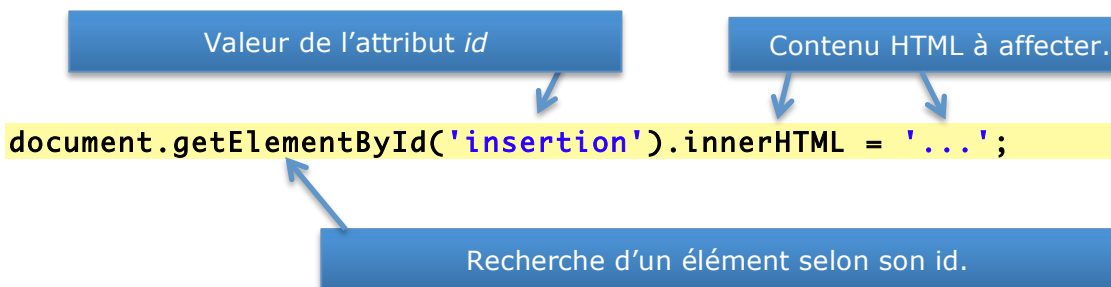
```
function ecriture() {
  document.getElementById('insertion').innerHTML = 'Voici du texte en provenance de
<b>Javascript</b>';
}
```

Vous pouvez voir le paragraphe mystère en cliquant [ici](#).

Voici du texte en provenance de **Javascript**

Vous pouvez voir le paragraphe mystère en cliquant [ici](#).

Remarque : Le paragraphe ayant pour attribut *id="insertion"* est entièrement vide au lancement de la page. Un lien permet de lancer des instructions Javascript et la fonction `ecriture()`. La méthode `getElementById()` permet de s'adresser à une balise selon son attribut *id*.



### A.3.2.1. Affectation d'un contenu dynamique

```
<body>
  <p>Bienvenue sur mon site <b id="ici">mon ami</b></p>
  <p>
    <input type="text" id="prenom" value="Ecrire votre prénom ici" />
    <input type="button" onclick="changeTexte()" value="Valider"/>
  </p>
</body>
```

```
function changeTexte() {
  var prenomSaisi;
  prenomSaisi = document.getElementById('prenom').value;
  document.getElementById('ici').innerHTML = prenomSaisi;
}
```

Bienvenue sur mon site **mon ami**

Ecrire votre prénom ici

Valider

Bienvenue sur mon site **Jacques**

Jacques

Valider

Remarque : Une zone de texte permet de saisir le prénom qui est affecté à la balise **<b id="ici"></b>** grâce à une fonction (**changeTexte()**) déclenchée par un événement (l'appui sur un bouton). Il s'agit de programmation événementielle avec l'attribut *onclick*.

Déclaration, puis affectation d'une variable.

```
Var prenomSaisi;  
prenomSaisi = document.getElementById('prenom').value;
```

Récupère la valeur saisie.

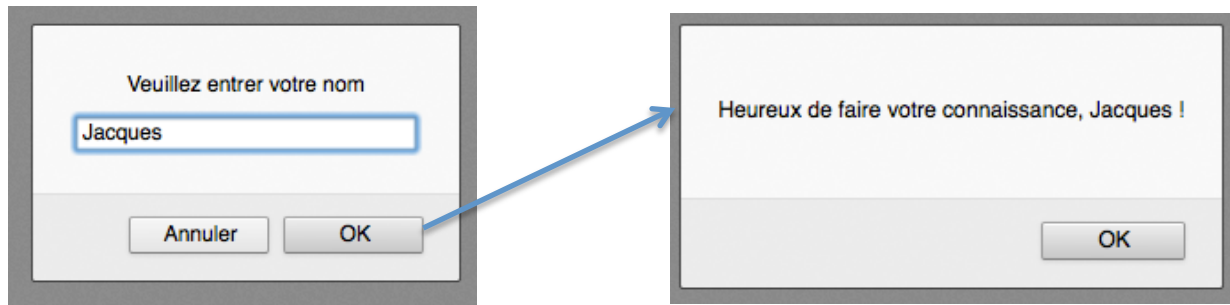


## B. Programmation en Javascript

### B.1. Entrées / sorties, utilisation de variables

#### B.1.1. Affectation

```
/* Script qui parle */  
var nom;  
nom = prompt('Veuillez entrer votre nom','');  
alert('Heureux de faire votre connaissance, ' + nom + ' !');
```



Remarque : Il se peut qu'un avertissement de sécurité du navigateur se déclenche. La fonction `prompt()` sert à ouvrir une boîte de saisie (elle sert donc à affecter une variable) Javascript, `alert()` permet d'ouvrir une boîte de dialogue.

Affectation d'une variable.

```
nom = prompt('Veuillez entrer votre nom','');
```

Boîte de saisie avec un texte et une valeur par défaut.

Boîte de dialogue et son contenu sous forme de chaîne de caractères.

```
alert('Heureux de faire votre connaissance, ' + nom + ' !');
```

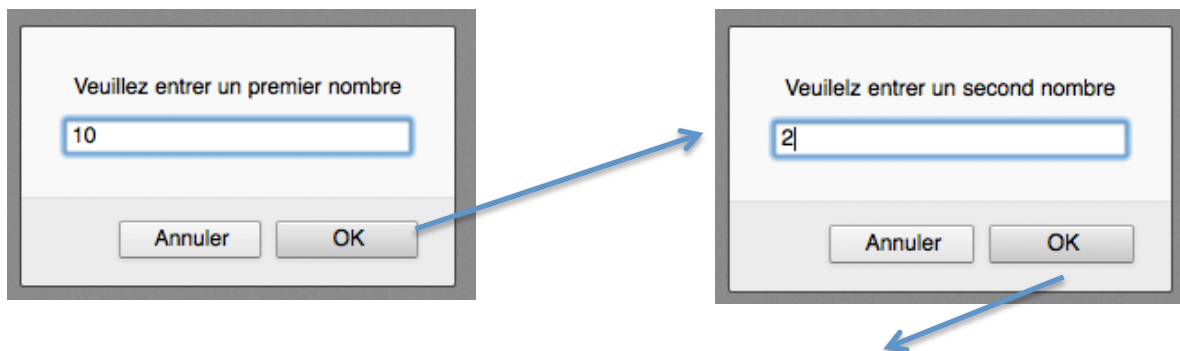
Utilisation d'une variable concaténée à la chaîne de caractères avec « + »

## B.1.2. Calculs

Evidemment JavaScript dispose de toutes les possibilités d'un langage :

Opérateur	Signification	Exemple
+	Addition	<b>a = 10 + c;</b>
-	Soustraction	<b>c = b - 1;</b>
*	Multiplication	<b>u = 10 * 3</b>
/	Division	<b>t = x / y</b>
++	Incrémentation	<b>i++</b>
--	Décrémentation	<b>j--</b>

```
var a, b, resultat;  
a=prompt('Veuillez entrer un premier nombre','0');  
b=prompt('Veuillez entrer un second nombre','0');  
resultat=a*b;  
document.write('<p>Le résultat de l'opération est le suivant : ' + resultat + ' !</p>');
```



Remarque : Une valeur par défaut 0 est proposée dans la saisie. Cette valeur apparaîtra en surbrillance et sera effacée dès que l'on saisira un nombre.

```
<body>  
  <p>Une multiplication :</p>  
  <p>  
    <input type="text" id="n1" value="0" /> x <input type="text" id="n2" value="0" /> =  
  <b id="result"></b><br/>  
    <input type="button" onclick="multiplie()" value="Calculer"/>  
  </p>  
</body>
```

```
function multiplie() {  
  var a, b, resultat;  
  a=document.getElementById('n1').value;  
  b=document.getElementById('n2').value;  
  resultat = a * b;  
  document.getElementById('result').innerHTML = resultat;  
}
```

Une multiplication :

x  =

Une multiplication :

x  = 20  
 →

### B.1.3. Déclarations des variables

JavaScript utilise l'instruction **var** pour la déclaration. Toute nouvelle variable devrait être théoriquement déclarée.

#### *B.1.3.1. Déclaration des variables*

Pour déclarer une variable de type entier, numérique, chaîne de caractères, etc. il ne faut pas déclarer le type. Le navigateur détecte le type lors de l'affectation.

```
var prenomVisiteur = 'Marcel';  
var nomVisiteur = 'Dupond';  
var ageVisiteur = 29;  
var accueil = 'Bonjour ' + prenomVisiteur + ' ' + nomVisiteur + ', vous avez ' + ageVisiteur +  
'ans !';
```

#### *B.1.3.2. Déclaration et création d'objets*

JavaScript intègre nativement plusieurs types d'objets. Par exemple, l'objet **Date** très utile dans un environnement internet.

La déclaration se fait toujours avec **var**. Pour créer un objet, il faut utiliser le mot-clé **new** suivi du type d'objet **Date**. Attention, le respect des majuscules/minuscules est indispensable et source de nombreuses erreurs.

```
var jour = new Date();
```

Remarque : Cette ligne crée un objet **Date** contenant la date du jour.

```
var uneDate = new Date(annee,mois-1,jour,heure,min);
```

Remarque : Cette ligne crée un objet date avec une date paramétrable.

#### *B.1.3.3. Déclaration et utilisation des tableaux*

En JavaScript, les tableaux sont des objets et leur déclaration est identique à celle vue plus haut.

```
var unTableau = new Array(10);
```

Cette ligne crée un tableau de 10 éléments de type indéfini (réel, entier, chaîne de caractères). En JavaScript, le premier élément est indexé à 0. Il est possible de déclarer un tableau sans dimension fixée. La taille du tableau s'adapte en fonction du contenu :

```
var unAutreTableau = new Array;
```

Pour accéder aux éléments du tableau, on utilise les crochets « [ » et « ] » :

```
unTableau[0] = 10;  
unTableau[9] = 5;
```

Des attributs associés à l'objet permettent d'effectuer des traitements ou d'accéder à des propriétés. On utilise la notation pointée pour appliquer une méthode sur un objet ou pour accéder à une propriété.

```
var dimension = unTableau.length;
```

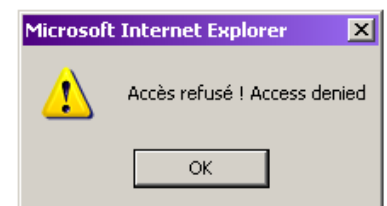
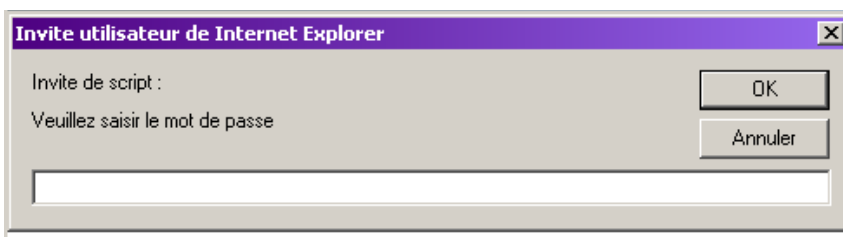
Cette ligne retourne le nombre d'éléments de l'objet **Array** appelé « unTableau ».

## B.2. Structures algorithmiques usuelles

### B.2.1. Les conditions

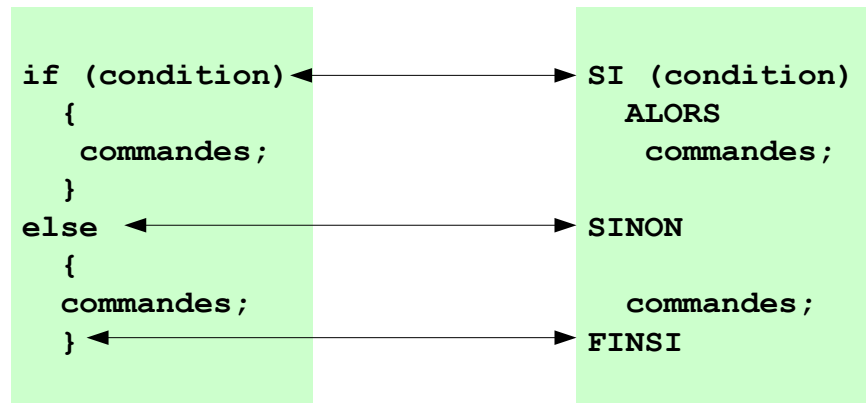
Si l'utilisateur entre le bon mot de passe il peut accéder à la page d'accueil :

```
var a, b, c , motdepasse;  
a = 'tra';  
b = 'la';  
c = 'lere';  
motdepasse = prompt('Veuillez saisir le mot de passe','');  
if (motdepasse == a + b + b + c)  
{  
    location.href='pageaccueil.html';  
}  
else  
{  
    alert('Accès refusé ! Access denied');  
}
```



Remarque : l'exemple est trop « simpliste » au niveau de la sécurité pour être considéré comme un moyen de fixer un mot de passe.

On aura reconnu la forme conditionnelle **SI – ALORS – SINON – FINSI** :



Changement de couleur de la page (avec la commande **switch**) :

```
<body>
  <p>Une multiplication sur fond coloré :</p>
  <p>
    <input type="text" id="n1" value="0" /> x <input type="text" id="n2" value="0" /> =
  <b id="result"></b><br/>
    <input type="button" onclick="multiplie();chgfond();" value="Calculer et colorer"/>
  </p>
</body>
```

```
function multiplie() {
  var a, b, resultat;
  a=document.getElementById('n1').value;
  b=document.getElementById('n2').value;
  resultat = a * b;
  document.getElementById('result').innerHTML = resultat;
}

function chgfond() {
  var couleur;
  couleur = prompt('Choisissez parmi ces couleurs : jaune, gris, rouge','jaune');
  switch(couleur) {
    case 'rouge' :
      document.body.style.backgroundColor = 'red';
      break;
    case 'jaune' :
      document.body.style.backgroundColor = 'yellow';
      break;
    case 'gris' :
      document.body.style.backgroundColor = 'gray';
      break;
    default :
      alert('Cette couleur n'existe pas !!');
  }
}
```

Une multiplication sur fond coloré :

10 x 2 =

Choisissez parmi ces couleurs : jaune, gris, rouge

Une multiplication sur fond coloré :  
10 x 2 = 20

Remarque : l'attribut *onclick* déclenche deux fonctions (*multiplie()* et *chgfond()*). L'attribut *onclick* permet d'insérer directement du code Javascript et donc le lancement des fonctions (séparées par un « ; ») comme pour n'importe quelle instruction.

Accès à une balise selon son nom.

```
document.body.style.backgroundColor = 'gray';
```

Modification du style de la balise avec un background-color « gray ».

Remarque : les styles CSS en Javascript s'inscrivent sans tiret lorsqu'ils en comportent. A la place on met en majuscule l'initiale du mot.

On aura reconnu la forme conditionnelle **SELONQUE FAIRE – CAS )... FINCAS - FINSELONQUE** :

<pre>switch (variable) {   case "valeur1" :     commandes;     break;   case "jaune" :     commandes;     break;   default :     commandes; }</pre>	<pre>SELONQUE (variable) FAIRE   CAS "valeur1" :     commandes;   FINCAS;   CAS "jaune" :     commandes;   FINCAS;   DEFAUT :     commandes;   FINSELONQUE</pre>
---	--

Déviation vers des pages spécifiques (avec la commande **switch**) :

```
var nom;
nom = prompt('Quel est votre nom d'utilisateur ?', '');
switch(nom) {
  case 'pierre' :
    location.href = 'pagepierre.html';
    break;
  case 'marc' :
    location.href = 'pagemarc.html';
    break;
  case 'evelyne' :
    location.href = 'pageevelyne.html';
    break;
  case 'angelique' :
    location.href = 'pageangelique.html';
    break;
  default :
    alert('Cet utilisateur n\'existe pas vous ne pouvez pas accéder aux sites!!');
    location.href = 'pageaccueil.html';
}
```

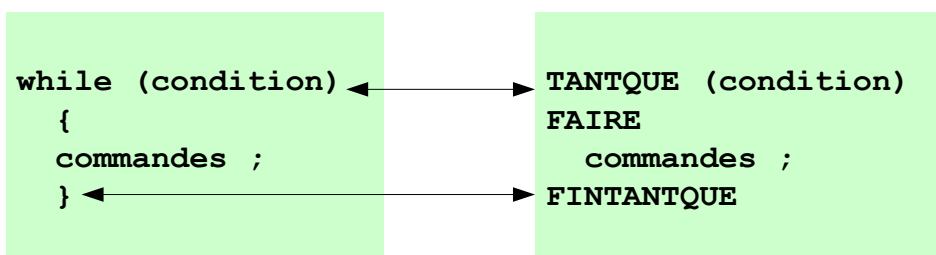
Remarque : **location.href** permet de changer l'URL de la page active.

## B.2.2. Les structures itératives

Mot de passe - version avec un nombre limité d'essais :

```
var secret, motdepasse, i = 0;
secret = 'tralalalere';
while ( (motdepasse != secret) && (i < 3) ) {
  motdepasse = prompt('Veuillez saisir le mot de passe', '');
  if (motdepasse==secret)
  {
    location.href='page.html';
  }
  else
  {
    alert('Le mot de passe est incorrect !');
  }
  i++;
}
```

On aura reconnu la forme itérative **TANTQUE FAIRE – FINTANTQUE** :



## B.3. La gestion d'événements

### B.3.1. Les événements

Les événements JavaScript permettent de lancer des actions spécifiques (plusieurs fois de suite ou plusieurs fonctions différentes) dans une même page. Pour ce faire on les associe souvent à des gestionnaires d'événements.

Événement	Se produit quand...
<b>onabort</b>	Le chargement a été interrompu.
<b>onblur</b>	La souris quitte une zone formulaire.
<b>onclick</b>	On clique sur un objet.
<b>ondblclick</b>	On double-clique sur un objet.
<b>ondragdrop</b>	On dépose en faisant glisser-déposer un élément vers un objet.
<b>onchange</b>	On change une zone texte ou une sélection d'élément.
<b>onerror</b>	Il y a une erreur de script.
<b>onkeydown</b>	On appuie sur une touche de clavier.
<b>onkeypress</b>	On maintient enfoncé une touche de clavier.
<b>onkeyup</b>	On relâche une touche de clavier.
<b>onfocus</b>	La souris entre dans une zone formulaire.
<b>onload</b>	On charge la page dans le navigateur ou un objet graphique.
<b>onmouseout</b>	La souris sort d'une zone image-map, d'un objet
<b>onmouseover</b>	La souris passe au-dessus d'un lien ou d'un formulaire, d'un objet graphique.
<b>onreset</b>	On vide le contenu d'un formulaire.
<b>onresize</b>	On redimensionne la fenêtre du navigateur.
<b>onselect</b>	On sélectionne un élément dans un formulaire.
<b>onsubmit</b>	On envoie un formulaire.
<b>onunload</b>	On sort de la page (de la fenêtre).



### B.3.2. Insertion d'un bouton d'action avec un événement

Changement de couleur de la page :

```
<body>
  <h1>Page avec fonction de couleur</h1>
  <p>
    <input type="button" value="Gris clair" onclick="silver()" />
  </p>
</body>
```

```
function silver() {
  document.body.style.backgroundColor = 'silver';
}
```

**Page avec fonction de couleur**

Gris clair

**Page avec fonction de couleur**

Gris clair

On aura reconnu la déclaration d'une fonction **FONCTION – FIN** :

<pre>function nom_fonction(paramètre) {   commandes ; }</pre>	↔	<pre>FONCTION nom(paramètre) DEBUT   commandes ; FIN</pre>
---	---	--

Remarque : les variables doivent être déclarées avant la fonction.

### B.3.3. Evénements à l'ouverture de la page

```
<body onload="silver()">
  <h1>Page avec fond de couleur</h1>
</body>
```

```
function silver() {
  document.body.style.backgroundColor = 'silver';
}
```

**Page avec fond de couleur**

Avec un passage de paramètre :

```
<body onload="silver('silver')">
  <h1>Page avec fond de couleur</h1>
</body>
```

```
Var couleur;
function silver(couleur){
  document.body.style.backgroundColor = couleur;
}
```

**Page avec fond de couleur**

### B.3.4. Lien hypertexte

```
<body>
  <p>
    Si vous voulez passer à la couleur grise cliquez <a
href="javascript:silver('silver')">ici</a>
  </p>
</body>
```

```
Var couleur;
function silver(couleur){
  document.body.style.backgroundColor = couleur;
}
```

Si vous voulez passer à la couleur grise cliquez [ici](#) → Si vous voulez passer à la couleur grise cliquez [ici](#)

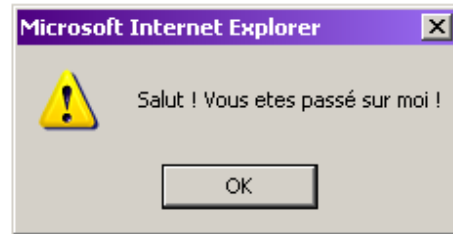
### B.3.5. Evénements liés à la souris

```
<body>
  <h1>Page avec fonction</h1>
  <p>
    
  </p>
</body>
```

```
function bonjour() {
  alert('Salut ! Vous êtes passé sur moi !');
}
```

```
img {
  border: 0px;
}
```

# Page avec fonction



## B.3.6. Fonctions avec passage de paramètre

Changement d'arrière plan :

```
<body>
  <h1>Page avec fonction choix de couleur :</h1>
  <form>
    <p>
      <input type="button" value="rouge" onclick="arrierePlan('red')" />
      <input type="button" value="jaune" onclick="arrierePlan('yellow')" />
      <input type="button" value="gris clair" onclick="arrierePlan('silver')" />
    </p>
  </form>
</body>
```

```
var couleur;
function arrierePlan(couleur) {
  document.body.style.backgroundColor = couleur;
}
```

## Page avec fonction choix de couleur :

rouge jaune gris clair

### B.3.7. Commandes JavaScript « inline »

Ces boutons produisent un changement de page vers Google ou Yahoo.

```
<body>
  <h1>Page avec instructions inline :</h1>
  <form>
    <p>
      <input type="button" value="Google" onclick="location.href='http://www.google.fr' " />
      <input type="button" value="Yahoo" onclick="location.href='http://fr.yahoo.fr' " />
    </p>
  </form>
</body>
```

Ces boutons ouvrent une nouvelle fenêtre (ou un nouvel onglet) vers Google ou Yahoo.

```
<body>
  <h1>Page avec instructions inline :</h1>
  <form>
    <p>
      <input type="button" value="Google" onclick="window.open('http://www.google.fr') " />
      <input type="button" value="Yahoo" onclick="window.open('http://fr.yahoo.fr') " />
    </p>
  </form>
</body>
```

## Page avec instructions inline :

Google Yahoo

Ouverture d'une nouvelle fenêtre.

`window.open('http://www.google.fr');`

URL ou contenu de la nouvelle fenêtre  
(cf. plus bas).

Ces boutons permettent d'aller dans l'historique du navigateur :

```
<body>
  <h1>Page avec instructions inline :</h1>
  <form>
    <p>
      <input type="button" value="Précédent" onclick="javascript:history.back() " />
      <input type="button" value="Suivant" onclick="javascript:history.forward() " />
    </p>
  </form>
</body>
```

## C. Utilisation des objets fenêtres avec JavaScript

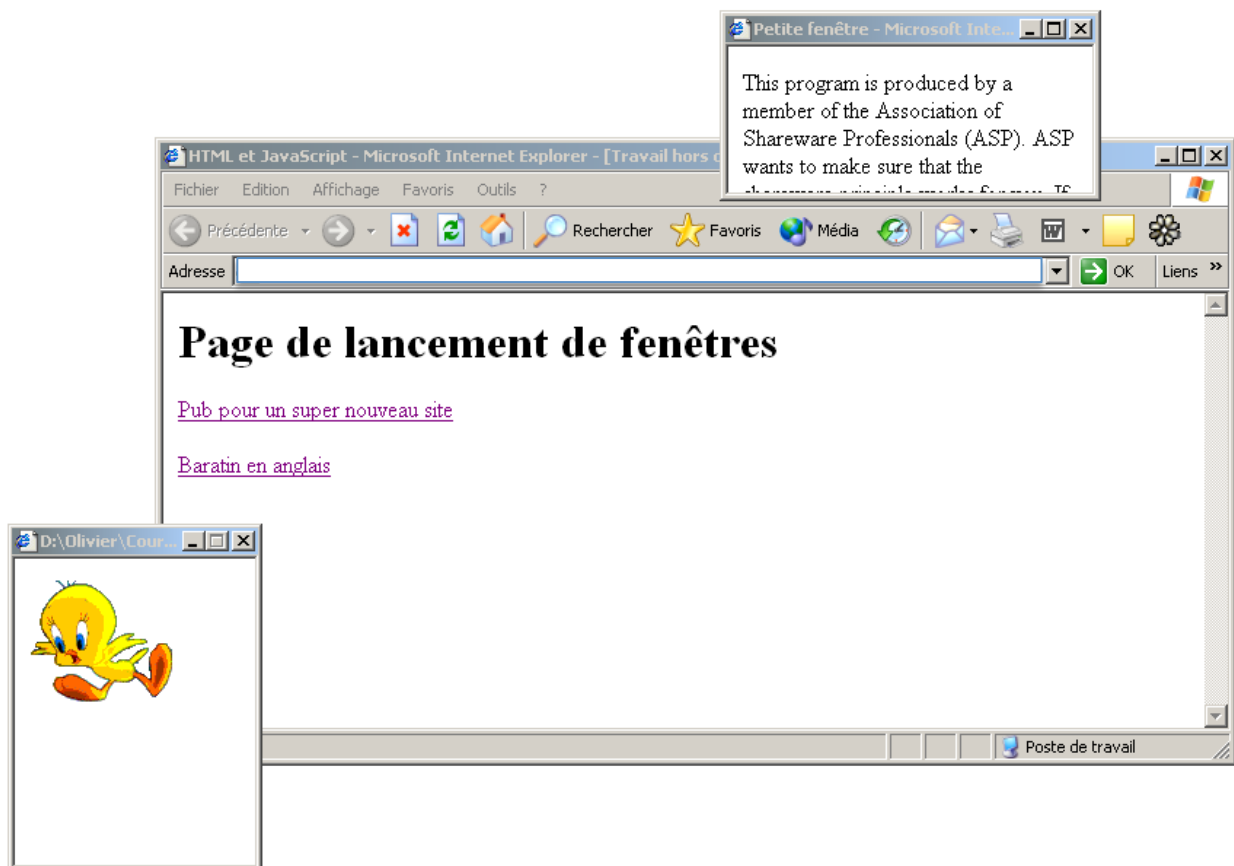
### C.1. Les fenêtres pop-up

Les fenêtres pop-up sont les petites fenêtres (souvent publicitaires) qui se lancent lorsque l'on visite certaines pages. Les navigateurs bloquent ces fenêtres, aussi il est possible que vous ne voyez pas le résultat de votre travail.

```
<body>
  <h1>Page de lancement de fenêtres</h1>
  <p><a href="javascript:fenetre1()">Pub pour un super nouveau site</a></p>
  <p><a href="javascript:fenetre2()">Baratin en anglais</a></p>
</body>
```

```
function fenetre1() {
  var win;
  win = window.open('img/titileft.gif', 'Titi', 'height=210, width=165, left=120,
top=200');
}

function fenetre2() {
  var win;
  win = window.open('texte.html', 'Blablaba', 'height=100, width=250, left=300,
top=150, resizable=yes');
  win.setTimeout('window.close()',20000);
}
```



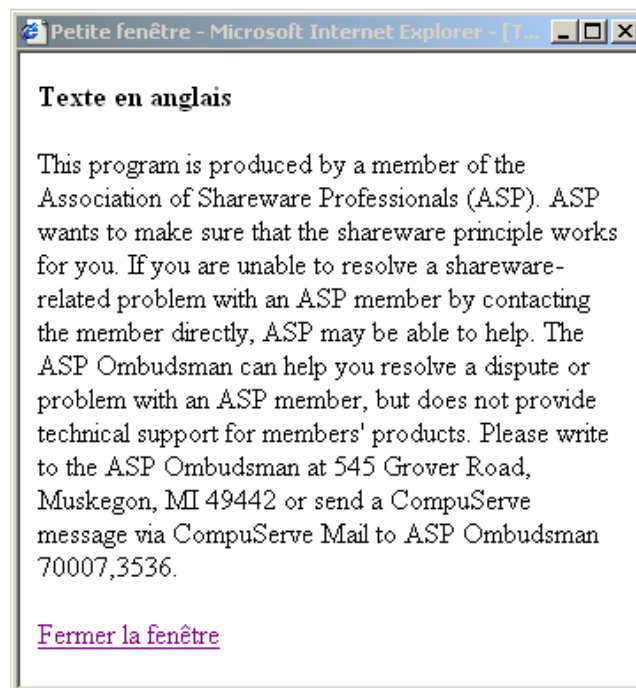
Remarque : On peut prévoir dans la fenêtre qui est affichée sa fermeture automatique au bout d'un délai (à l'aide de `.setTimeout`) ou avec un lien, ou avec un bouton.

La page suivante va s'inscrire dans une « fenêtre » pop-pup :

```
<body>
  <h4>Texte en anglais</h4>
  <p>
    This program is produced by a member of the Association of Shareware Professionals (ASP).
    ASP wants to make sure that the shareware principle works for you. If you are unable to resolve
    a shareware-related problem with an ASP member by contacting the
    member directly, ASP may be able to help. The ASP Ombudsman can help you resolve a dispute
    or problem with an ASP member, but does not provide technical support for members' products.
    Please write to the ASP Ombudsman at 545 Grover Road, Muskegon, MI
    49442 or send a CompuServe message via CompuServe Mail to ASP Ombudsman 70007,3536.<br/>
    <a href="javascript:window.close()">Fermer la fenêtre</a>
  </p>
</body>
```

Le script suivant générera la fenêtre :

```
function fenetre()
{
  var win;
  win = window.open('fichiers/texte2.html', 'Blablabla', 'height=200, width=250, left=300,
top=150, resizable=yes');
}
```



Remarque : L'instruction avec un lien et l'ordre `javascript:window.close()` aurait pu être remplacée par un bouton :

```
<input type="button" value="Fermer" onclick="window.close()" />
```

## C.2. Créer une fenêtre à la volée

Il est également possible de créer une fenêtre XHTML avec JavaScript sans que celle-ci n'ait d'existence propre dans un fichier source (cf. plus haut la fenêtre avec un oiseau qui est également dans ce cas).

Exemple : une fenêtre est créée à la volée, elle est programmée pour se fermer au bout de 20000 millisecondes (mais on y a tout de même ajouté un bouton pour fermer).

```
var win;
win=window.open('', 'Nom', 'height=200, width=300, left=0, top=0');
win.document.write('<?xml version="1.0" encoding="UTF-8" ?>');
win.document.write('<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">');
win.document.write('<html xmlns="http://www.w3.org/1999/xhtml">');
win.document.write('<head><title>Super document</title>');
win.document.write('<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />');
win.document.write('<link rel="stylesheet" type="text/css" media="screen" href="css/style.css"
/></head>');
win.document.write('<body><h3>Texte de la fenêtre créée à la volée</h3>');
win.document.write('<p><input type="button" value="Fermer" onclick="window.close()" />');
win.document.write('</form></body></html>');
win.setTimeout('window.close()',20000);
```

Même exemple en concaténant dans une seule chaîne tout le contenu de la fenêtre :

```
var win;
win=window.open('', 'Nom', 'height=200, width=300, left=0, top=0');
win.document.write('<?xml version="1.0" encoding="UTF-8" ?>'
+ '<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">'
+ '<html xmlns="http://www.w3.org/1999/xhtml">'
+ '<head><title>Super document</title>'
+ '<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />'
+ '<link rel="stylesheet" type="text/css" media="screen" href="css/style.css" /></head>'
+ '<body><h3>Texte de la fenêtre créée à la volée</h3>'
+ '<p><input type="button" value="Fermer" onclick="window.close()" />'
+ '</form></body></html>');
win.setTimeout('window.close()',20000);
```

Même exemple mais avec une variable `contenu` qui concatène (`+=`) tous les éléments de la fenêtre :

```
var win, contenu;
win=window.open('', 'Nom', 'height=200, width=300, left=0, top=0');
contenu = '<?xml version="1.0" encoding="UTF-8" ?>';
contenu += '<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">';
contenu += '<html xmlns="http://www.w3.org/1999/xhtml">';
contenu += '<head><title>Super document</title>';
contenu += '<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />';
contenu += '<link rel="stylesheet" type="text/css" media="screen" href="css/style.css"
/></head>';
contenu += '<body><h3>Texte de la fenêtre créée à la volée</h3>';
contenu += '<p><input type="button" value="Fermer" onclick="window.close()" />';
contenu += '</form></body></html>';
win.document.write(contenu);
win.setTimeout('window.close()',20000);
```

---

## Texte de la fenêtre créée à la volée

Fermer

Remarque : la fenêtre contient bien toutes les balises pour déclarer un document XHTML. Les multiples `win.document.write()` (ou la longue chaîne concaténée) servent à construire le document avec un aisance plus grande que si l'on saisisait toute la structure en une seule ligne.

Remarque : les guillemets restent dédiés aux instructions HTML alors que les quotes concernent le langage Javascript et ses chaînes de caractères (ici toutes les instructions HTML sont inscrites dans des chaînes de caractères). S'il y avait d'autres quotes cela signifierait une rupture dans la chaîne de caractère pour ajouter, par exemple, le contenu d'une variable comme vu précédemment.

Voici les différentes propriétés que l'on peut gérer dans une fenêtre :

Propriétés	Effets	Valeurs possibles
<b>directories</b>	Affichage de la barre de liens.	yes   no
<b>menubar</b>	Affichage de la barre de menu.	yes   no
<b>status</b>	Affichage de la barre de statut.	yes   no
<b>location</b>	Affichage de la zone d'adresse.	yes   no
<b>scrollbars</b>	Affichage des barres de scrolling.	yes   no   auto
<b>resizable</b>	Autorise le redimensionnement du pop-up.	yes   no
<b>height</b>	Hauteur en pixels.	nombre entier
<b>width</b>	Largeur en pixels/	nombre entier
<b>left</b>	Position horizontale en pixels sur l'écran.	nombre entier
<b>top</b>	Position verticale en pixels sur l'écran.	nombre entier
<b>fullscreen</b>	Pop-up en plein écran.	yes   no



## D. Fonctions et objets usuels de Javascript

### D.1. Déviations de pages

Grâce à JavaScript on peut également opérer des déviations (en cas de déménagement par exemple).

Déviation simple :

```
<body>
  <h1>Nous avons déménagé</h1>
</body>
```

```
location.href='autrepageweb.html';
```

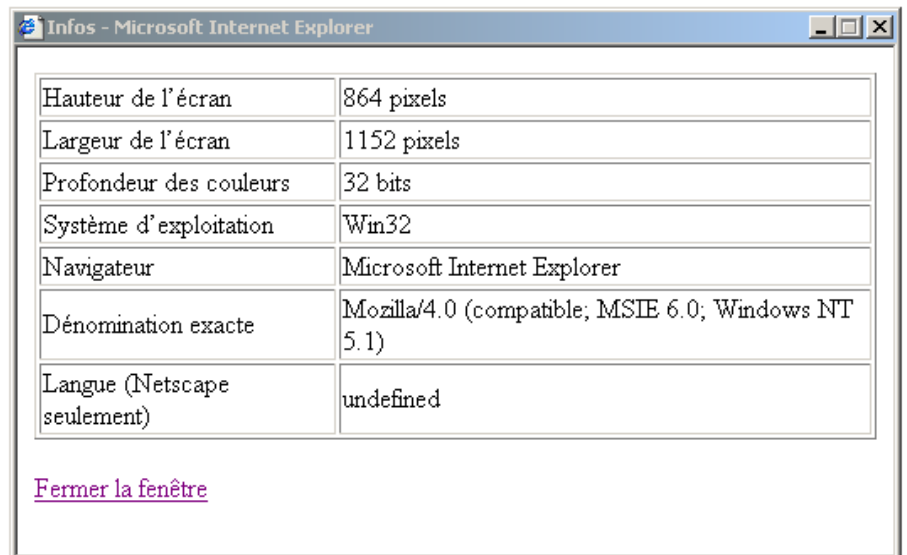
Déviation différée (pour laisser le temps de lire l'annonce) :

```
setTimeout('location.href=\'autrepageweb.html\'',5000);
```

### D.2. Afficher ou utiliser les paramètres liés au navigateur

```
<body>
  <p>
    <input type="button" value="Informations sur votre système" onclick="info()" />
  </p>
</body>
```

```
function info() {
  var hauteur, largeur, couleurs, os, nom, navigateur, langue, showin;
  hauteur = screen.height;
  largeur = screen.width;
  couleurs = screen.colorDepth;
  os = navigator.platform;
  nom = navigator.appName;
  navigateur = navigator.userAgent;
  langue = navigator.language ;
  showin=window.open('', 'Nom', 'height=200, width=300, left=0, top=0');
  showin.document.write('<?xml version="1.0" encoding="UTF-8" ?>\'
    + \'<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN "
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">\'
    + \'<html xmlns="http://www.w3.org/1999/xhtml">\'
    + \'<head><title>Infos</title>\'
    + \'<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />\'
    + \'<link rel="stylesheet" type="text/css" media="screen" href="css/style.css "
/></head>\'
    + \'<body><table>\'
    + \'<tr><td>Hauteur de l\'écran</td><td>\' + hauteur + \' pixels</td></tr>\'
    + \'<tr><td>Largeur de l\'écran</td><td>\' + largeur + \' pixels</td></tr>\'
    + \'<tr><td>Profondeur des couleurs</td><td>\' + couleurs + \' bits</td></tr>\'
    + \'<tr><td>Système d\'exploitation</td><td>\' + os + \'</td></tr>\'
    + \'<tr><td>Navigateur</td><td>\' + nom + \'</td></tr>\'
    + \'<tr><td>Dénomination exacte</td><td>\' + navigateur + \'</td></tr>\'
    + \'<tr><td>Langue (Netscape seulement)</td><td>\' + langue + \'</td></tr></table>\'
    + \'<p><a href="javascript:window.close()">Fermer la fenêtre</a></p>\'
    + \'</body></html>');
}
```



### D.3. Affichage dynamique de la date de la machine cliente

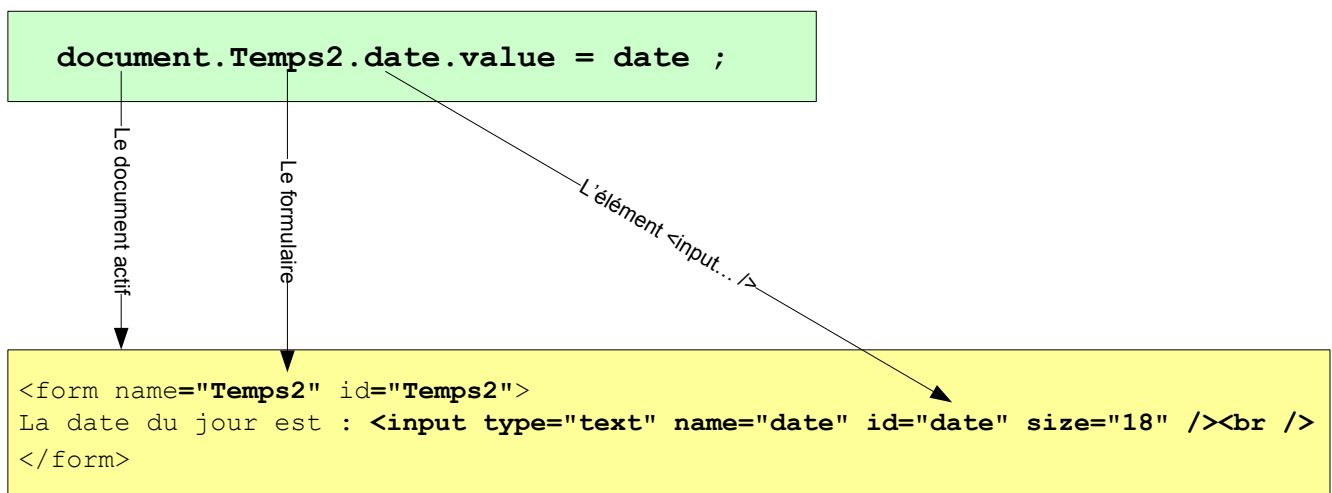
```
<body onload="afficheDate()">
  <form name="Temps2" id="Temps2">
    <p>
      La date du jour est : <input type="text" name="date" id="date" size="18" /><br />
    </p>
  </form>
</body>
```

```
function afficheDate() {
  var adate, date, amois;
  adate = new Date();
  date = adate.getDate();
  amois = adate.getMonth()+1;
  if (amois == 1) date += ' janvier';
  else if (amois == 2) date += ' février';
  else if (amois == 3) date += ' mars';
  else if (amois == 4) date += ' avril';
  else if (amois == 5) date += ' mai';
  else if (amois == 6) date += ' juin';
  else if (amois == 7) date += ' juillet';
  else if (amois == 8) date += ' août';
  else if (amois == 9) date += ' septembre';
  else if (amois == 10) date += ' octobre';
  else if (amois == 11) date += ' novembre';
  else if (amois == 12) date += ' décembre';
  date += ' ' + adate.getYear();
  document.Temps2.date.value = date ;
}
```

La date du jour est : 22 février 2002

Remarque : Une feuille de style associée à l'élément `<input/>` pourra améliorer l'affichage de la date.

On remarquera de quelle manière on interagit avec un formulaire (ici pour afficher la date).



## D.4. Navigation et menus avec JavaScript

Il est également possible de lancer l'exécution de plusieurs fonctions ou de lancer plusieurs pages avec les menus déroulants (cf. les chapitres précédents pour plus de détails sur les formulaires).

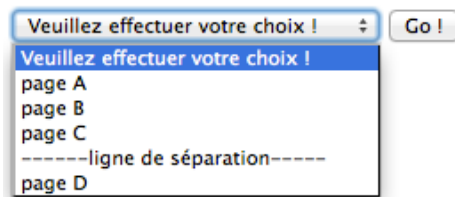
### D.4.1. Menus déroulant

Un menu déroulant avec un bouton GO :

```
<body>
  <h1>Menu déroulant avec un bouton Go</h1>
  <form id="liste1">
    <p>
      <select name="choix" id="choix">
        <option value="rien" selected="selected">Veuillez effectuer votre choix !</option>
        <option value="pagea.html">page A</option>
        <option value="pageb.html">page B</option>
        <option value="pagec.html">page C</option>
        <option value="rien">-----ligne de séparation-----</option>
        <option value="paged.html">page D</option>
      </select>
      <input type="button" value="Go !" onclick="changer()" />
    </p>
  </form>
</body>
```

```
function changer() {
    var selection, page;
    selection = document.liste1.choix.selectedIndex;
    page = document.liste1.choix.options[selection].value;
    if (page == 'rien')
    {
        document.liste1.reset();
        return false;
    }
    else
    {
        location.href = page;
    }
}
```

## Menu déroulant avec un bouton Go



### Remarque :

On récupère l'indice du choix effectué (ici de 0 à 5) :

- `selection = document.liste1.choix.selectedIndex;`

Puis on récupère le contenu du choix opéré (qui est assigné avec `<option value="..."></option>`) :

- `page = document.liste1.choix.options[selection].value;`

L'ajout de `return false` permet d'arrêter le script.

### D.4.2. Images sensibles

Il est possible de changer l'aspect d'un objet lorsque l'on passe dessus avec la souris. On peut recourir à des images pour changer l'aspect des liens ou changer leur système d'apparence.

Nous avons construit deux images qui formeront un lien vers une page web. Nous avons décidé qu'au moment où l'on positionnera la souris sur notre image celle-ci changera d'aspect :

Avant (bout1.gif)

**Menu**

Après (bout2.gif)

**Menu**

Passage de la souris = changement d'image

```
<body>
<h1>Le menu dynamique</h1>
<p>
    <a href="pagea.html" onmouseover="document.lien1.src=menu2.src"
onmouseout="document.lien1.src=menu1.src"></a>
</p>
</body>
```

Départ de la souris = changement d'image

```
var menu1 = new Image();
var menu2= new Image();
menu1.src = 'img/bout1.gif';
menu2.src = 'img/bout2.gif';
```

```
body {
    background: darkcyan;
}

img {
    width: 50px;
    height: 25px;
    border: 0px;
}
```

Autre approche avec le code en ligne situé dans la balise `<a></a>` :

```
<body>
  <h1>Le menu dynamique</h1>
  <p>
    <a href="pagea.html" onmouseover="document.lien1.src='img/bout2.gif' "
onmouseout="document.lien1.src='img/bout1.gif' "></a>
  </p>
</body>
```

Utilisation complémentaire de Javascript (redirection) et de CSS (image sensitive) :

```
<body>
  <h1>Le menu dynamique</h1>
  <p id="lien1" onclick="location.href='pagea.html' " >
  </p>
</body>
```

```
p#lien1 {
    background-image: url("../img/bout1.gif");
    width: 50px;
    height: 25px;
}

p#lien1:hover {
    background-image: url("../img/bout2.gif");
}
```

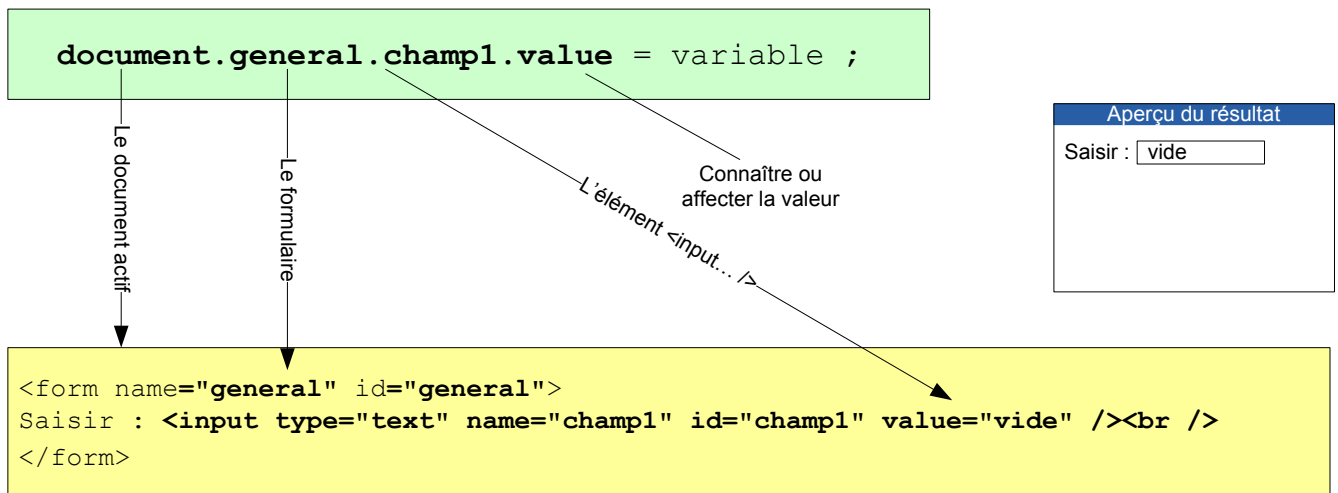
## E. Les formulaires

### E.1. L'accès aux formulaires

#### E.1.1. Principe de l'accès aux éléments de formulaire

Les éléments de formulaire sont des objets Javascript auxquels on peut accéder via des scripts.

Voici le principe selon lequel on accède à un objet :



#### E.1.2. Accéder au formulaire

Le formulaire est un élément de l'objet document. Pour accéder au formulaire « general », il faut écrire :

`document.forms['general']`

ou

`document.forms[0]`

ou

`document.general`

`forms` est le tableau des formulaires de document. On peut accéder à un formulaire par son nom ou par son indice (commençant à 0), via le tableau `elements` ou directement par son nom.

### E.1.3. Accéder à un élément

Pour accéder maintenant à une zone de texte, on écrit :

```
document.forms['general'].elements['champ1']
```

ou

```
document.forms['general'].elements[0]
```

ou

```
document.forms['general'].champ1
```

ou

```
document.general.champ1
```

`elements` est le tableau de tous les éléments du formulaire. On peut accéder à un élément par son nom ou par son indice, ou directement par son nom.

### E.1.4. Manipuler les propriétés d'un élément

Une fois que l'élément est atteint, il est possible de manipuler ses propriétés. Par exemple, pour placer dans la zone de texte le mot « NOUVEAU », il faut écrire :

```
document.forms['general'].elements['champ1'].value = 'NOUVEAU';
```

ou

```
document.general.champ1.value = 'NOUVEAU';
```

### E.1.5. Appeler une méthode sur un élément

Pour donner le focus au champ texte du haut de cette page, il faut appeler la méthode `focus()` sur cet élément.

```
document.forms['general'].elements['champ1'].focus();
```

ou

```
document.general.champ1.focus();
```

### E.1.6. Modifier un élément sur un événement

Un événement peut intervenir sur un élément de formulaire. Reprenons le même exemple, il s'agit de placer « NOUVEAU » dans la zone de texte du formulaire.

```
<body>
  <form id="changer">
    <p>
      <input type="text" name="zoneTexte" value="Valeur initiale" />
      <input type="button" value="Changer la zone de texte"
      onclick="document.forms['changer'].elements['zonetexte'].value = 'Encore nouveau' " />
    </p>
  </form>
</body>
```



Valeur initiale    Changer la zone de texte → Encore nouveau    Changer la zone de texte

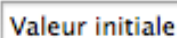
Dans le bouton, on a rajouté l'attribut *onclick* qui reçoit le code JavaScript à exécuter lors du clic sur le bouton.

### E.1.7. Désigner l'objet par « this »

Il est fastidieux d'accéder aux éléments de formulaire par toute la chaîne `document.forms.elements`. Un objet JavaScript `this` permet de raccourcir ce chemin d'accès. `this` représente l'objet JavaScript en cours.

Le texte initial d'une zone de texte disparaît lorsque l'on fait le focus dessus :

```
<body>
  <form id="changer">
    <p>
      <input type="text" name="zoneTexte" value="Valeur initiale" onfocus="this.value=''" />
    </p>
  </form>
</body>
```



Valeur initiale    Changer la zone de texte



## E.2. L'accès aux éléments de formulaire

### E.2.1. L'accès aux éléments de type INPUT

#### E.2.1.1. Les zones de texte

La principale action en JavaScript sur une zone de texte est de manipuler son contenu. Imaginons un formulaire appelé « monForm » qui possède un champ texte appelé « monChamp ». On accède au contenu du champ par :

```
document.forms['monForm'].elements['monChamp'].value
```

ou

```
document.monForm.monChamp.value
```

Il faut bien penser à ajouter la propriété `.value` pour accéder au contenu.

#### E.2.1.2. Les boutons

Un bouton sert principalement à déclencher une action JavaScript par le clic de la souris. La propriété `.value` contient le libellé du bouton. Comme pour une zone de texte, ce libellé est accessible.

```
document.forms['monForm'].elements['monBouton'].value
```

ou

```
document.monForm.monBouton.value;
```

#### E.2.1.3. Les cases à cocher

Pour détecter qu'une case est cochée, il faut utiliser sa propriété `.checked` :

```
document.forms['monForm'].elements['maCase'].checked
```

ou

```
document.monForm.maCase.checked
```

`.checked` est de type booléen, il contient `true` pour vrai et `false` pour faux. Il faudra une condition pour vérifier si la case a été cochée ou non.

```

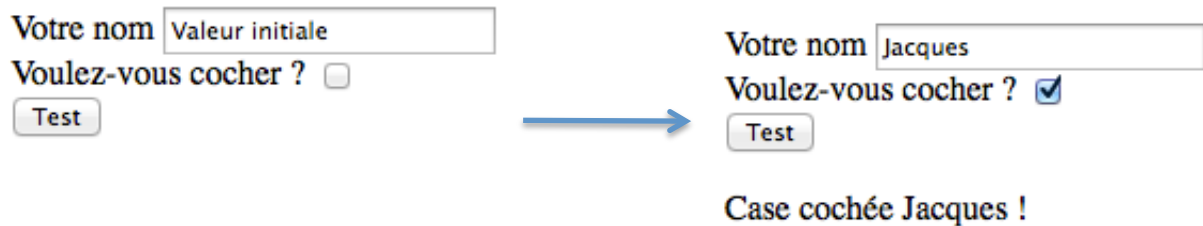
<body>
  <form id="recup" name="recup">
    <p>
      Votre nom <input type="text" name="zoneTexte" value="Valeur initiale"
onfocus="this.value=''" /><br />
      Voulez-vous cocher ? <input type="checkbox" name="coche" /><br />
      <input type="button" value="Test" onclick="go()" />
    </p>
    <p id="resultat"></p>
  </form>
</body>

```

```

function go() {
  if (document.recup.coche.checked) {
    var nom = document.recup.zoneTexte.value;
    document.getElementById('resultat').innerHTML = 'Case cochée ' + nom + ' !';
  }
}

```



Remarque : La condition peut s'écrire de manière raccourcie (`document.recup.coche.checked`) ou avec une égalité explicite (`document.recup.coche.checked == 'true'`).

Pour accéder au contenu de la balise `<p></p>` nous avons utilisé la méthode `getElementById()` qui permet de s'adresser à n'importe quelle balise selon son *id*. Cela implique donc que tous les éléments doivent avoir un *id*. Ici le formulaire a un attribut *name* et *id* car les attributs *name* sont indispensables dans la forme d'accès aux éléments présentée.

#### E.2.1.4. Les cases à cocher et la méthode `getElementById()`

Grâce aux *id* des éléments on va pouvoir faire la même chose, de façon plus condensée, en s'adressant directement aux éléments. On pourra en plus récupérer une valeur sur la case à cocher.

```

<body>
  <form id="recup">
    <p>
      Votre nom <input type="text" name="zoneTexte" id="zoneTexte" value="Valeur initiale"
onfocus="this.value=''" /><br />
      Voulez-vous cocher ? <input type="checkbox" name="coche" id="coche"
value="Il a coché !" /><br />
      <input type="button" value="Test" onclick="go()" />
    </p>
    <p id="resultat"></p>
  </form>
</body>

```

```

function go() {
  if (document.getElementById('coche').checked) {
    var nom = document.getElementById('zoneTexte').value;
    var cont = document.getElementById('coche').value;
    document.getElementById('resultat').innerHTML = 'Case cochée ' + nom + ' ! ' + cont;
  }
}

```

Votre nom

Voulez-vous cocher ? ☒

Case cochée Jacques ! Il a coché !

Remarque : On a pu récupérer toutes les valeurs des éléments grâce à leur *id*.

### E.2.1.5. Les boutons-radios

On utilisera également la propriété `.checked` :

`document.forms['monForm'].elements['monBouton'][i].checked`

ou

`document.monForm.monBouton[i].checked`

où *i* est incrémenté de 0 jusqu'au nombre de boutons-radios nommés `monBouton`, et permet de parcourir l'état de chaque bouton jusqu'à ce que l'on trouve lequel est sélectionné.

```
<body>
  <form id="recup" name="recup">
    <p>
      Votre nom <input type="text" name="zoneTexte" id="zoneTexte" value="Valeur initiale"
onfocus="this.value=''" /><br />
      Quel est votre OS ?
      <input type="radio" name="os" value="Windows 9x" />Windows 9x
      <input type="radio" name="os" value="Windows 2000" />Windows 2000
      <input type="radio" name="os" value="Windows XP" checked="checked" />Windows XP
      <input type="radio" name="os" value="Windows Vista" />Windows Vista
      <input type="radio" name="os" value="Windows Seven" />Windows Seven
      <input type="radio" name="os" value="Linux" />Linux
      <input type="radio" name="os" value="Mac" />Mac<br />
      <input type="button" value="Test" onclick="go()" />
    </p>
    <p id="resultat"></p>
  </form>
</body>
```

```
function go() {
  for (var i = 0; i < document.recup.os.length; i++) {
    if (document.recup.os[i].checked) {
      var nom = document.recup.zoneTexte.value;
      var cont = document.recup.os[i].value;
      document.getElementById('resultat').innerHTML = 'Eh bien ' + nom + ', vous utilisez
1\'os ' + cont;
    }
  }
}
```

Remarques : Le formulaire contient un groupe de boutons-radios liés dont le nom est « os ».

Le script JavaScript parcourt le groupe des boutons-radios avec une boucle. On repère le bouton qui a la propriété `.checked` à `true` et on affiche alors la valeur correspondante.

Votre nom

Quel est votre OS ? ☐ Windows 9x ☐ Windows 2000 ☐ Windows XP ☐ Windows Vista ☐ Windows Seven ☐ Linux ☒ Mac

Eh bien Jacques, vous utilisez l'os Mac

### E.2.1.6. L'accès aux éléments de type SELECT

Pour récupérer l'indice la ligne sélectionnée :

```
document.monForm.elements['maListe'].selectedIndex
```

Pour récupérer le nombre de lignes :

```
document.monForm.elements['maListe'].options.length
```

Pour récupérer la valeur de la ligne sélectionnée :

```
document.monForm.elements['maListe'].  
options[this.monForm.elements['maListe'].selectedIndex].value
```

ou (avec trois lignes) :

```
var selection, contenu;  
selection = document.monForm.maListe.selectedIndex;  
contenu = document.monForm.maListe.options[selection].value;
```

En JavaScript, la structure d'un élément de type select reprend ce schéma :

<b>name</b>	Nom de la liste.
<b>selectedIndex</b>	Indice de la ligne sélectionnée (ligne 1 : indice=0).
<b>options</b>	Tableau des lignes.
<b>length</b>	Nombre de lignes.
<b>value</b>	Valeur d'une ligne.
<b>text</b>	Libellé d'une ligne.

```
<body>  
  <form id="recup" name="recup">  
    <p>  
      Votre nom <input type="text" name="zoneTexte" id="zoneTexte" value="Valeur initiale"  
onfocus="this.value=''" /><br />  
      Quel est votre OS ?  
      <select name="os" id="os">  
        <option value="Windows 9x">Windows 9x</option>  
        <option value="Windows 2000">Windows 2000</option>  
        <option value="Windows XP" selected="selected">Windows XP</option>  
        <option value="Windows Vista">Windows Vista</option>  
        <option value="Windows Seven">Windows Seven</option>  
        <option value="Linux">Linux</option>  
        <option value="Mac">Mac</option>  
      </select><br />  
      <input type="button" value="Test" onclick="go()" />  
    </p>  
  </form>  
  <p id="resultat"></p>  
</body>
```

```
function go() {
    var nom, selection, contenu;
    nom = document.getElementById('zoneTexte').value;
    selection = document.recup.os.selectedIndex;
    contenu = document.recup.os.options[selection].value;
    document.getElementById('resultat').innerHTML = 'Eh bien ' + nom + ', vous utilisez l\'os '
+ contenu;
}
```

Votre nom

Quel est votre OS ?

Eh bien Jacques, vous utilisez l'os Mac

Avec `getElementById()` :

```
function go() {
    var nom, contenu;
    nom = document.getElementById('zoneTexte').value;
    contenu = document.getElementById('os').value;
    document.getElementById('resultat').innerHTML = 'Eh bien ' + nom + ', vous utilisez l\'os '
+ contenu;
}
```

#### *E.2.1.7. L'accès aux éléments de type TEXTAREA*

Une zone de texte multi-lignes a comme propriété principale `value` qui contient le texte de la zone. Pour récupérer le contenu de la zone, on utilise le classique :

```
document.forms['monForm'].elements['maZone'].value
```

ou

```
document.monForm.maZone.value
```

Le « contenu traité » est obtenu par l'appel de la fonction `escape()` qui convertit tous les caractères spéciaux et non visibles (tabulations, retour à la ligne...) :

```
escape(document.forms['monForm'].elements['maZone'].value)
```

ou

```
escape(document.monForm.maZone.value)
```

La fonction inverse est `unescape()`.

## E.2.2. Exemple complet

Soit le formulaire qui permet à un client de choisir un produit :

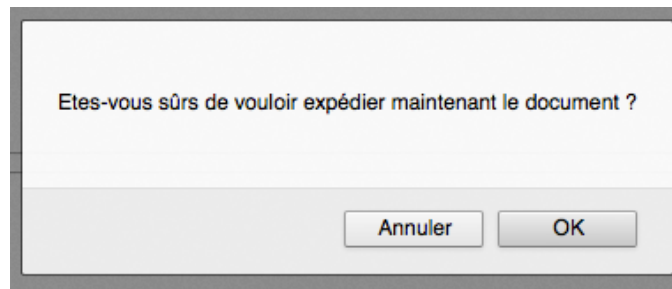
```
<body>
  <h1>Commandez un produit !</h1>
  <form action="valideCde.php" method="post" name="question" id="question" onreset="return
securite('vider')" onsubmit="return securite('expédier maintenant')">
    <fieldset>
      <legend>Votre Commande :</legend>
      Le produit coûte 300€ et les frais de port s'élèvent à 45€.<br/><br/>
      Je commande exactement <input type="text" size="2" maxlength="2" name="nombre"
id="nombre" value="1"/> unités du produit<br /><br/>
      <input type="checkbox" name="express" id="express" value="oui"/> Livraison express (70€
de supplément)<br/><br/>
      Je le voudrais en
      <input type="radio" name="sorte" id="sorte" value="noir" checked="checked"/>Noir
      <input type="radio" name="sorte" id="sorte" value="argent"/>Argent
      <input type="radio" name="sorte" id="sorte" value="or"/>Or<br/>
    </fieldset>
    <fieldset>
      <legend>Vos Coordonnées :</legend>
      Titre :
      <select name="titre" id="titre" size="1">
        <option value="M.">M.</option>
        <option value="Mme">Mme</option>
      </select><br /><br/>
      Prénom : <input type="text" name="prenom" id="prenom" /><br /><br/>
      Nom : <input type="text" name="nom" id="nom" /><br /><br/>
      Numéro client : <input type="text" name="nrclient" id="nrclient" maxlength="8" /><br
/><br/>
      Adresse E-mail : <input type="text" name="email" id="email" /><br /><br/>
      <input type="button" value="Vérifier la saisie et calculer le montant"
onclick="formtest()" />
      <input type="reset" value="Vider le formulaire" />
      <input type="submit" value="Envoyer les données" /><br/>
    </fieldset>
  </form>
</body>
```

La fonction `securite()` peut se présenter de deux manières :

```
/* Cette partie se lance pour confirmer les actions (reset ou submit) */
var action;
function securite(action) {
  if (confirm('Etes-vous sûrs de vouloir ' + action + ' le document ?')) {
    if (action == 'vider') {
      document.forms['question'].reset();
    }
    else {
      document.forms['question'].submit();
    }
  }
}
```

La même fonction `securite()` qui retourne false lorsque l'on répond négativement :

```
/* Cette partie se lance pour confirmer les actions (reset ou submit) */
var action;
function securite(action) {
  if (!confirm('Etes-vous sûrs de vouloir ' + action + ' le document ?')) {
    return false ;
  }
}
```



Remarques : La fonction de confirmation `securite(action)` affiche un message adapté suivant l'action décidée (*reset* ou *submit*).

L'instruction `return` permet de retourner un résultat (`true/false`) et d'indiquer si le contrôle ou l'action effectuée par la fonction s'est bien déroulée. Cela peut permettre de reprendre ce résultat dans une autre fonction par exemple ou de laisser l'événement se poursuivre (*onreset* ou *onsubmit* déclenchent leur action sauf en cas de `return false`).

La fonction `formtest()` appelée par un bouton d'action :

```
/* Cette partie contrôle les champs du formulaire */

function formtest() {
    var prenom, nom, nrclient, email, arobase, point, i, resultat;
    prenom = document.question.prenom.value;
    nom = document.question.nom.value;
    nrclient = document.question.nrclient.value;
    email = document.question.email.value;
    if (nom == '' || prenom == '' || nrclient == '' || email == '') {
        alert('Remplissez s\'il vous plait tous les champs !');
        return false;
    }
    arobase = 0;
    point = 0;
    for (i = 0; i < email.length; i++) {
        if (email.charAt(i) == '@')
        {
            arobase = 1;
        }
        if (email.charAt(i) == '.')
        {
            point = 1;
        }
    }
    if (arobase != 1 || point != 1 || email.length < 7) {
        alert('Ce n\'est pas une adresse E-mail valide !');
        document.question.email.focus();
        document.question.email.select();
        return false;
    }
    resultat = document.question.nombre.value * 300;
    resultat = resultat + 45;
    if (document.question.express.checked) {
        resultat = resultat + 25;
    }
    alert('Bonjour, ' + prenom + ' ' + nom + ',\nVous devez payer ' + resultat + ' € !');
}
```

# Commandez un produit !

Votre Commande :  
Le produit coûte 300€ et les frais de port s'élèvent à 45€.

Je commande exactement  unités du produit

☒ Livraison express (70€ de supplément)

Je le voudrais en ☐ Noir ☒ Argent ☐ Or

Vos Coordonnées :

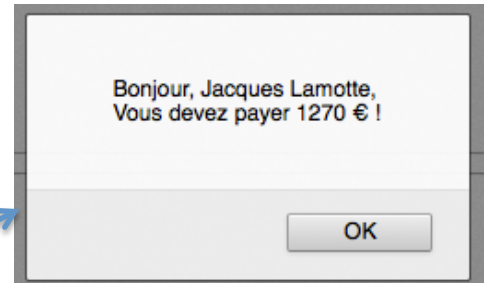
Titre :

Prénom :

Nom :

Numéro client :

Adresse E-mail :



# Commandez un produit !

Votre Commande :  
Le produit coûte 300€ et les frais de port s'élèvent à 45€.

Je commande exactement  unités du produit

☐ Livraison express (70€ de supplément)

Je le voudrais en ☒ Noir ☐ Argent ☐ Or

Vos Coordonnées :

Titre :

Prénom :

Nom :

Numéro client :

Adresse E-mail :



Remarque : La vérification du contenu du formulaire (champs non vides, composition de l'adresse mail) est effectuée par la fonction `formtest()`.



## F. Le DHTML

### F.1. Qu'est-ce que le DHTML ?

Le DHTML (Dynamic HTML) est devenu incontournable depuis que les navigateurs supportent de mieux en mieux ses préconisations.

Malheureusement, l'apprentissage du DHTML est complexe, tant à cause des possibilités offertes qu'en raison des soucis de compatibilité entre navigateurs.

Au tout premier plan du DHTML on trouve la gestion événementielle et l'utilisation de fenêtre pop-up déjà expliquées plus haut, la gestion des cookies, la gestion des styles (calques, affectation des styles, etc.)

L'utilisation d'Ajax (*Asynchronous Javascript and XML*) que nous verrons à l'occasion de l'étude du langage PHP permet également d'améliorer la dynamique des pages web.

### F.2. Les cookies

Les cookies sont très utilisés, par tous les sites commerciaux et par de plus en plus de sites personnels. La raison est simple : un cookie permet de stocker de manière permanente des informations sur le poste du visiteur, qui pourront être récupérées lors des futures visites.

Voyons quelques unes des principales informations stockées dans les cookies :

- Le nombre de visites, la date de la dernière visite ;
- Un identifiant et un mot de passe pour une reconnaissance automatique du visiteur ;
- Une liste de mots-clés utilisés dans les moteurs de recherche pour cibler les publicités à afficher (comme le font beaucoup de moteurs de recherche) ;
- Une liste de paramètres de préférences de navigation pour personnaliser la page présentée ;
- Des informations à transférer d'une page à l'autre du site.

Exemple de choix d'une couleur de fond d'écran :

L'utilisateur choisi un fond d'écran, qui est enregistré dans un cookie. Les autres pages du site lisent la couleur choisie dans le cookie et l'appliquent.

```
<body>
  <!-- Page de choix de la couleur de fond d'écran -->
  <form name="fond" id="fond">
    <fieldset>
      <legend>Choix d'une couleur :</legend>
      Je veux une couleur de fond
      <select name="couleur" id="couleur">
        <option value="#0000FF" selected="selected">bleue</option>
        <option value="#00FF00">verte</option>
        <option value="#FF0000">rouge</option>
      </select><br/>
      <input type="button" value="Valider votre choix" onclick="enregFond()" /><br/>
    </fieldset>
  </form>
</body>
```

```

/* Enregistrement d'un cookie */
function ecrireCookie(nomCook, valeurCook) {
    /* Détection de l'activation des cookies */
    if (navigator.cookieEnabled) {
        /* Création d'une date d'expiration du cookie, ici un an */
        var maintenant = new Date(), expiration = new Date();
        expiration.setTime(maintenant.getTime() + (365 * 24 * 60 * 60 * 1000));
        /* Ecriture du cookie = une rubrique (nomCook), une valeur (valeurCook) et une date
d'expiration */
        document.cookie = nomCook + '=' + encodeURIComponent(valeurCook) + ';expires=' +
expiration.toGMTString();
    }
    else {
        alert('Vous devez activer les cookies pour voir ce site');
    }
}

/* La fonction qui gère le fond d'écran */
function enregFond() {
    var fondChoisi;
    fondChoisi = document.getElementById('couleur').value;
    /* Application du fond choisi */
    document.body.style.backgroundColor = fondChoisi;
    /* Mémorisation du fond choisi dans un cookie */
    ecrireCookie('CouleurFond', fondChoisi);
}

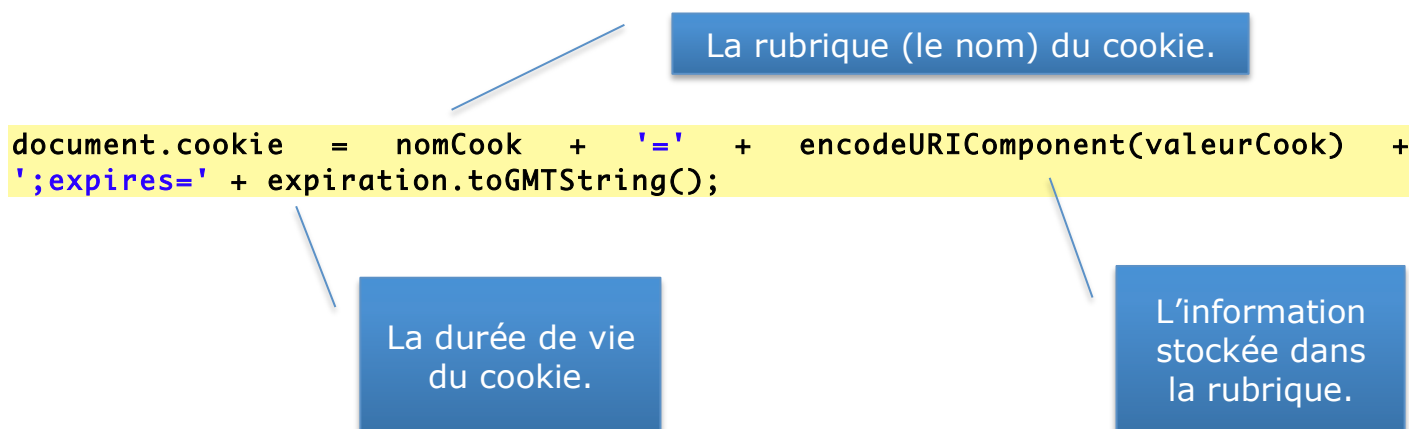
```

Remarque : La gestion des cookie nécessite deux fonctions :

- une pour écrire le cookie : `ecrireCookie()` ;
- une pour lire le cookie : `lireCookie()` (cf. plus bas) ;

La gestion globale du cookie est faite par `enregFond()` qui récupère le choix de l'utilisateur, l'applique à la page courante et l'enregistre dans un cookie sous la rubrique '`CouleurFond`'.

Un cookie se compose des éléments suivants :



Le fond d'écran est appliqué aux pages suivantes :

```
<body onload="recupFond()">
  <h1>HELLO</h1>
  <h2>Everybody !</h2>
</body>
```

```
/* Récupération d'un cookie */
function lireCookie(nomCook) {
  /* Détection de l'activation des cookies */
  if (navigator.cookieEnabled) {
    /* Avec une expression régulière on récupère la valeur recherchée */
    var oRegex = new RegExp('(?:; )?' + nomCook + '=[^;]*;?');
    if (oRegex.test(document.cookie)) {
      /* On retourne la valeur trouvée */
      return decodeURIComponent(RegExp['$1']);
    }
    else {
      /* La valeur n'a pas été trouvée */
      return null;
    }
  }
  else {
    alert('Vous devez activer les cookies pour voir ce site');
  }
}

function recupFond() {
  var fondChoisi;
  /* Récupération du fond choisi dans un cookie */
  fondChoisi = lireCookie('CouleurFond');
  /* Application du fond choisi (sans test du cas où il y aurait false) */
  document.body.style.backgroundColor = fondChoisi;
}
```

Remarque : La fonction `recupFond()` se charge de lire la valeur stockée dans le cookie pour la rubrique qui nous intéresse (ici `'CouleurFond'`) puis de l'appliquer à la page courante dès son chargement. Il faudrait prévoir un test avant d'affecter `fondChoisi` au cas où on ne retrouve pas le cookie.

## F.3. Gestion dynamique de la présentation des sites

### F.3.1. Les objets du DHTML

Nous allons nous attarder sur le dernier atout du DHTML qui concerne la gestion des documents Web.

En HTML, tous les éléments de la page sont considérés comme des objets de type différent. Les images, les liens, les formulaires, les champs de formulaires, etc. sont des objets que le JavaScript peut déjà manipuler.

Le DHTML introduit la notion de calque ou couche ou layer. Ces calques sont à la base du DHTML car ils vont pouvoir être manipulés : déplacés, cachés, modifiés, redessinés...

Un calque est défini par exemple par la balise `<div></div>`.

Un calque doit posséder :

- un identifiant, pour le repérer ;
- un style d'affichage, pour indiquer sa position, sa taille, sa couleur, etc. ;
- un contenu, qui sera affiché.

L'identifiant est enregistré grâce à l'attribut *id*. Sur une page, l'identifiant de tous les calques (et de tous les autres objets) doit être unique, pour éviter les ambiguïtés. Une erreur ne surviendra pas forcément en cas de doublons, mais le fonctionnement normal risque d'être perturbé.

### F.3.2. Manipuler les calques avec JavaScript

Le calque peut être accédé par JavaScript grâce à `document.getElementById()` qui retourne l'objet calque et que nous avons déjà rencontré.

L'objet de type calque ainsi retourné a comme propriété un nouvel objet style qui contient toutes les propriétés d'affichage du calque : visibilité, position, dimensions, couleurs... Cet objet style est souvent manipulé dans le DHTML.

Cette ligne montre le calque identifié comme « monCalque » :

```
document.getElementById('monCalque').style.visibility = 'visible';
```

Cette ligne cache le calque identifié comme « moncalque » :

```
document.getElementById('monCalque').style.visibility = 'hidden';
```

Pour déplacer le calque, on peut utiliser les propriétés `top` et `left` :

```
document.getElementById('monCalque').style.left = <nb pixel>;
```

Pour modifier la taille du calque :

```
document.getElementById('monCalque').style.width = <nb pixel>;
```

Attention, il n'est possible en JavaScript de modifier un attribut seulement si il a été défini lors de la création du calque.

### F.3.3. Manipulation sur le style

Pour changer la couleur de fond, on utilise l'attribut `backgroundColor` :

```
document.getElementById('monCalque').style.backgroundColor = '<couleur>';
```

On peut aussi changer la couleur du texte :

```
document.getElementById('monCalque').style.color = '<couleur>';
```

On peut attribuer/réattribuer une classe de style à un objet :

```
document.getElementById('monCalque').className = '<nom_classe>';
```

### F.3.4. Exemple DHTML sur un formulaire

Un très bon exemple d'utilisation du DHTML est la gestion des formulaires. On peut spécifier des informations importantes ou des erreurs grâce au couple Javascript/CSS.

Pour cela on peut choisir de cacher certains éléments et de les révéler suivant des événements, ou d'écrire le contenu des éléments selon les événements.

Un formulaire dispose de plusieurs balises cachées et de styles par défaut :

```
<body>
  <p>Votre inscription</p>
  <form id="f1" method="post" action="inscrire.php">
    <fieldset>
      <legend>Identité :</legend>
      Entrer votre nom <input onclick="newstyle('indic1','nom')" onblur="raz('indic1','nom')"
class="question" type="text" id="nom" name="nom" />
<span class="indication" id="indic1">Attention à la casse !</span>
<span class="indication" id="indic1b">Votre nom ne doit pas dépasser 10
caractères</span><br/><br/>
      Entrer votre prénom <input onclick="newstyle('indic2','prenom')"
onblur="raz('indic2','prenom')" class="question" type="text" id="prenom" name="prenom" />
<span class="indication" id="indic2">Attention à la casse !</span><br/><br/>
      <input name="b1" type="reset" value="effacer" /><input name="b2" type="button"
value="valider" onclick="controle()" />
    </fieldset>
  </form>
</body>
```

Une feuille de style contient les styles actifs de la page ainsi que d'autres, qui seront activés par Javascript.

```
body { font-family: verdana; }

input { font-size: 12pt; }

input.question { /* Style de départ d'une question */
  border: 1px solid orange;
}

input.saisie { /* Style des questions en cours de saisie */
  border: 1px solid orange;
  background-color: grey;
  color: #dddddd;
}

input.erreur { /* Visible si le nom saisi fait plus de 10 caractères */
  border: 2px solid red;
  color: red;
}

input.ok { /* Visible si le nom saisi fait moins de 10 caractères */
  border: 1px solid orange;
  background-color: lightblue;
  color: #888888;
}

span.indication { /* Indications des questions visible lors de la saisie */
  color: red;
  visibility:hidden;
}
```

Le script contient toutes les fonctions déclenchées :

```
var indication, elem;
/* Fonction qui change le style d'un élément et fait apparaître une indication masquée */
function newstyle(indication, elem)
{
  document.getElementById(elem).className = 'saisie';
  document.getElementById(indication).style.visibility = 'visible';
}

/* Fonction qui réinitialise le style d'un élément et masque l'indication */
function raz(indication, elem)
{
  document.getElementById(elem).className = 'question';
  document.getElementById(indication).style.visibility = 'hidden';
}

/* Fonction qui contrôle la longueur du nom */
function controle()
{
  var nomsaisi;
  nomsaisi=document.getElementById('nom').value;
  if (nomsaisi.length > 10)
  {
    document.getElementById('nom').className = 'erreur';
    document.getElementById('indic1b').style.visibility = 'visible';
  }
  else
  {
    document.getElementById('nom').className = 'ok';
    document.getElementById('indic1b').style.visibility = 'hidden';
  }
}
```

Lancement de la fonction `newstyle()` :

Votre inscription

Identité :

Entrer votre nom  Attention à la casse !

Entrer votre prénom

Lancement de la fonction `controle()` sans erreur :

Votre inscription

Identité :

Entrer votre nom

Entrer votre prénom

Lancement de la fonction `controle()` avec erreur :

Votre inscription

Identité :

Entrer votre nom  Votre nom ne doit pas dépasser 10 caractères

Entrer votre prénom

Mention JavaScript	Court descriptif
background	image d'arrière plan
backgroundAttachment	effet filigrane
backgroundColor	couleur d'arrière plan
backgroundImage	image d'arrière plan
backgroundPosition	Position de l'image d'arrière plan
backgroundRepeat	effet papier peint
border	bordures
borderBottom	bordure du bas
borderBottomColor	couleur de la bordure du bas
borderBottomStyle	style de la bordure du bas
borderBottomWidth	épaisseur de la bordure du bas
borderColor	couleur de la bordure du bas
borderLeft	bordure de gauche
borderLeftColor	couleur de la bordure de gauche
borderLeftStyle	style de la bordure de gauche
borderLeftWidth	épaisseur de la bordure de gauche
borderRight	bordure de droite
borderRightColor	couleur de la bordure de droite
borderRightStyle	style de la bordure de droite
borderRightWidth	épaisseur de la bordure de droite
borderStyle	style de la bordure
borderTop	bordure en haut
borderTopColor	couleur de la bordure en haut
borderTopStyle	style de la bordure en haut
borderTopWidth	épaisseur de la bordure en haut
borderWidth	épaisseur de la bordure
captionSide	légende de tableau
clear	suite pour le cours du texte
clip	limiter le domaine d'affichage
color	couleur du texte
cursor	pointeur de souris
direction	sens d'écriture
display	non-affichage (sans prendre de place)
emptyCells	représentation de cellules de tableau vides



Mention JavaScript	Court descriptif
float	flux de texte
font	police
fontFamily	famille de police
fontSize	taille de police
fontStretch	espacement des caractères de la police
fontStyle	style de police
fontVariant	variante de police
fontWeight	poids de police
height	hauteur d'un élément
left	position à partir de la gauche
letterSpacing	espacement des signes
lineHeight	hauteur de ligne
listStyle	représentation de listes
listStyleImage	graphiques de puces personnalisés
listStylePosition	retrait de listes
listStyleType	type de représentation de liste
margin	espace/marge
marginBottom	espace/marge en bas
marginLeft	espace/marge à gauche
marginRight	espace/marge à droite
marginTop	espace/marge en haut
maxHeight	hauteur maximale
maxWidth	largeur maximale
minHeight	hauteur minimale
minWidth	largeur minimale
overflow	contenu trop important
padding	espace intérieur
paddingBottom	espace intérieur en bas
paddingLeft	espace intérieur à gauche
paddingRight	espace intérieur à droite
paddingTop	espace intérieur en haut
pageBreakAfter	saut de page après
pageBreakBefore	saut de page avant
position	mode de positionnement
right	position à partir de la droite

Mention JavaScript	Court descriptif
scrollbar3dLightColor	couleur pour effet relief (Scrollbars)
scrollbarArrowColor	couleur pour les pointeurs de défilement (Scrollbars)
scrollbarBaseColor	couleur de base des barres de défilement (Scrollbars)
scrollbarDarkshadowColor	couleur pour les ombres (Scrollbars)
scrollbarFaceColor	couleur pour la surface (Scrollbars)
scrollbarHighlightColor	couleur pour le bord haut et le bord gauche (Scrollbars)
scrollbarShadowColor	= couleur pour le bord droit et le bord du bas (Scrollbars)
scrollbarTrackColor	= couleur pour la barre de défilement non-cachée par le pointeur de défilement (Scrollbars)
tableLayout	type de tableau
textAlign	alignement
textDecoration	décoration du texte
textIndent	retrait du texte
textTransform	transformation du texte
top	position à partir du haut
verticalAlign	alignement vertical
visibility	non affichage avec réservation de place
width	largeur d'un élément
wordSpacing	espacement des mots
zIndex	position de la couche en cas de superposition

Instruction	Utilisation
<code>array()</code>	Tableau, déclaré : <ul style="list-style-type: none"> <li>- <code>monTableau = new Array(n)</code> (n la longueur du tableau) ;</li> <li>- <code>monTableau[0]</code> est le premier élément de ce tableau ;</li> <li>- <code>monTableau = new Array('cerise', 'tomate', 'stylo')</code>.</li> </ul>
<code>charAt(x)</code>	Livre le caractère à la x-ième position de la chaîne de caractères (qui débute à la position 0). <ul style="list-style-type: none"> <li>- <code>variable = phrase.charAt(x)</code>.</li> </ul>
<code>confirm()</code>	Boite de dialogue de confirmation qui renvoi la valeur true si 'Ok' est pressé, false si annuler est pressé.
<code>date()</code>	Objet date, déclaré : <ul style="list-style-type: none"> <li>- <code>maDate = new Date()</code>.</li> </ul>
<code>document.write('Texte')</code>	Permet d'écrire dans la page web.
<code>getDate()</code>	Renseigne sur le jour du mois sous la forme d'un nombre compris entre 1 et 31 : <ul style="list-style-type: none"> <li>- <code>var date;</code></li> <li>- <code>maDate = new Date()</code> ;</li> <li>- <code>date = maDate.getDate()</code>;</li> </ul>
<code>getDay()</code>	Renseigne sur le jour de la semaine sous la forme d'un nombre compris entre 0 et 6 (0 pour dimanche).
<code>getFullYear()</code>	Renseigne sur l'année sous forme d'un nombre à quatre chiffres.
<code>getHours()</code>	Informe sur les heures.
<code>getMinutes()</code>	Informe sur les minutes.
<code>getMonth()</code>	Renseigne sur le mois sous la forme d'un nombre compris entre 0 et 11 (0 pour janvier).
<code>getSeconds()</code>	Informe sur les secondes
<code>getYear()</code>	Renseigne sur l'année.
<code>image()</code>	Objet image, déclaré de la façon suivante : <ul style="list-style-type: none"> <li>- <code>monImage = new Image()</code>;</li> </ul>
<code>length</code>	Détermine la longueur d'une chaîne de caractères (ou d'un tableau). <ul style="list-style-type: none"> <li>- <code>longueur = phrase.length</code>.</li> </ul>
<code>location.href='adresse url'</code>	Indication d'une URL vers une adresse.
<code>math.random()</code>	Génère des nombres pseudo aléatoires : <ul style="list-style-type: none"> <li>- <code>math.random()*(b-a)+a</code> (b le plus grand nombre désiré, a le plus petit nombre désiré).</li> </ul>
<code>math.round</code>	Arrondi un nombre : <ul style="list-style-type: none"> <li>- <code>math.round(1.5)</code> est égal à 2.</li> </ul>
<code>navigator.appName</code>	Nom du navigateur.
<code>navigator.langage</code>	Langue (uniquement pour Netscape).
<code>navigator.platform</code>	Système d'exploitation.
<code>navigator.userAgent</code>	Désignation précise du navigateur.

Instruction	Utilisation
<code>parent.location.href='url'</code>	Assigne (ou Donne) l'adresse de l'URL parente.
<code>screen.availheight</code>	Hauteur visible de l'écran en pixels.
<code>screen.availwidth</code>	Largeur visible de l'écran en pixels.
<code>screen.colorDepth</code>	Profondeur en couleurs (8, 16, 32 bits).
<code>Screen.height</code>	Hauteur de l'écran en pixels.
<code>screen.width</code>	Largeur de l'écran en pixels.
<code>setTimeout('Action', millisecondes)</code>	Permet d'exécuter une action avec retardement.
<code>split('; ')</code>	Cette méthode convertit une chaîne de caractères en un tableau. Le ; est le séparateur choisi ici entre les différents éléments du tableau.
<code>src</code>	Spécifie l'URL de la source d'un objet : - <code>document.img.src=...</code>
<code>substring(position1, position2)</code>	Affiche les caractères entre deux positions d'une phrase.
<code>toGMTString()</code>	La date est transformée en une chaîne de caractères correspondant à la norme IETF. Le 23 février 2002 deviendra : Sat, 23 feb 2002 12 :24 :23 UTC.
<code>toLowerCase()</code>	Convertit une chaîne de caractères en minuscules.
<code>toUpperCase()</code>	Convertit une chaîne de caractères en majuscules.
<code>variable=prompt('texte', 'valeur par défaut')</code>	Permet d'ouvrir une boîte de dialogue d'entrée dont la valeur sera affectée à variable.
<code>window.close()</code>	Fermeture d'une fenêtre.
<code>window.open('adresse url')</code>	Permet d'ouvrir une nouvelle fenêtre.
<code>Window.opener.location.href='url'</code>	Cela permet de manipuler la fenêtre qui a créé (ouvert) la fenêtre appelée.
<code>window.status</code>	Texte de la ligne d'état d'une fenêtre Windows.