Ecole Industrielle de Namur 2001 - 2002

Cours d'Initiation au langage JavaScript

Y. Mine

Tables des matières

Chapitre 1 : Introduction

1.1 Qu'est-ce que le JavaScript?

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

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

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

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

Javascript n'est donc pas propre aux navigateurs de Netscape. Microsoft l'a aussi adopté à partir de Internet Explorer 3. On le retrouve, de façon améliorée, dans Explorer 4.

Les versions de Javascript se sont succédées avec les différentes versions de Netscape : Javascript pour Netscape 2, Javascript 1.1 pour Netscape 3 et Javascript 1.2 pour Netscape 4. Ce qui n'est pas sans poser certains problèmes de compatibilité, selon le browser utilisé, des pages comportant du code Javascript.

Différences entre JavaScript et Java.

JavaScript	Java
Code intégré dans la page HTML.	Code non intégré dans la page HTML mais dans une applet.
Code interprété par le navigateur au moment de l'exécution.	Code source compilé avant son exécution.
Code de programmation simple mais limité.	Code de programmation complexe mais plus étendu.
Accès aux objets du navigateur.	Pas d'accès aux objets du navigateur.
Confidentialité des codes nulle (code source visible)	Sécurité (code source compilé)

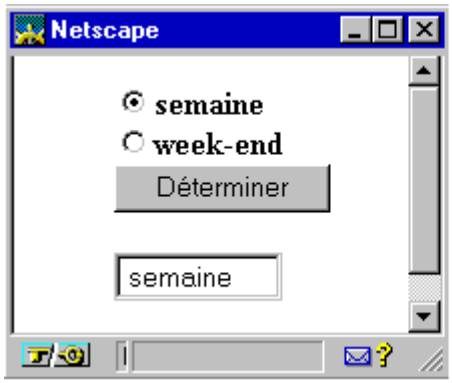
- Javascript est plus simple à mettre en oeuvre car c'est du code que vous ajouterez à votre page écrite en Html. Java pour sa part, nécessite une compilation préalable de votre code.
- Le champ d'application de Javascript est somme toute assez limité alors qu'en Java vous pourrez en principe tout faire.
- Comme votre code Javascript est inclus dans votre page Html, celui-ci est visible et peut être copié par tout le monde (view source). Par contre, en Java, votre code source est broyé par le compilateur et est ainsi indéchiffrable.
- Les codes Javascript ne ralentissent pas le chargement de la page alors que l'appel à une applet Java peut demander quelques minutes de patience supplémentaire à votre lecteur.

1.2 Théorie objet

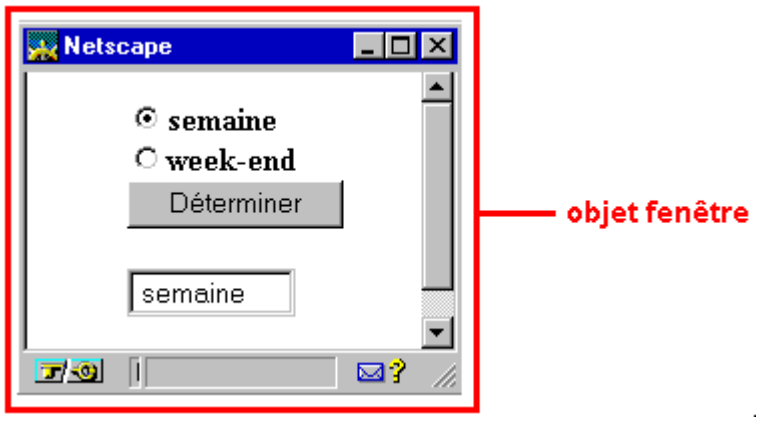
Les objets et leur hiérarchie

L'internaute voit une page Web sur son écran.
Javascript va diviser cette page en objets et surtout va permettre d'accéder à ces objets, d'en retirer des informations et de les manipuler.

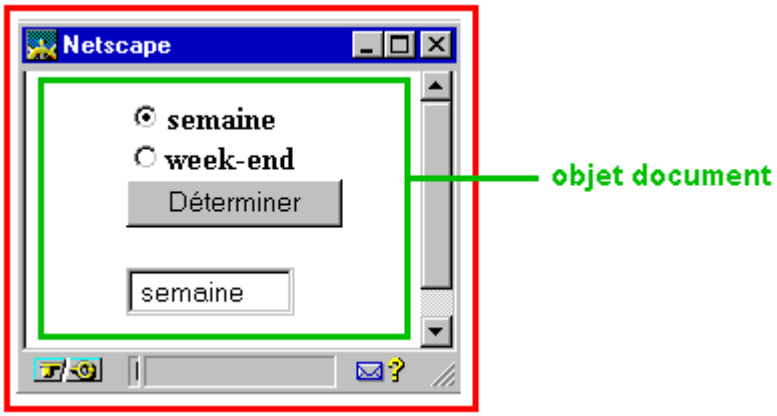
Voici une illustration des différents objets qu'une page peut contenir.



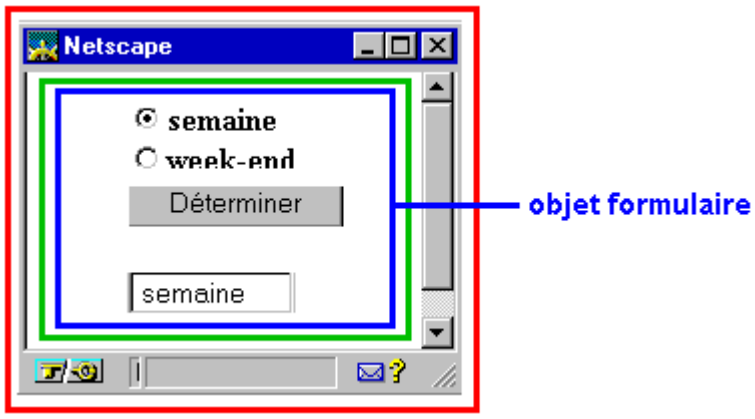
Cette page s'affiche dans une fenêtre. C'est l'objet fenêtre.



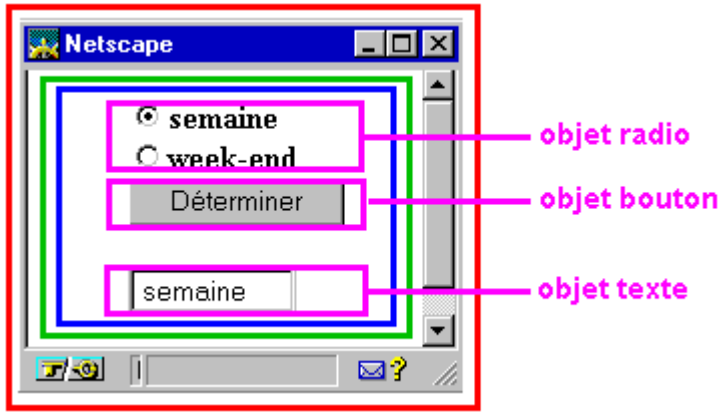
Dans cette fenêtre, il y a un document Html. C'est l'objet document. Autrement dit (et c'est là que l'on voit apparaître la notion de la hiérarchie des objets Javacript), l'objet fenêtre contient l'objet document.



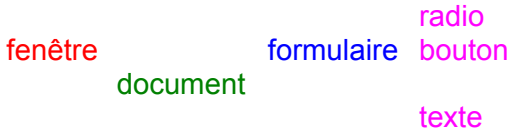
Dans ce document, on trouve un formulaire au sens Html. C'est l'objet formulaire. Autrement dit, l'objet fenêtre contient un objet document qui lui contient un objet formulaire.



Dans ce document, on trouve trois objets. Des boutons radio, un bouton classique et une zone de texte. Ce sont respectivement l'objet radio, l'objet bouton, l'objet texte. Autrement dit l'objet fenêtre contient l'objet document qui contient l'objet formulaire qui contient à son tour l'objet radio, l'objet fenêtre contient l'objet document qui contient l'objet formulaire qui contient à son tour l'objet bouton et l'objet fenêtre contient l'objet document qui contient l'objet formulaire qui contient à son tour l'objet texte.



La hiérarchie des objets de cet exemple est donc



Pour accéder à un objet, il faudra donner le chemin complet de l'objet en allant du contenant le plus extérieur à l'objet référencé.
Soit par exemple pour le bouton radio "semaine" : (window).document.form.radio[0].
L'objet window est entre parenthèses car comme il occupe la première place dans la hiérarchie, il est repris par défaut par Javascript et devient donc facultatif.

1.3 Les propriétés des objets

Une propriété est un attribut, une caractéristique, une description de l'objet. Par exemple, l'objet volant d'une voiture a comme propriétés qu'il peut être en bois ou en cuir. L'objet livre a comme propriétés son auteur, sa maison d'édition, son titre, son numéro ISBN, etc.

De même les objets Javascript ont des propriétés personnalisées. Dans le cas des boutons radio, une de ses propriétés est, par exemple, sa sélection ou sa non-sélection (checked en anglais).

En Javascript, pour accéder aux propriétés, on utilise la syntaxe :
nom_de_l'objet.nom_de_la_propriété

Dans le cas du bouton radio "semaine", pour tester la propriété de sélection, on écrira
document.form.radio[0].checked

1.4 Le Javascript minimum

La balise <SCRIPT>

Le script vient s'ajouter aux pages Web.
Le langage Html utilise des tags ou balises pour "dire" au browser d'afficher une portion de texte en gras, en italique, etc.
Dans la logique du langage Html, il faut donc signaler au browser par une balise, que ce qui suit est un script et que c'est du Javascript (et non du VBScript). C'est la balise
<SCRIPT LANGUAGE="Javascript">.
De même, il faudra informer le browser de la fin du script.
C'est la balise </SCRIPT>.

Les commentaires

Il vous sera peut-être utile d'inclure des commentaires personnels dans vos codes Javascript. C'est même vivement recommandé comme pour tous les langages de programmation.

Javascript utilise les conventions utilisées en C et C++ soit
// commentaire
Tout ce qui est écrit entre le // et la fin de la ligne sera ignoré.

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

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

Masquer le script pour les anciens browsers

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

Votre premier Javasript ressemblera à ceci :
<SCRIPT LANGUAGE="javascript">
<!-- Masquer le script pour les anciens browsers
...
programme Javascript
...
// Cesser de masquer le script -->
</SCRIPT>

1.5 Où placer le JavaScript ?

Vous devez placer le script entre les tags **<SCRIPT LANGUAGE="JavaScript" TYPE="text/javascript">** et **</SCRIPT>**.

Il faut remarquer que le tag TYPE="text/javascript" n'est pas obligatoire.
Tout ce qui sera entre ces tags sera reconnu comme du JavaScript. Si vous mettez du code HTML, cela provoquera une erreur.

Vous pouvez placer le script entre les tags **<HEAD> </HEAD>** et **<BODY> </BODY>**.

```
<html>
<head>
<title>Ma page</title>
</head>
<body>
<script LANGUAGE="JavaScript">
<!--
alert('Bienvenue sur ma page');
// -->
</script>
</body>
</html>
```

Quelques remarques :

1. Rappel : <!-- et // --> sont des commentaires. Ils servent uniquement si votre navigateur de reconnaît pas le JavaScript, ainsi le code du script ne sera pas affiché.
2. Les lignes du script se terminent généralement par un point-virgule (;). Il n'est obligatoire que si vous mettez plusieurs instructions sur une même ligne, ceci afin que le browser puisse voir où se termine l'instruction.
3. Il faut mettre le texte entre " " ou entre ' ' sinon il sera considéré comme une variable.
   Il vaut mieux utiliser ' ' lorsque vous utilisez du texte directement lors du traitement d'un

événement sans faire appel à une fonction qui effectue ce traitement. C'est nécessaire car si vous mettez " ", il va considérer la fin du traitement de l'événement au deuxième "

4. Les commentaires :
   // : commentaire sur une seul ligne.
   /* ... */ : commentaires qui peuvent être mis sur plusieurs lignes.

Exemple 1 : bonne façon de faire.

```
<html>
<head>
<title>Ma page</title>
</head>
<body>
<a href="#" onClick="alert('message');">cliquez ici</a>
<a href="javascript:alert('message');">cliquez ici</a>
</body>
</html>
```

Exemple 2 : bonne façon de faire.

```
<html>
<head>
<title>Ma page</title>
<script LANGUAGE="JavaScript">
<!--
function message(){
alert("message");
alert('autre message');
//ici on peut utiliser " " ou ' '
}
// -->
</script>
</head>
<body>
<a href="#" onClick="message();">cliquez ici</a>
</body>
</html>
```

1.6 Les différentes versions de JavaScript

Selon que vous voulez utiliser l'une ou l'autre version de JavaScript, vous devrez mettre des balises différentes.

Version	Compatibilité	Balise
JavaScript 1.0	I.Explorer 3.0 - Netscape 2.0	<SCRIPT LANGUAGE="JavaScript1.0">
JavaScript 1.1	Netscape 3.0	<SCRIPT LANGUAGE="JavaScript1.1">
JavaScript 1.2	I.Explorer 4.x - Netscape 4.x	<SCRIPT LANGUAGE="JavaScript1.2">
JavaScript 1.3	I.Explorer 4.x,5.0 - Netscape 4.x	<SCRIPT LANGUAGE="JavaScript1.3">

Le JavaScript fait la différence entre les majuscules et les minuscules (case sensitive). Il est donc important de faire attention lorsque vous appelez une fonction ou un variable. Certaines méthodes contiennent des majuscules, vous devez respecter ces majuscules sinon la méthode ne sera pas reconnue

Ainsi, il faudra écrire alert() et non Alert(). Pour l'écriture des instructions Javascript, on utilisera l'alphabet ASCII classique (à 128 caractères) comme en Html. Les caractères accentués comme é ou à ne peuvent être employés que dans les chaînes de caractères c.-à-d. dans 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("...') donnera un message d'erreur. Si vous souhaiter utiliser des guillemets dans vos chaînes de caractères, tapez \" ou \' pour les différencier vis à vis du compilateur.

Extension .js pour scripts externes

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

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

1.7 Les commentaires

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

Les commentaires ne peuvent être imbriqués sous peine de message d'erreur. La formulation suivante est donc à éviter :
/* script réalisé ce jour /* jour mois */
et testé par nos soins*/

Alert() ... rouge

Joujou des débutants en Javascript, cette petite fenêtre est à utiliser avec parcimonie pour attirer l'attention du lecteur pour des choses vraiment importantes. Javascript met à votre disposition la possibilité de créer de nouvelles fenêtres de la dimension de votre choix qui apparaissent un peu comme les popup des fichiers d'aide.

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

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

Chapitre 2 : Afficher du texte

2.1 Méthode de l'objet document

Ce qui apparaît sur l'écran, peut être "découpé" en objets et que Javascript peut donner la possibilité d'accéder à ces objets. La page Html qui s'affiche dans la fenêtre du browser est un objet de type document.

A chaque objet Javascript, le concepteur du langage a prévu un ensemble de méthodes (ou fonctions dédiées à cet objet) qui lui sont propres. A la méthode document, Javascript a dédié la méthode "écrire dans le document", c'est la méthode write().

L'appel de la méthode se fait selon la notation :
nom_de_l'objet.nom_de_la_méthode

Pour appeler la méthode write() du document, on notera
document.write();

2.2 La méthode write()

La syntaxe est assez simple soit
write("votre texte");

On peut aussi écrire une variable, soit la variable resultat,
write(resultat);

Pour associer du texte (chaînes de caractères) et des variables, on utilise l'instruction write("Le résultat est " + resultat);

On peut utiliser les balises Html pour agrémenter ce texte
write("<B>Le résultat est</B>" + resultat); ou
write ("<B>" + "Le résultat est " + "</B>" + resultat)

Exemple

On va écrire du texte en Html et en Javascript.

```
<HTML>
<BODY>
<H1>Ceci est du Html</H1>
<SCRIPT LANGUAGE="Javascript">
<!--
document.write("<H1>Et ceci du Javascript</H1>");
//-->
</SCRIPT>
</BODY>
</HTML>
```

Ce qui donnera comme résultat :

Ceci est du Html
Et ceci du Javascript

2.3 En bref, les instructions de formatage de document

Ce qui suit est optionnel et vous pouvez utiliser l'instruction document.write() de façon tout à fait classique.
Soit document.write("<BODY BGCOLOR="#FFFFFF");

document.bgColor

Cette instruction permet de spécifier la couleur d'arrière-plan d'un objet document. On peut employer le nom ou la valeur RGB de la couleur.

document.bgColor="white";
document.bgColor="#FFFFFF";

document.fgColor

Cette instruction permet de spécifier la couleur d'avant-plan (texte) d'un objet document. On peut employer le nom ou la valeur RGB de la couleur.

document.fgColor="black";
document.fgColor="#000000";

document.alinkColor

Cette instruction permet de spécifier la couleur d'un lien actif (après le clic de la souris mais avant de quitter le lien) d'un objet document. On peut employer le nom ou la valeur RGB de la couleur.

document.alinkColor="white";
document.alinkColor="#FFFFFF";

document.linkColor

Cette instruction permet de spécifier la couleur d'un hyperlien d'un objet document. On peut employer le nom ou la valeur RGB de la couleur.

document.linkColor="white";
document.linkColor="#FFFFFF";

document.vlinkColor

Cette instruction permet de spécifier la couleur d'un hyperlien déjà visité d'un objet document. On peut employer le nom ou la valeur RGB de la couleur.

document.linkColor="white";
document.linkColor="#FFFFFF";

Chapitre 3 Les opérateurs

3.1 Opérateurs arithmétiques

+	Additionne 2 nombres.
++	Incrémente un nombre d'une unité.
-	Soustrait 2 nombres ou donne la valeur négative d'un nombre.
--	Soustrait un nombre d'une unité.
*	Multiplie 2 nombres.
/	Divise 2 nombres.
%	Modulo (calcule le reste de la division entière de 2 nombres).

Dans les exemples, la valeur initiale de x sera toujours égale à 11

Signe	Nom	Signification	Exemple	Résultat
+	plus	addition	x + 3	14
-	moins	soustraction	x - 3	8
*	multiplié par	multiplication	x*2	22
/	divisé	par division	x /2	5.5
%	modulo	reste de la division par	x%5	1
=	a la valeur	affectation	x=5	5

3.2 Opérateurs bit à bit

&	Opérateur ET (AND) binaire. 1 & 1 = 1 1 & 0 = 0 0 & 0 = 0
^	Opérateur OU excelsuif (XOR) binaire. 1 ^ 1 = 0 1 ^ 0 = 1 0 ^ 0 = 0
|	Opérateur OU (OR) binaire. 1 | 1 = 1 1 | 0 = 1 0 | 0 = 0
~	Opérateur de négation (NOT) : inverse les bits de l'opérande.
<<	Left shift. a << n décale a en représentation binaire de n bits vers la gauche.

>>	Right shift. a >> n décale a en représentation binaire de n bits vers la droite.
>>>	Zero-fill right shift. a >>> n décale a en représentation binaire de n bits vers la droite en rajoutant des 0 à gauche.
><	a >< b. Les valeurs de a et de b sont interchangées.

3.3 Opérateurs d'assignation

=	Assigne la valeur de l'opérande de droite à celui de gauche.
+=	Additionne le nombre à gauche par la valeur de l'opérande de droite.
-=	Soustrait le nombre à gauche par la valeur de l'opérande de droite.
*=	Multiplie le nombre à gauche par la valeur de l'opérande de droite.
/=	Divise le nombre à gauche par la valeur de l'opérande de droite.
%=	Calcule le modulo du nombre de gauche par le nombre de droite.
&=	Calcule le AND des 2 nombres et l'assigne au nombre de gauche.
^=	Calcule le XOR des 2 nombres et l'assigne au nombre de gauche.
|=	Calcule le OR des 2 nombres et l'assigne au nombre de gauche.
<<=	Calcule le "left shift" des 2 nombres et l'assigne au nombre de gauche.
>>=	Calcule le "right shift" des 2 nombres et l'assigne au nombre de gauche.
>>>=	Calcule le "Zero-fill right shift" des 2 nombres et l'assigne au nombre de gauche.

Dans les exemples suivants x vaut toujours 11 et y aura comme valeur 5.

Signe	Description	Exemple	Signification	Résultat
+=	plus égal	x += y	x = x + y	16
-=	moins égal	x -= y	x = x - y	6
*=	multiplié égal	x *= y	x = x * y	55
/=	divisé égal	x /= y	x = x / y	2.2

3.4 Opérateurs de comparaison

==	Renvoie true si les opérandes sont égaux.
!=	Renvoie true si les opérandes ne sont pas égaux.
===	Renvoie true si les opérandes sont égaux et du même type.
!==	Renvoie true si les opérandes ne sont pas égaux et/ou ne sont pas du même type.
>	Renvoie true l'opérande de gauche est supérieure à celui de droite.
>=	Renvoie true l'opérande de gauche est supérieure ou égal à celui de droite.
<	Renvoie true l'opérande de gauche est inférieur à celui de droite.
<=	Renvoie true l'opérande de gauche est inférieur ou égal à celui de droite.

Signe	Nom	Exemple	Résultat
==	égal	x==11	true
<	inférieur	x<11	false
<=	inférieur ou égal	x<=11	true
>	supérieur	x>11	false

=<	supérieur ou égal	x>=11	true
!=	différent	x!=11	false

Important. On confond souvent le = et le == (deux signes =). Le = est un opérateur d'attribution de valeur tandis que le == est un opérateur de comparaison. Cette confusion est une source classique d'erreur de programmation.

3.5 Opérateurs de string

+	Concatène 2 strings.
+=	Concatène la string de droite à celle de gauche.

3.6 Opérateurs logiques

&&	(ET logique) Renvoie true lorsque les 2 opérandes sont vrais, sinon renvoie false.
	ex.:
	x = 1;
	y = 2;
	if ((x < 10) && (y > 0)) { vrai}
	ex.:
	x = 1;
	y = 2;
	if ((x < 0) && (y > 0)) { faux}

||	(OU logique) Renvoie true si un des 2 opérandes est vrai, sinon renvoie false.
!	(négation logique) Renvoie false si l'opérande peut être convertit en true, sinon renvoie false.

Aussi appelés opérateurs booléens, servent à vérifier deux ou plusieurs conditions.

Signe	Nom	Exemple	Signification
&&	et	(condition1) && (condition2)	condition1 et condition2
||	ou	(condition1) || (condition2)	condition1 ou condition2

3.7 Opérateurs spéciaux

?:	Effectue un "if ... then ... else"
	ex.:
	a = 10;
	b = 5;
	x = (a > 5? a:a+b)
	C'est équivalent à
	if (a > 5)
	x = a;
	else
	x = a + b;

3.8 Les opérateurs d'incrémentation

Ces opérateurs vont augmenter ou diminuer la valeur de la variable d'une unité. Ce qui sera fort utile, par exemple, pour mettre en place des boucles.
Dans les exemples, x vaut 3.

Signe	Description	Exemple	Signification	Résultat
x++	incrémentation (x++ est le même que x=x+1)	y = x++	3 puis plus 1	4
x--	décrémentation (x-- est le même que x=x-1)	y= x--	3 puis moins 1	2

3.9 La priorité des opérateurs Javascript

Les opérateurs s'effectuent dans l'ordre suivant de priorité (du degré de priorité le plus faible ou degré de priorité le plus élevé).
Dans le cas d'opérateurs de priorité égale, de gauche à droite.

Opération	Opérateur
,	virgule ou séparateur de liste
= += -= *= /= %=	affectation
? :	opérateur conditionnel
||	ou logique
&&	et logique
== !=	égalité
< <= >= >	relationnel
+ -	addition soustraction
* /	multiplier diviser
! - ++ --	unaire
( )	parenthèses

Chapitre 4 Les structures de contrôle

Ce chapitre reprend toutes les instructions pour les boucles, les conditions, ...

4.1 Les conditions

L'expression if

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

Dans sa formulation la plus simple, l'expression if se présente comme suit
if (condition vraie) {
une ou plusieurs instructions;
}

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

De façon un peu plus évoluée, il y a l'expression if...else

```
if (condition vraie) {
instructions1;
}
else {
instructions2;
}
```

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

Dans le cas où il n'y a qu'une instruction, les accolades sont facultatives.

Grâce aux opérateurs logiques "et" et "ou", l'expression de test pourra tester une association de conditions. Ainsi if ((condition1) && (condition2)), testera si la condition 1 et la condition 2 est réalisée. Et if ((condition1) || (condition2)), testera si une au moins des conditions est vérifiée.

Pour être complet (et pour ceux qui aiment les écritures concises), il y a aussi :

(expression) ? instruction a : instruction b

Si l'expression entre parenthèse est vraie, l'instruction a est exécutée.Si l'expression entre parenthèses retourne faux, c'est l'instruction b qui est exécutée.

L'expression for

L'expression for permet d'exécuter un bloc d'instructions un certain nombre de fois en fonction de la réalisation d'un certain critère. Sa syntaxe est :

```
for (valeur initiale ; condition ; progression) {
instructions;
}
```

Prenons un exemple concret

```
for (i=0, i<10, i++) {
document.write(i + "<BR>")
}
```

Au premier passage, la variable i, étant initialisée à 0, vaut bien entendu 0. Elle est bien inférieure à 10. Elle est donc incrémentée d'une unité par l'opérateur d'incrémentation i++ (i vaut alors 2) et les instructions s'exécutent.
A la fin de l'exécution des instructions, on revient au compteur. La variable i (qui vaut 2) est encore toujours inférieure à 10. Elle est augmentée de 1 et les instructions sont à nouveau exécutées. Ainsi de suite jusqu'à ce que i vaille 10. La variable i ne remplit plus la condition i<10. La boucle s'interrompt et le programme continue après l'accolade de fermeture.

L'expression While

L'instruction while permet d'exécuter une instruction (ou un groupe d'instructions) un certain nombre de fois.

```
while (condition vraie){
continuer à faire quelque chose
}
```

Aussi longtemps que la condition est vérifiée, Javascript continue à exécuter les instructions entre les accolades. Une fois que la condition n'est plus vérifiée, la boucle est interrompue et on continue le script.

Prenons un exemple.

```
compt=1;
while (compt<5) {
document.write ("ligne : " + compt + "<BR>");
compt++;
}
document.write("fin de la boucle");
```

Voyons comment fonctionne cet exemple. D'abord la variable qui nous servira de compteur compt est initialisée à 1. La boucle while démarre donc avec la valeur 1 qui est inférieure à 5. La condition est vérifiée. On exécute les instructions des accolades. D'abord, "ligne : 1" est affichée et ensuite le compteur est incrémenté de 1 et prend donc la valeur 2. La condition est encore vérifiée. Les instructions entre les accolades sont exécutées. Et ce jusqu'à l'affichage de la ligne 4. Là, le compteur après l'incrémentation vaut 5. La condition n'étant plus vérifiée, on continue dans le script et c'est alors fin de boucle qui est affiché.

Attention ! Avec ce système de boucle, le risque existe (si la condition est toujours vérifiée), de faire boucler indéfiniment l'instruction. Ce qui à la longue fait misérablement planter le browser !

Ce qui donnerait à l'écran :

```
ligne : 1
ligne : 2
ligne : 3
ligne : 4
fin de la boucle
```

L'expression Break

L'instruction break permet d'interrompre prématurément une boucle for ou while.

Pour illustrer ceci, reprenons notre exemple :

```
compt=1;
while (compt<5) {
if (compt == 4)
break;
document.write ("ligne : " + compt + "<BR>");
compt++;
}
document.write("fin de la boucle");
```

Le fonctionnement est semblable à l'exemple précédent sauf lorsque le compteur vaut 4. A ce moment, par le break, on sort de la boucle et "fin de boucle" est affiché.

Ce qui donnerait à l'écran :

```
ligne : 1
ligne : 2
ligne : 3
fin de la boucle
```

L'expression Continue

L'instruction continue permet de sauter une instruction dans une boucle for ou while et de continuer ensuite le bouclage (sans sortir de celui-ci comme le fait break).

Reprenons notre exemple ;

```
compt=1;
while (compt<5) {
if (compt == 3){
compt++
continue;}
document.write ("ligne : " + compt + "<BR>");
compt++;
}
document.write("fin de la boucle");
```

Ici, la boucle démarre. Lorsque le compteur vaut 3, par l'instruction continue, on saute l'instruction document.write (ligne : 3 n'est pas affichée) et on continue la boucle. Notons qu'on a dû ajouter compt++ avant continue; pour éviter un bouclage infini et un plantage du navigateur (compt restant à 3).

Ce qui fait à l'écran :

```
ligne : 1
ligne : 2
ligne : 4
fin de la boucle
```

break	Permet de quitter la boucle while, for ou un switch et donne le contrôle aux instructions suivant les boucles
	Syntaxe : break [label]
	Paramètre : label : Identifie un label
	ex.:
	i = 0
	while (i < 10){
	if (i == 3) break;
	i++;
	}
	//le programme continue ici après le break
	continue, label, switch

comment	Les commentaires sont du texte mis par l'auteur. Ils sont ignorés par l'interpréteur. // commentaire sur une ligne /* commentaire sur plusieurs lignes */

continue	Termine l'exécution du bloc d'une boucle while ou for et continue l'exécution à la prochaine itération.
	Syntaxe : continue [label]
	Paramètre : label : Identifie un label
	ex.:
	i = 0;
	while (i < 4){
	i++;
	if (i==2) continue;
	document.write(i + " - ");
	}
	Cela donne : 1 - 3 - 4 -
	break, label

do...while	Exécute les instructions tant que la condition de fin est false. Les instructions sont exécutées au moins 1 fois.
	Syntaxe :
	do{
	code
	}while (condition);
	Paramètre : condition : la condition de sortie de la boucle.
	ex.:
	i=0;
	do{
	i++;
	document.write(i);
	} while (i<10);

for	Crée une boucle qui exécute le code x fois. Attention aux boucles infinies : faites attention aux conditions de sortie.
	for(; ;) est une boucle infinie.
	Syntaxe :
	for ([valeur initiale] ; [condition] ; [incrément]) {
	code
	}
	Paramètre :
	• valeur initiale : valeur initiale de la variable compteur.
	• condition : condition de sortie de la boucle.

	• incrément : pour incrémenter ou décrémenter le compteur.
	ex.:
	for(i = 0 ; i < 10 ; i++){
	document.write(i);
	}
	Cela donne : 0123456789
	ex.:
	for(i = 0 ,j = 0; i < 10, j < 5 ; i++, j++){
	document.write(i);
	}
	Cela donne : 01234

for...in	Exécute une boucle avec une variable sur les propriétés d'un objet.
	Syntaxe :
	for (variable dans l'objet) {
	code
	}
	Paramètres :
	• variable : variable qui contiendra chaques propriétés de l'objet.
	• objet : un objet.
	ex.:
	var result = ""
	var jours = new
	Array("Lundi","Mardi","Mercredi","Jeudi","Vendredi","Samedi","Dimanche");
	for (var i in jours) {
	result += "jours." + i + " = " + jours[i] + "\n"
	}
	document.write(result);
	Cela donne :
	jours.0 = Lundi jours.1 = Mardi jours.2 = Mercredi jours.3 = Jeudi jours.4 = Vendredi jours.5 = Samedi jours.6 = Dimanche

function	Déclare une fonction avec ou sans paramètres. Les paramètres peuvent être des string, des nombres ou des objets.
	Syntaxe :
	function nom([param1] [,param2] [..., paramN]) {
	code
	}
	ex.:
	function carre(x) {
	return x * x;
	}

if...else	Exécute des instructions si la condition est true, sinon exécute d'autres instructions.
	Syntaxe :
	if (condition) {
	code
	}[else {
	code
	}]
	ex.:
	if (x > 3) {
	y += 2
	}

label	Fournit un endroit dans le code qui peut être référencé ailleurs dans le code. **Syntaxe** : label :

	nom_label break, continue

return	Spécifie la valeur de retour d'une fonction.
	Syntaxe : return expression
	Paramètre : expression : l'expression à renvoyer.
	ex.:
	function carre(x) {
	return x * x; //renvoie le carré de x
	}

switch	Permet au programme d'évaluer une expression et de la comparer à une valeur d'un "case label".
	Syntaxe :
	switch (expression) {
	case label :
	code;
	break;
	case label :
	code;
	break;
	...
	default : code;
	}
	Le break est obligatoire afin de permettre à l'interpréteur de savoir où se trouve la fin du code d'un "case".
	ex.:
	switch (langue) {
	case "français" :
	document.write("Bonjour");
	break;
	case "english" :
	document.write("Hello");
	break;
	default : document.write("Another language");
	}

throw	Lance une exception définie par l'utilisateur.
	Une exception est une erreur d'exécution détectée par le programme et lancée par celui-ci.
	Syntaxe : trhow epression
	Paramètre : expression : la valeur à lancer
	ex.:
	function Exception(message) {
	this.message = message
	this.nom = "Exception utilisateur"
	}
	function nomJours(n) {
	var jours = new
	Array("Dimanche","Lundi","Mardi","Mercredi","Jeudi","Vendredi","Samedi")
	if (n != null) {
	return jours[n]
	} else {
	monException = new Exception("Jours invalide")
	throw monException
	}
	}

	try {
	jours = nomJours(n)
	}
	catch (e) {
	jours = "Inconnu"
	document.write(e.message + " - " + e.nom)
	}

try...catch	Marque un bloque d'instructions à exécuter avec try. Si une exception est lancée, elle est récupérée dans le bloque catch.
	Syntaxe :
	try {
	code
	}
	[catch (nom) {
	code
	}]
	[finally {
	code
	}]
	Paramètres :
	• nom : nom de l'exception à traiter
	• finally : ce bloque d'instructions est exécuté qu'il y ait eu où non une exception.

var	Déclare une variable et l'initialise à une valeur. **Syntaxe** : var nom [= valeur] [..., nom [=valeur]] ex.: var a = 1, b = "bonjour"

while	Crée une boucle qui répète l'exécution du code tant que la condition est vraie. Si la condition est vraie dès le début, le programme n'entre pas dans la boucle.
	Syntaxe :
	while (condition) {
	code
	}

with	Indique l'objet sur lequelle on veut travailler pour un bloque d'instructions.
	Syntaxe :
	with (objet) {
	code
	}
	Paramètre : objet : objet sur lequel on veut travailler.
	ex.:
	with (Math) {
	a = PI * 2;
	// au lieu de Math.PI * 2
	b = cos(45);
	}

4.2 Les boucles infinies

1. while (true) { ... }
2. for (; ;) { ... }

Il faut faire attention à ces boucles car elle peuvent geler votre navigateur. **Il est déconseillé d'en faire**.
Ssi vous utilisez une telle boucle, vous recevrez un message d'erreur.
Pour quitter une telle boucle, vous devez utiliser l'instruction "break".

Chapitre 5 Fonctions et propriétés de haut niveau

Ce chapitre reprend toutes les fonctions et propriétés qui ne sont pas rattachées à un objet.

5.1 Propriétés

Infinity	Valeur numérique représentant l'inifni.
NaN	Valeur représentant Not-A-Number.
Undefined	Valeur indéfinie

5.2 Fonctions

escape	Renvoie l'encodage hexadécimal d'une string. Utilisé pour créer une string à ajouter à une URL. Les caractères encodés sont le banc, les ponctuations et tous les caractères non ASCII à l'exception de ceux-ci : * @ - _ + . / **Syntaxe** : escape("string") ex.: escape("&") donne %26 escape("Après l'hiver, c'est le printemps") donne Apr%E8s%20l%27hiver%2C%20c%27est%20le%20printemps unescape
eval	Evalue une string. **Syntaxe** : eval(string) **Paramètre** : string : Une expression qui peut inclure des variables ou des propriétés d'objets. ex.: eval("3+3") donne 6
isFinite	Evalue si l'argument est un nombre fini. Renvoie true si c'est vrai. **Syntaxe** : isFinite(nombre)
isNaN	Evalue si l'argument n'est pas un nombre. Renvoie true si c'est vrai. **Syntaxe** : isNaN(nombre) ex.: isNaN(3) donne false

	isNaN("a") donne true NaN, parseFloat, parseInt
Number	Convertit un objet en un nombre. **Syntaxe** : Number(objet) ex.: d = new Date("January 1, 2000 00:00:00"); alert(Number(d)) donne : 946681200000
parseFloat	Analyse une string et renvoie un nombre réel. **Syntaxe** : parseFloat(string) isNaN, parseInt
parseInt	Analyse une string et renvoie un nombre entier selon la base choisie. **Syntaxe** : parseInt(string [,base]) **Paramètres** : • string : string contenant le nombre. • base : entier représentant la base de la valeur de retour. La base peut être 8 (octal), 10 (décimal) ou 16 (hexadécimal). Si la base n'est pas spécifiée ou si elle est 0, le JavaScript analyse la string : si elle commence par "0x" la base sera 16. Si elle commence par "0", la base sera 8. Si la string commence par une autre valeur, la base sera 10. ex.: Ces exemples renvoient tous 10 parseInt("A",16) parseInt("12",8) parseInt("10.1",10) parseInt("AXY1",16) parseInt("1010",2) parseInt("10+1",10) parseInt("10+1") parseInt("0xA") Ces exemples renvoient tous NaN parseInt("Bonjour") parseInt("A",10)
String	Convertit un objet en string. **Syntaxe** : String(objet) ex.: d = new Date(946681200000); alert(String(d)) donne : Sat Jan 1 00:00:00 UTC+0100 2000
unescape	Renvoie la chaîne ASCII pour la valeur spécifiée en hexadécimal. **Syntaxe** : unescape(string) ex.: unescape("Apr%E8s%20l%27hiver%2C%20c%27est%20le%20printemps") renvoie : Après l'hiver, c'est le printemps escape

Chapitre 6 Les variables

Les variables sont des zones réservées en mémoire qui contiennent les valeurs que vous voulez garder.

En JavaScript, on n'est pas obligé de déclarer les variables, mais il est conseillé de le faire dans l'hypothèse où vous passeriez à un autre langage où là, vous devriez le faire. Les variables ne sont pas typées : elles peuvent contenir un entier, un chaîne de caractères ou un nombre réel.

6.1 Déclarer une variable

Pour déclarer une variable, il faut utiliser le mot clé **var**
var t = null
var u, v = 2 // ici, seul v est initialisé, u renverra *undefined* lorsqu'on demandera son contenu
var w = "3"
var x = 2
var y = "Bonjour"
var z = 1.5

Le contenu de chaque variable dépend de l'utilisation qu'on fait de cette variable :

- y + x donne Bonjour2
  y est une string et x est un nombre. Le JavaScript interprète x comme une string.
- x * w donne 6
  Le JavaScript interprète w comme un nombre et évalue l'expression

6.2 Types de variables

- String : "Bonjour"
- Nombre : Tout nombre entier ou avec virgule tel que 22 ou 3.1416
- Booléen (Valeur logique) : 2 états possibles : **true** (1) ou **false** (0).
- null : variable qui n'a pas de valeur
- undefined : variable pour laquelle on a pas encore assigné une valeur

6.3 Règles sur les variables

La déclaration de variables obéit à quelques contraintes :

- Les noms de variables doivent commencer par une lettre ou un underscore (_) ou un dollar ($)
- Les caractères suivants peuvent être des lettres, des chiffres, des underscores ou des dollars
- Les variable ne peuvent pas être des mots réservés
- Faire attention aux majuscules et minuscules

exemples de variables valides :

- nombre_Elements
- compteur1

exemples de variables invalides :

- nombre&Element // contient le &
- 1compteur //commence par un chiffre

6.4 Variables globales vs locales

- Les variables déclarées tout au début du script, en dehors et avant toutes fonctions seront toujours globales, qu'elles soient déclarées avec var ou de façon contextuelle. Les variables globales peuvent être utilisées dans tout le script et toute la page HTML, y compris à l'intérieur de plusieurs fonctions à la fois. Elles sont déclarées en dehors d'une fonction, entre les tags <script> et </script>.
- Une variable locale n'est utilisée que dans la fonction ou elle est déclarée et aura une portée limitée à cette seule fonction. On ne pourra donc pas l'exploiter ailleurs dans le script. D'où son nom de locale. Par contre, toujours dans une fonction, si la variable est déclarée contextuellement (sans utiliser le mot var), sa portée sera globale. Un fois que le programme quitte la fonction, la variable sera détruite.

6.5 Variables structurées

Il est possible de créer des variables structurées comme dans d'autres langages de programmation. Ce sont les types "record" en Delphi et "struct" en C.
ex.:

```
personne = {nom:"Gates",prenom:"Bill"}
document.write(personne.nom)
personne.activite = "entarté"
document.write(personne.activite)
```

Comme vous l'avez vu, il est possible de rajouter des propriétés à une telle variable.
1 + 1 n'est pas toujours égal à 2
Il faut faire attention lorsque vous manipulez des variables.
Selon le type de la variable, le résultat sera différent.

```
var a = 1 // chiffre
var b = "1" // chaîne de caractère
var result
result = a + b
// result contiendra 11 car b est une chaîne de caractère.
// La variable "a" est transformée en une chaîne de caractère, ce qui donne "1" + "1" = "11"
result = a + parseInt(b)
// result contiendra 2 car on a transformé la chaîne de caractère en un chiffre.
```

Exercice

Employer la méthode write() pour afficher des variables. On définit une variable appelée texte qui contient une chaîne de caractères "Mon chiffre préféré est " et une autre appelée variable qui est initialisée à 7.

```
<HTML>
<BODY>
<SCRIPT LANGUAGE="Javascript">
<!--
var texte = "Mon chiffre préféré est le "
var variable = 7
document.write(texte + variable);
//-->
</SCRIPT>
</BODY>
```

</HTML>

Le résultat se présente comme suit :
Mon chiffre préféré est le 7

6.6 Les mots réservés

Les mots de la liste ci-après ne peuvent être utilisés pour des noms de fonctions et de variables Certains de ces mots sont des mots clés Javascript, d'autres ont été réservés par Netscape pour un futur usage éventuel.

A	abstract
B	boolean break byte
C	case catch char class const continue
D	default do double delete debugger
E	else extends export enum
F	false final finally float for function
G	goto
I	if implements import in instanceof int interface
L	long
N	native new null
P	package private protected public
R	return
S	short static super switch synchronized
T	this throw throws transient true try typeof transient
V	var void volatile
W	while with

Chapitre 7 Les tableaux

Les tableaux commencent toujours à l'index 0.

7.1 Constructeurs

new Array(*longueur*)
new Array(*element0*, *element1*, ..., *elementN*)
tableau = [*element0*, *element1*, ..., *elementN*]

7.2 Paramètres

longueur	longueur initiale du tableau. Si la longueur n'est pas un nombre, le tableau aura la longueur 1 avec comme première valeur, la valeur spécifiée. La longueur maximum d'un tableau est de 4 294 967 295 éléments.
elementN	Une liste d'éléments pour initialiser le tableau.

exemples :

1. var tab = new Array(3);
   tab[0] = "Index 0";
   tab[1] = "Index 1";
   tab[2] = "Index 2";

2. var tab = new Array("Lundi","Mardi","Mercredi","Jeudi","Vendredi","Samedi","Dimanche")
   tab.length = 7
3. var tab = ["Lundi","Mardi","Mercredi","Jeudi","Vendredi","Samedi","Dimanche"]

7.3 Propriétés

Propiétés	Description
length	Renvoie le nombre d'éléments du tableau.

7.4 Méthodes

Méthodes	Description
concat	Réunit deux ou plusieurs tableaux en un seul. **Syntaxe** : concat(*tableau1*, *tableau2*, ..., *tableauN*) **Paramètres** : tableauN : des tableaux à assembler. ex.: tab1 = [0,1,2,3] tab2 = [4,5,6] tab3 = [7,8,9] tab = tab1.concat(tab2,tab3) //tab contiendra [0,1,2,3,4,5,6,7,8,9]

join	Réunit tous les éléments du tableau dans un string.
	Syntaxe : join(séparateur)
	Paramètres : séparateur : spécifie le séparateur entre chaque élément tu tableau. Si le séparateur n'est pas mis, les éléments seront séparés par une virgule (,).
	ex.:
	var tab = new Array("Lundi","Mardi","Mercredi","Jeudi","Vendredi","Samedi","Dimanche");
	var x = tab.join(" - ");
	x contiendra : Lundi - Mardi - Mercredi - Jeudi - Vendredi - Samedi - Dimanche

pop	Renvoie et retire le dernier élément du tableau.
	Pour Netscape uniquement
	Syntaxe : pop()
	Paramètres : aucun
	var tab = new Array("Lundi","Mardi","Mercredi","Jeudi","Vendredi","Samedi","Dimanche");
	elm = tab.pop();
	elm contiendra "Dimanche"
	tab contiendra "Lundi","Mardi","Mercredi","Jeudi","Vendredi","Samedi"
	push, shift, unshift

push	Ajoute un ou plusieurs éléments à la fin du tableau et renvoie la nouvelle longueur.
	Pour Netscape uniquement
	Syntaxe : push(*element1*, *element2*, ..., *elementN*)
	Paramètres : elementN : éléments à rajouter au tableau.
	var tab = new Array("Lundi","Mardi","Mercredi","Jeudi","Vendredi");
	longueur = tab.push("Samedi","Dimanche");
	longueur contiendra 7
	tab contiendra "Lundi","Mardi","Mercredi","Jeudi","Vendredi","Samedi","Dimanche"
	pop, shift, unshift

reverse	Inverse un tableau : le premier élément devient le dernier et le dernier devient le premier.
	Syntaxe : reverse()
	Paramètre : aucun
	ex.:
	var tab = new Array("1","2","3","4");
	tab.reverse();

	tab[0] = "4" tab[1] = "3" tab[2] = "2" tab[3] = "1" join, sort

shift	Enlève le premier élément du tableau et renvoie celui-ci.
	Pour Netscape uniquement
	Syntaxe : shift()
	Paramètre : aucun
	ex.:
	var tab = new Array("1","2","3","4");
	x = tab.shift();
	tab contiendra ("2,"3","4")
	x = "1"
	pop, push, unshift

slice	Extrait une partie d'un tableau et renvoie un nouveau tableau. **Syntaxe** : slice(*debut* [, *fin*]) **Paramètre**s : • debut : index de début • fin : index de fin. L'élément à l'index *fin* ne sera pas contenu dans le tableau. ex.: var tab = new Array("1","2","3","4"); x = tab.slice(1,3) x contiendra 2,3
splice	Change le contenu du tableau en ajoutant de nouveaux éléments et en supprimant des éléments. Renvoie un tableau avec les éléments retirés. Pour Netscape uniquement **Syntaxe** : splice(*index*, *nombre*, [*element1*][, ..., *elementN*])) **Paramètre**s : • index : index à partir duquel on change le tableau • nombre : nombre d'éléments à enlever • elementN : éléments à rajouter ex.: tab = ["1","2","3","4"] tab.splice(1,0,"5") tab contiendra 1,5,2,3,4 tab = ["1","2","3","4"] x = tab.splice(2,2,"5") tab contiendra 1,2,5 x contiendra 3,4

sort	Trie les éléments d'un tableau.
	Syntaxe : sort([*fncCompare*])
	Paramètres : fncCompare : une fonction de comparaison. Si elle n'est pas mise, le tableau est trié lexicographiquement (dans l'ordre du dictionnaire).
	Le tableau est trié en fonction de la valeur de retour de la fonction :
	• si a < b, la fonction doit être < 0
	• si a > b, la fonction doit être > 0
	• si a = b, la fonction doit être = 0
	ex.:
	function fncCompare(a,b){ //comparaison de strings
	if (a < b) return -1;
	if (a > b) return 1;

	return 0; //a == b
	}
	function fncCompare(a,b){ //comparaison de nombres
	return a - b;
	}
	//sort sans fncCompare
	var tab = new Array(91,10,6,20,75,8)
	tab.sort()
	//tab contiendra 10,20,6,75,8,91
	//sort avec fncCompare
	var tab = new Array(91,10,6,20,8,75)
	function compare(a,b){
	return a-b;
	}
	tab.sort(compare);
	//tab contiendra 6,8,10,20,75,91
	join, reverse

toSource	Renvoie le code source du tableau.
	Syntaxe : toSource()
	Paramètres : aucun
	ex.:
	var tab = new Array("1","2","3")
	tab.toSource() //renvoie 1,2,3
	toString

toString	Renvoie une string représentant le tableau et ses éléments. **Syntaxe** : toString() **Paramètres** : aucun ex.: var tab = new Array("1","2","3") var x = tab.toString() //x contiendra "1,2,3" toSource

unshift	Ajoute un ou plusieurs éléments au début du tableau et renvoie la nouvelle longueur du tableau.
	Pour Netscape uniquement
	Syntaxe : unshift(*element1*, *element2*, ..., *elementN*)
	Paramètres : elementN : éléments à rajouter au début du tableau.
	ex.:
	var tab = new Array("4","5","6","a")
	var longueur = tab.unshift("1","2","3")
	//tab contiendra 1,2,3,4,5,6
	//longueur contiendra 7
	shift

valueOf	Renvoie la valeur primitive d'un tabeau. **Syntaxe** : valueOf() **Paramètres** : aucun

Chapitre 8 Les expressions régulières

Les expressions régulières servent pour la recherche et le remplacement de chaînes de caractères.

8.1 Constructeur

new RegExp(*"pattern"* [, *"flags"*])

8.2 Paramètres :

- pattern : le texte de l'expression régulière
- flags :

g	"global match" : recherche sur toute la string
i	"ignore case" : ignore la différence entre majuscules et minuscules
g i	combine les 2 premiers paramètres

ex.:
x = new RegExp("a*", "i") □ recherche 0 ou plusieurs a
Liste et description des caractères des expressions régulières.

Caractères	Description
\	Indique que le caractère suivant est un caractère spécial et ne doit pas être interprété littérallement. ex.: pour rechercher 'a*', il faut utiliser /a*/ car * est un caractère spécial qui veut dire 0 ou plusieurs. Si vous mettez 'a*', il va rechercher 0 ou plusieurs a.
^	Correspond au début de la ligne. ex.: /^a/ ne correspond pas à "bca" mais correspond à "abc"
$	Correspond à la fin de la ligne. ex.: /^a/ ne correspond pas à "abc" mais correspond à "bca"
*	Recherche le caractère précédent 0 ou plusieurs fois. ex.: /ab*/ correspond "abbab" mais ne correspond à "aaa"
+	Recherche le caractère précédent 1 ou plusieurs fois.
?	Recherche le caractère précédent 0 ou 1 fois. ex.: /a?v/ correspond au "av" de "JavaScript"
. (point décimal)	Recherche n'importe quel caractère unique à l'exception du caractère de début de ligne. ex.: /.c/ correspond au "ci" et au "éc" de décimal, mais pas à "ceci" dans "ceci est un exemple avec le point décimal"
x|y	Recherche 'x' ou 'y'.
(n)	Quand n est un entier positif, il recherche exactement n occurences du caractère

	précédent. ex.: /a(2)/ n'est pas trouvé dans "Netscape" mais prend les 2 premiers 'a' dans "JavaScript de Netscape"
(n,)	Quand n est un entier positif, il recherche n occurences ou plus du caractère précédent.
(n,m)	Quand n et m sont des entiers positifs, il recherche au moins n et au plus m occurences du caractère précédent.
[xyz]	Recherche n'importe quel caractère de ceux spécifiés. ex.: [abcd] [a-d] dans "JavaScript" la correspondance est trouvée avec le 'a'.
[^xyz]	Recherche n'importe quel caractère qui ne font pas partie de ceux spécifiés. ex.: [^abcd] dans "JavaScript" la première correspondance est trouvée avec le 'J'.
[\b]	Recherche la backspace. A ne pas confondre avec \b
\b	Recherche dans l'extrémité d'un mot. A ne pas confondre avec [\b] ex.: /\bJ\w/ correspond à 'Ja' dans 'JavaScript' /\wt\b/ correspond à 'pt' dans 'JavaScript'
\B	Recherche dans l'intérieur d'un mot. /\wBi/ correspond à 'ri' dans 'JavaScript'
\d	Recherche les nombres. Equivalent à [0-9]
\D	Recherche tous les caractères sauf les nombres. Equivalent à [^0-9]
\f	Recherche le "form feed" (saut de page).
\n	Recherche le "line feed" (interligne, saut de ligne).
\r	Recherche le "carriage return" (retour à la ligne).
\s	Rechereche un simple espace : espace, form feed, line feed, tabulation. Equivalent à [\f\n\v\t\r].
\S	Rechereche un autre caractère qu'un espace. Equivalent à [^ \f\n\v\t\r].
\t	Rechereche une tabulation.
\v	Rechereche une tabulation verticale.
\w	Rechereche n'importe quel caractère alphanumérique, y compris le underscore (_) Equivalent à [A-Za-z0-9_].
\W	Rechereche n'importe quel caractère non alphanumérique, y compris le underscore (_). Equivalent à [^A-Za-z0-9_].
\ooctal \xhex	\ooctal est une valeur octale et \xhex est une valeur hexadécimale. Cela vous permet d'introduire un code ASCII dans une expressions régulière.

Chapitre 9 Les principales fonctions du javascript

9.1 Définition d'une fonction

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

En Javascript, il existe deux types de fonctions :

- les fonctions propres à Javascript. On les appelle des "méthodes". Elles sont associées à un objet bien particulier comme c'était le cas de la méthode Alert() avec l'objet window.
- les fonctions écrites par vous-même pour les besoins de votre script. C'est à celles-là que nous nous intéressons maintenant.

9.2 Déclaration des fonctions

Pour déclarer ou définir une fonction, on utilise le mot (réservé) function.
La syntaxe d'une déclaration de fonction est la suivante :

```
function nom_de_la_fonction(arguments) {
... code des instructions ...
}
```

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

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

Lorsque une accolade est ouverte, elle doit impérativement, sous peine de message d'erreur, être refermée. Prenez la bonne habitude de fermer directement vos accolades et d'écrire votre code entre elles.

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

9.3 L'appel d'une fonction

L'appel d'une fonction se fait le plus simplement du monde par le nom de la fonction (avec les parenthèses).

Soit par exemple nom_de_la_fonction();

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

9.4 Les fonctions dans <HEAD>...<HEAD>

Il est prudent et judicieux de placer toutes les déclarations de fonction dans l'en-tête de votre page c.-à-d .dans la balise <HEAD> ... <HEAD>. Vous serez ainsi assuré que vos fonctions seront déjà prises en compte par l'interpréteur avant qu'elles soient appelées dans le <BODY>.

Exemple

Dans cet exemple, on définit dans les balises HEAD, une fonction appelée message() qui affiche le texte "Bienvenue à ma page". cette fonction sera appelée au chargement de la page voir onLoad=.... dans le tag <BODY>.

```
<HTML>
<HEAD>
<SCRIPT LANGUAGE="Javascript">
<--
function message() {
document.write("Bienvenue à ma page");
}
//-->
</SCRIPT>
</HEAD>
<BODY onLoad="message()">
</BODY>
</HTML>
```

9.5 Passer une valeur à une fonction

On peut passer des valeurs ou paramètres aux fonctions Javascript. La valeur ainsi passée sera utilisée par la fonction.
Pour passer un paramètre à une fonction, on fournit un nom d'une variable dans la déclaration de la fonction.

Un exemple simple pour comprendre. J'écris une fonction qui affiche une boite d'alerte dont le texte peut changer.
Dans la déclaration de la fonction, on écrit :

```
function Exemple(Texte) {
alert(texte);
}
```

Le nom de la variable est Texte et est définie comme un paramètre de la fonction.
Dans l'appel de la fonction, on lui fournit le texte :

```
Exemple("Salut à tous");
```

9.6 Passer plusieurs valeurs à une fonction

On peut passer plusieurs paramètres à une fonction. Comme c'est souvent le cas en Javascript, on sépare les paramètres par des virgules.

```
function nom_de_la_fonction(arg1, arg2, arg3) {
... code des instructions ...
}
```

Notre premier exemple devient pour la déclaration de fonction :
```
function Exemplebis(Texte1, Texte2){...}
```

et pour l'appel de la fonction
```
Exemplebis("Salut à tous", "Signé Luc")
```

9.7 Variables locales et variables globales

Avec les fonctions, le bon usage des variables locales et globales prend toute son importance.

Une variable déclarée dans une fonction par le mot clé var aura une portée limitée à cette seule fonction. On ne pourra donc pas l'exploiter ailleurs dans le script. On l'appelle donc variable locale.

```
function cube(nombre) {
var cube = nombre*nombre*nombre
}
```

Ainsi la variable cube dans cet exemple est une variable locale. Si vous y faites référence ailleurs dans le script, cette variable sera inconnue pour l'interpréteur Javascript (message d'erreur).

Si la variable est déclarée contextuellement (sans utiliser le mot var), sa portée sera globale -- et pour être tout à fait précis, une fois que la fonction aura été exécutée--.

```
function cube(nombre) {
cube = nombre*nombre*nombre
}
```

La variable cube déclarée contextuellement sera ici une variable globale.

Les variables déclarées tout au début du script, en dehors et avant toutes fonctions, seront toujours globales, qu'elles soient déclarées avec var ou de façon contextuelle.

```
<SCRIPT LANGUAGE="javascript">
var cube=1
function cube(nombre) {
var cube = nombre*nombre*nombre
}
</SCRIPT>
```

La variable cube sera bien globale.

Pour la facilité de gestion des variables, on ne peut que conseiller de les déclarer en début de script (comme dans la plupart des langages de programmation). Cette habitude vous met à l'abri de certaines complications.

Voici une liste avec description des objets employés dans JavaScript.

Date

La date est mesurée en millisecondes depuis le 01 janvier 1970 00:00:00.
Pour la compatibilité avec l'an 2000, vous devez toujours spécifié l'année complète : 1999 au lieu de 99. Pour cela, il y a les fonctions getFullYear, setFullYear, getFullUTCYear et setFullUTCYear.

new Date()
new Date(*millisecondes*)
new Date(*dateString*)
new Date(*année*,*mois*,*jour* [,*heures*,*minutes*,*secondes*,*millisecondes*])

Paramètres

milliseconds	Valeur entière représentant le nombre de millisecondes depuis le 1 January 1970 00:00:00.
dateString	Chaîne de caractères représentant une date. La chaîne doit être dans un format reconnu par la méthode Date.parse.
année, mois, jour	Valeure entière représentant une partie de date. January = 0, February = 1, ..., December = 11.
heures, minutes, secondes, millisecondes	Valeur entière représentant une partie de date.

Méthodes

Toutes les méthodes contenant UTC utilisent le temps universel. Les autres utilisent le temps local.

Méthodes	Description

getDate	Renvoie le jour du mois.
	Syntaxe : getDate()
	Paramètres : aucun.
	ex.:
	bug = new Date("January 01, 2000 00:00:00")
	jour = bug.getDate() //renvoie 1
	Date.getUTCDate, Date.getUTCDay, Date.setDate

getDay	Renvoie le jour de la semaine. 0 = dimanche, 1 = lundi, ... **Syntaxe** : getDay() **Paramètres** : aucun. Date.getUTCDate, Date.setDate
getFullYear	Renvoie l'année en long format : 1999 et non 99. **Syntaxe** : getFullYear() **Paramètres** : aucun. Date.getYear, Date.getUTCFullYear, Date.setFullYear
getHours	Renvoie l'heure. **Syntaxe** : getHours() **Paramètres** : aucun. Date.getUTCHours, Date.setHours
getMilliSeconds	Renvoie les millisecondes.

	Syntaxe : getMilliSeconds() **Paramètres** : aucun. Date.getUTCMilliSeconds, Date.setMillSeconds
getMinutes	Renvoie les minutes. **Syntaxe** : getMinutes() **Paramètres** : aucun. Date.getUTCMinutes, Date.setMinutes
getMonth	Renvoie le mois de l'année de 0 à 11. **Syntaxe** : getMonth() **Paramètres** : aucun. Date.getUTCMonth, Date.setMonth
getSeconds	Renvoie les secondes. **Syntaxe** : getSeconds() **Paramètres** : aucun. Date.getUTCSeconds, Date.setSeconds
getTime	Renvoie le nombre de millisecondes depuis le 1 January 1970 00:00:00 **Syntaxe** : getTime() **Paramètres** : aucun. Date.getUTCHours, Date.setTime
getTimeZoneOffset	Renvoie la différence entre l'heure locale et l'heure GMT (Greenwich Mean Time). **Syntaxe** : getTimeZoneOffset() **Paramètres** : aucun. ex.: maDate = new Date() FuseauHoraireEnHeures = maDate.getTimeZoneOffset()/60
getUTCDate	Renvoie le jour du mois **Syntaxe** : getUTCDate() **Paramètres** : aucun. Date.getDate, Date.getUTCDay, Date.setUTCDate
getUTCDay	Renvoie le jour de la semaine. 0 = dimanche, 1 = lundi, ... **Syntaxe** : getUTCDay() **Paramètres** : aucun. Date.getDay, Date.getUTCDate, Date.setUTCDate
getUTCFullYear	Renvoie l'année en long format. **Syntaxe** : getUTCFulleYear() **Paramètres** : aucun. Date.getFullYear, Date.setFullYear
getUTCHours	Renvoie l'heure. **Syntaxe** : getUTCHours() **Paramètres** : aucun. Date.getHours, Date.setUTCHours
getUTCMilliseconds	Renvoie les millisecondes. **Syntaxe** : getUTCMilliSeconds() **Paramètres** : aucun. Date.getMilliseconds, Date.setUTCMilliseconds
getUTCMinutes	Renvoie les minutes. **Syntaxe** : getUTCMinutes() **Paramètres** : aucun. Date.getMinutes, Date.setUTCMinutes
getUTCMonth	Renvoie le mois de l'année. **Syntaxe** : getUTCMonth() **Paramètres** : aucun.

	Date.getMonth, Date.setUTCMonth
getUTCSeconds	Renvoie les secondes. **Syntaxe** : getUTCSeconds() **Paramètres** : aucun. Date.getSeconds, Date.setUTCSeconds
getYear	Renvoie l'année moins 1900. **Syntaxe** : getYear() **Paramètres** : aucun. Pour les années 1900 <= année <= 1999, getYear renvoie un chiffre entre 0 et 99. ex.: x = new Date("January 01, 2000 00:00:00") x.getYear() renvoie 2000 x = new Date("January 01,1999 00:00:00") x.getYear() renvoie 99 Date.getFullYear, Date.getUTCFullYear, Date.setYear
parse	Renvoie le nombre de millisecondes depuis le 01 January 1970 00:00:00. Le format de la date peut être : • "Dec 25, 1999" • "Fri, 31 Dec 1999 23:59:59 GMT" • "Fri, 31 Dec 1999 23:59:59 GMT+0100" Ce format spécifie le fuseau horaire dans lequel on veut travailler. Si vous ne spécifiez pas de fuseau horaire, GMT et UTC sont considéré comme équivalent. Vous devez utiliser Date.parse() car c'est une méthode static. **Syntaxe** : Date.parse(*dateString*) **Paramètres** : une date en chaîne de caractères. ex.: maDate.setTime(Date.parse("Jan 1, 2000") maDate.setTime(Date.parse("Sat, 01 Jan 2000 12:00:00 GMT+0100") Date.UTC
setDate	Change le jour du mois pour une date donnée. **Syntaxe** : setDate(*jour*) **Paramètres** : un entier compris entre 1 et 31. ex.: maDate = new Date("January 01, 2000 12:00:00") maDate.setDate(2) Date.getDate, Date.setUTCDate
setFullYear	Change la date en long format. **Syntaxe** : setFullYear(*année* [, *mois*, *jour*]) **Paramètres** : • année : un entier représentant la date. ex.: 2000. • mois : un entier compris entre 0 et 11. • jours : un entier compris entre 1 et 31. **Remarque** : Si vous ne spécifiez pas le mois et le jour, les valeurs utilisées sont celles renvoyées par getMonth et getDay. setFullYear adapte la date si les paramètres dépassent les valeurs permises. ex.: Si vous mettez mois = 16, l'année est incrémentée de 1 (an+1) et mois=4. Date.getUTCFullYear, Date.setUTCFullYear, Date.setYear
setHours	Change l'heure. **Syntaxe** : setHours(*heures* [,*minutes*, *secondes*, *millisecondes*]) **Paramètres** : • heures : un entier compris entre 0 et 23.

	• minutes : un entier compris entre 0 et 59. • secondes : un entier compris entre 0 et 59. • millisecondes : un entier compris entre 0 et 999. Si les derniers paramètres ne sont pas mis, les valeurs renvoyées sont proviennent de getUTCMinutes, getUTCSeconds et getUTCMilliSeconds. **Remarque** : même remarque que pour setFullYear. Date.getHours, Date.setUTCHours
setMilliSeconds	Change les millisecondes. **Syntaxe** : setMilliSeconds(*millisecondes*) **Paramètres** : millisecondes : un entier compris entre 0 et 999. **Remarque** : même remarque que pour setFullYear. Date.getMilliseconds, Date.setUTCMilliseconds
setMinutes	Change les minutes. **Syntaxe** : setMinutes(*minutes*) **Paramètres** : minutes : un entier compris entre 0 et 59. **Remarque** : même remarque que pour setFullYear. Date.getMinutes, Date.setUTCMilliseconds
setMonth	Change le mois. **Syntaxe** : setMonth(*mois* [, *jour*]) **Paramètres** : • mois : un entier compris entre 0 et 11. • jours : un entier compris entre 1 et 31. **Remarque** : même remarque que pour setFullYear. Date.getMonth, Date.setUTCMonth
setSeconds	Change les secondes. **Syntaxe** : setSeconds(*secondes* [, *millisecondes*]) **Paramètres** : • secondes : un entier compris entre 0 et 59. • millisecondes : un entier compris entre 0 et 999. **Remarque** : même remarque que pour setFullYear. Date.getSeconds, Date.setUTCSeconds
setTime	Change la date. **Syntaxe** : setTime(*valeur*) **Paramètres** : valeur est le temps en millisecondes depuis le 01 January 1970 00:00:00 ex.: maDate = newDate("January 01, 1999") nouvelleDate = new Date() nouvelleDate.setTime(maDate.getTime()) Date.getTime, Date.setUTCHours
setUTCDate	Change le jour du mois pour une date donnée. **Syntaxe** : setUTCDate(*jour*) **Paramètres** : un entier compris entre 1 et 31. Date.getUTCDate, Date.setDate
setUTCFullYear	Change la date en long format. **Syntaxe** : setUTCFullYear(*année* [, *mois*, *jour*]) **Paramètres** : • année : un entier représentant la date. ex.: 2000. • mois : un entier compris entre 0 et 11.

	• jours : un entier compris entre 1 et 31. **Remarque** : Si vous ne spécifiez pas le mois et le jour, les valeurs utilisées sont celles renvoyées par getMonth et getDay. setUTCFullYear adapte la date si les paramètres dépassent les valeurs permises. ex.: Si vous mettez mois = 16, l'année est incrémentée de 1 (an+1) et mois=4. Date.getUTCFullYear, Date.setFullYear
setUTCHours	Change l'heure. **Syntaxe** : setUTCHours(*heures* [,*minutes*, *secondes*, *millisecondes*]) **Paramètres** : • heures : un entier compris entre 0 et 23. • minutes : un entier compris entre 0 et 59. • secondes : un entier compris entre 0 et 59. • millisecondes : un entier compris entre 0 et 999. Si les derniers paramètres ne sont pas mis, les valeurs renvoyées sont proviennent de getUTCMinutes, getUTCSeconds et getUTCMilliSeconds. **Remarque** : même remarque que pour setUTCFullYear. Date.getUTCHours, Date.setHours
setUTCMilliSeconds	Change les millisecondes. **Syntaxe** : setUTCMilliSeconds(*millisecondes*) **Paramètres** : millisecondes : un entier compris entre 0 et 999. **Remarque** : même remarque que pour setUTCFullYear. Date.getUTCMilliseconds, Date.setMillseconds
setUTCMinutes	Change les minutes. **Syntaxe** : setUTCMinutes(*minutes*) **Paramètres** : minutes : un entier compris entre 0 et 59. **Remarque** : même remarque que pour setUTCFullYear. Date.getUTCMinutes, Date.setMinutes
setUTCMonth	Change le mois. **Syntaxe** : setUTCMonth(*mois* [, *jour*]) **Paramètres** : • mois : un entier compris entre 0 et 11. • jours : un entier compris entre 1 et 31. **Remarque** : même remarque que pour setUTCFullYear. Date.getUTCMonth, Date.setMonth
setUTCSeconds	Change les secondes. **Syntaxe** : setUTCSeconds(*secondes* [, *millisecondes*]) **Paramètres** : • secondes : un entier compris entre 0 et 59. • millisecondes : un entier compris entre 0 et 999. **Remarque** : même remarque que pour setUTCFullYear. Date.getUTCSeconds, Date.setSeconds
setYear	Change l'année. **Syntaxe** : setYear(*année*) **Paramètres** : année : un entier. **Remarque** : setYear n'est plus utilisé et est remplacé par setFullYear.

	Date.getYear, Date.setFullYear, Date.setUTCFullYear
toGMTString	Convertit une date en une chaîne de caractères en utilisant les conventions GMT (UTC). **Syntaxe** : toGMTString() **Paramètres** : aucun. **Remarque** : toGMTString n'est plus utilisé et est remplacé par toUTCString. ex.: var today = new Date() today.toGMTString() donne : Tue, 10 Jul 2001 14:23:28 UTC Date.toLocaleString, Date.toUTCString
toLocaleString	Convertit une date en une chaîne de caractères en utilisant les conventions locales. **Syntaxe** : toLocaleString() **Paramètres** : aucun. ex.: var today = newDate() today.toLocaleString() donne : 07/10/2001 16:23:28 Date.toGMTString, Date.toUTCString
toSource	Renvoie une chaîne de caractères représentant le code source. **Syntaxe** : toSource() **Paramètres** : aucun.
toString	Renvoie une chaîne de caractères représentant la date. **Syntaxe** : toString() **Paramètres** : aucun. ex.: var today = newDate() today.toString() donne : Tue Jul 10 16:23:28 UTC+0200 2001
toUTCString	Renvoie une chaîne de caractères représentant la date en temps universel. **Syntaxe** : toUTCString() **Paramètres** : aucun. ex.: var today = newDate() today.toUTCString() donne : Tue, 10 Jul 2001 14:23:28 UTC Date.toLocaleString
UTC	Renvoie le nombre de millisecondes depuis le 1 January 1970 00:00:00 en temps universel. **Syntaxe** : Date.UTC(*année*, *mois*, *jour* [, *heures*, *minutes*, *secondes*, *millisecondes*]) **Paramètres** : • année : un entier > 1900. • mois : 0 <= mois <= 11. • jour : 1 <= jour <= 31. • heures : 0 <= heures <= 59. • minutes : 0 <= minutes <= 59. • secondes : 0 <= secondes <= 59. • millisecondes : 0 <= millisecondes <= 999. **Remarque :** Vous devez utiliser Date.UTC() car c'est une méthode static. ex.: gmtDate = new Date(Date.UTC(99,11,25,0,0,0)) donne : Sat Dec 25 01:00:00 UTC+0100 1999 Date.parse
valueOf	Renvoie le nombre de millisecondes depuis le 1 January 1970 00:00:00 en temps universel. **Syntaxe** : Date.valueOf()

	Paramètres : aucun. ex.: var today = new Date() today.valueOf() donne : 994775008160

9.8 Détail des méthodes concernant les dates et heures (l'objet date)

new Date();

Cette méthode renvoie toutes les informations "date et heure" de l'ordinateur de l'utilisateur.

variable=new Date();

Ces informations sont enregistrées par Javascript sous le format :
"Fri Dec 17 09:23:30 1998"

Attention ! La date et l'heure dans Javascript commence au 1e janvier 1970. Toute référence à une date antérieure donnera un résultat aléatoire.

La méthode new date () sans arguments renvoie la date et l'heure courante.
Pour introduire une date et une heure déterminée, cela se fera sous la forme suivante :
variable=new Date("Jan 1, 2000 00:00:00");

Toutes les méthodes suivantes vous faciliterons la tâche pour accéder à un point précis de cette variable (en fait un string) et pour modifier si besoin en est le format d'affichage.

getYear()

variable_date=new Date();
an=variable_date.getYear();

Retourne les deux derniers chiffres de l'année dans variable_date. Soit ici 97.
Comme vous n'avez que deux chiffres, il faudra mettre 19 ou bientôt 20 en préfixe soit
an="19"+variable_date.getYear();

getMonth()

variable_date=new Date();
mois=variable_date.getMonth();

Retourne le mois dans variable_date sous forme d'un entier compris entre 0 et 11 (0 pour janvier, 1 pour février, 2 pour mars, etc.). Soit ici 11 (le mois moins 1).

<Image>

getDate();

variable_date=new Date();
jourm=variable_date.getDate();

Retourne le jour du mois dans variable_date sous forme d'un entier compris entre 1 et 31.
Eh oui, ici on commence à 1 au lieu de 0 (pourquoi???).
A ne pas confondre avec getDay() qui retourne le jour de la semaine.

getDay();

variable_date=new Date();
jours=variable_date.getDay();

Retourne le jour de la semaine dans variable_date sous forme d'un entier compris entre 0 et 6 (0 pour dimanche, 1 pour lundi, 2 pour mardi, etc.).

getHours();

variable_date=new Date();
hrs=variable_date.getHours();

Retourne l'heure dans variable_date sous forme d'un entier compris entre 0 et 23.

getMinutes();

variable_date=new Date();
min=variable_date.getMinutes();

Retourne les minutes dans variable_date sous forme d'un entier compris entre 0 et 59.

getSeconds();

variable_date=new Date();
sec=variable_date.getSeconds();

Retourne les secondes dans variable_date sous forme d'un entier compris entre 0 et 59.

Exemple 1 : Un script qui donne simplement l'heure.

```
<HTML>
<HEAD>
<SCRIPT LANGUAGE="Javascript">
<!--
function getDt(){
dt=new Date();
cal=""+ dt.getDate()+"/"+(dt.getMonth()+1)
+ "/19" +dt1.getYear();
hrs=dt.getHours();
min=dt.getMinutes();
sec=dt.getSeconds();
tm=" "+((hrs<10)?"0":"") +hrs)+":";
tm+=((min<10)?"0":"")+min+":";
tm+=((sec<10)?"0":"")+sec+" ";
document.write(cal+tm);
}
// -->
</SCRIPT>
</HEAD>
<BODY >
<SCRIPT LANGUAGE="Javascript">
<!--
```

```
getDt();
// -->
</SCRIPT>
</BODY>
</HTML>
```

Exemple 2 : Un script avec une trotteuse.

Vous souhaitez que votre affichage de l'heure change toutes les secondes. Il faut ajouter au script un setTimeOut qui affiche l'heure toutes les secondes. La fonction qui affiche l'heure étant getDt(), l'instruction à ajouter est donc setTimeOut("getDt()",1000);

```
<HTML>
<HEAD>
<SCRIPT LANGUAGE="Javascript">
<!--
function getDt(){
dt=new Date();
hrs=dt.getHours();
min=dt.getMinutes();
sec=dt.getSeconds();
tm=" "+((hrs<10)?"0":"") +hrs+":";
tm+=((min<10)?"0":"")+min+":";
tm+=((sec<10)?"0":"")+sec+" ";
document.horloge.display.value=tm;
setTimeout("getDt()",1000);
}
// -->
</SCRIPT>
</HEAD>
<BODY onLoad="getDt()">
<FORM name="horloge">
<INPUT TYPE="text" NAME="display" SIZE=15 VALUE ="">
</FORM>
</BODY>
</HTML>
```

D'autres propriétés

variable.getTime();

Retourne l'heure courante dans variable_date sous forme d'un entier représentant le nombre de millisecondes écoulées depuis le 1 janvier 1970 00:00:00.

variable.getTimezoneOffset();

Retourne la différence entre l'heure locale et l'heure GMT (Greenwich, UK Mean Time) sous forme d'un entier représentant le nombre de minutes (et pas en heures).

variable.setYear(x);

Assigne une année à à l'actuelle valeur de variable_date sous forme d'un entier supérieur à 1900. Exemple : variable_date.setYear(98) pour 1998.

variable.setMonth(x);

Assigne un mois à l'actuelle valeur de variable_date sous forme d'un entier compris entre 0 et 11.
Exemple : variable_date.setMonth(1);

variable.setDate(x);

Assigne un jour du mois à l'actuelle valeur de variable_date sous forme d'un entier compris entre 1 et 31.
Exemple : variable_date.setDate(1);

variable.setHours(x);

Assigne une heure à l'actuelle valeur de variable_date sous forme d'un entier compris entre 1 et 23.
Exemple : variable_date.setHours(1);

variable.setMinutes(x);

Assigne les minutes à l'actuelle valeur de variable_date sous forme d'un entier compris entre 1 et 59.
Exemple : variable_date.setMinutes(1);

variable.setSeconds(x);

Assigne les secondes à l'actuelle valeur de variable_date sous forme d'un entier compris entre 1 et 59.
Exemple : variable_date.setSeconds(0);

variable.setTime(x);

Assigne la date souhaitée dans variable_date sous forme d'un entier représentant le nombre de millisecondes écoulées depuis le 1 janvier 1970 00:00:00. et la date souhaitée.

variable.toGMTString();

Retourne la valeur de l'actuelle valeur de variable_date en heure GMT (Greenwich Mean Time).

variable.toLocaleString()

Retourne la valeur de l'actuelle valeur de variable_date sous forme de String.
Il me semble qu'on aura plus vite fait avec les getHours(), getMinutes() and getSeconds().

Chapitre 10 Math

Toutes les méthodes de Math sont statiques : vous devez toujours les référencer par Math.Méthode.
Ex.: Math.PI, Math.sin(x)
Mais vous pouvez aussi vous simplifier la tâche en utilisant *with*.
Ex.:

```
with (Math){
a = PI * r * r
x = r * sin(45)
}
```

NaN : cela signifie Not-A-Number. Cette valeur est renvoyée lorsque, par exemple, vous utilisez une fonction mathématique avec une chaîne de caractères ou lorsque vous dépassez les paramètres autorisés.
-Infinity : Cette valeur est renvoyée lorsque une fonction mathématique renvoie l'infini comme résultat.

10.1 Propriétés

Propiétés	Description
E	Constante d'Euler et la base des logarithmes naturels. Vaut +/- 2.718
LN10	Logarithme naturel de 10. Vaut +/- 2.302
LN2	Logarithme naturel de 10. Vaut +/- 0.693
LOG10E	Logarithme en base 10 de E. Vaut +/- .434
LOG2E	Logarithme en base 2 de E. Vaut +/-1.442
PI	3.14159
SQRT1_2	Racine carrée de 0.5. Vaut +/- 0.707
SQRT2	Racine carrée de 2. Vaut 1.414

10.2 Méthodes

Méthodes	Description
abs	Renvoie la valeur absolue. **Syntaxe** : abs(valeur) **Paramètres** : valeur : un nombre. ex.: Math.abs(-10.5)
acos	Arccosinus en radian. Renvoie un nombre compris entre 0 et PI radians. Si le paramètre dépasse les valeurs permises, acos renvoie NaN. **Syntaxe** : acos(valeur) **Paramètres** : valeur : un nombre entre -1 et 1. ex.: Math.acos(0.5) Math.asin, Math.atan, Math.atan2, Math.cos, Math.sin, Math.tan
asin	Arcsinus en radian. Renvoie un nombre compris entre -PI/2 et PI/2 radians. Si le paramètre dépasse les valeurs permises, acos renvoie NaN. **Syntaxe** : asin(valeur) **Paramètres** : valeur : un nombre entre -1 et 1. ex.: Math.asin(0.5) Math.acos, Math.atan, Math.atan2, Math.cos, Math.sin, Math.tan
atan	Arctangente en radian. Renvoie un nombre compris entre -PI/2 et PI/2 radians. **Syntaxe** : atan(valeur) **Paramètres** : valeur : un nombre. ex.: Math.atan(-1) = -0.7853981633974483 Math.acos, Math.asin, Math.atan2, Math.cos, Math.sin, Math.tan
atan2	Arctangente en radian. Renvoie un nombre compris entre -PI et PI représentant l'angle théta d'un point (x,y). **Syntaxe** : atan2(x,y) **Paramètres** : x,y : un nombre.

	Math.acos, Math.asin, Math.atan, Math.cos, Math.sin, Math.tan
ceil	Renvoie le plus petit entier >= au nombre. **Syntaxe** : ceil(x) **Paramètres** : x : un nombre. ex.: Math.ceil(24.9) = 25 Math.ceil(-24.9) = -24 Math.floor
cos	Cosinus en radian. Renvoie un nombre entre -1 et 1. **Syntaxe** : cos(x) **Paramètres** : x : un nombre. Math.acos, Math.asin, Math.atan, Math.sin, Math.tan
exp	Renvoie E^x où E est la constante d'Euler. **Syntaxe** : exp(x) **Paramètres** : x : un nombre. ex.: exp(1) = 2.718281828459045 Math.E, Math.log, Math.pow
floor	Renvoie le plus grand entier <= au nombre. **Syntaxe** : floor(x) **Paramètres** : x : un nombre. ex.: Math.floor(24.9) = 24 Math.floor(-24.9) = -25 Math.ceil
log	Renvoie le logarithme en base E d'un nombre. **Syntaxe** : log(x) **Paramètres** : x : un nombre > 0. ex.: si le nombre < 0, Math.log(-1) renvoie NaN. si le nombre = 0, Math.log(0) renvoie -Infinity. Math.log(1) renvoie 0. Math.log(10) renvoie 2.302585092994046. Math.exp, Math.pow
max	Renvoie le plus grand nombre entre 2. **Syntaxe** : max (n1, n2) **Paramètres** : n1, n2 : 2 nombres Math.min
min	Renvoie le plus petit nombre entre 2. **Syntaxe** : min (n1, n2) **Paramètres** : n1, n2 : 2 nombres Math.max
pow	Renvoie $base^{exposant}$. **Syntaxe** : pow(x,y) **Paramètres** : x, y : 2 nombres ex.: Math.pow(4,2) = 16 Math.exp, Math.log
random	Renvoie un nombre aléatoire entre 0 et 1. Ce nombre est généré à partir de l'heure. **Syntaxe** : random() **Paramètres** : aucun ex.: x = Math.random()

round	Renvoie la valeur d'un nombre arrondi à l'entier le plus proche. **Syntaxe** : round(x) **Paramètres** : x : un nombre ex.: Math.round(10.49) = 10 Math.round(10.51) = 11 Math.round(-10.49) = -10 Math.round(-10.51) = -11
sin	Renvoie le sinus. **Syntaxe** : sin(x) **Paramètres** : x : un nombre Math.acos, Math.asin, Math.atan, Math.atan2, Math.cos, Math.tan
sqrt	Renvoie la racine carrée d'un nombre. **Syntaxe** : sqrt(x) **Paramètres** : x : un nombre ex.: sqrt(9) = 3
tan	Renvoie la tangente d'un nombre. **Syntaxe** : tan(x) **Paramètres** : x : un nombre Math.acos, Math.asin, Math.atan, Math.atan2, Math.cos, Math.sin

Chapitre 11 String

Une string est une chaîne de caractères.

Il ne faut pas oublier que l'index de départ d'une string est 0.
Dans "JavaScript" le "J" est à l'index 0, le "a" est à l'index 1, ...
Pour certaines méthodes, je parle d'expressions régulières, cliquez ici pour avoir la liste de ces expressions. Ce sont des expressions utilisées pour effectuer une recherche dans une string.

11.1 Propriétés

Propiétés	**Description**
length	Renvoie la longueur de la string. ex.: var s = "string" alert("la longueur est " + s.length)

11.2 Méthodes

Pour certaines méthodes, vous n'êtes pas obligé d'utiliser une variable de type string.
ex.:
var s = "JavaScript"
document.writeln(s.bold())
document.writeln("JavaScript".bold())
les 2 dernières lignes sont équivalentes.

Méthodes	**Description**
anchor	Crée une ancre HTML qui est utilisée comme une destination hypertext.

	A utiliser avec document.write ou document.writeln pour créer et afficher une ancre **Syntaxe** : anchor(nomAttribut) **Paramètres** : nomAttribut : une string. Le texte de la string représente la chaîne a afficher et nomAttribut représente l'attribut NAME du tag ex.: var s = "Table des matières" document.writeln(s.anchor("table_des_matières")) Cela correspond au tag : <A NAME="table_des_matières">Table des matières</A> String.link
big	Affiche la string en grande police de caractère. **Syntaxe** : big() **Paramètres** : aucun ex.: var s = "JavaScript" document.writeln(s.big()) donne : JavaScript Cela correspond au tag : <BIG>JavaScript</BIG> String.fontsize, String.small
blink	Affiche la string en mode clignotant. **Syntaxe** : blink() **Paramètres** : aucun ex.: var s = "JavaScript" document.writeln(s.big()) donne : JavaScript Cela correspond au tag : <BLINK>JavaScript</BLINK> String.bold, String.italics, String.strike
bold	Affiche la string en gras. **Syntaxe** : bold() **Paramètres** : aucun ex.: var s = "JavaScript" document.writeln(s.big()) donne **JavaScript** Cela correspond au tag : <B>JavaScript</B> String.blink, String.italics, String.strike
charAt	Renvoie un caractère choisit par vous de la string **Syntaxe** : charAt(index) **Paramètres** : index : un entier compris entre 0 et la longueur de la string. La première lettre correspond à l'index 0. Si index < 0, alors renvoie la première lettre. Si index > longueur de la string, alors renvoie une chaîne vide. String.indexOf, String.lastIndexOf, String.split
charCodeAt	Renvoie le code de la lettre choisie. **Syntaxe** : charCodeAt(index) **Paramètres** : index : un entier compris entre 0 et la longueur de la string. ex.: "JavaScript".charCodeAt(1) renvoie 97, qui est le code ASCII de "a".
concat	Concatène des strings. **Syntaxe** : concat(string2, string3[, ..., stringN]) **Paramètres** : string2, ..., stringN : des chaînes de caractères à concaténer à la string de départ.

	ex.: s1 = "Salut " s2 = "tout " s3 = "le monde." s4 = s1.concat(s2,s3) renvoie "Salut tout le monde."
fixed	Affiche la string avec une police de caractère de taille fixe. **Syntaxe** : fixed() **Paramètres** : aucun ex.: document.write("JavaScript".fixed()) donne : JavaScript Cela correspond au tag : <TT>JavaScript</TT>
fontcolor	Affiche la string dans la couleur choisie. **Syntaxe** : fontcolor(couleur) **Paramètres** : couleur : une string représentant le nom de la couleur ou la valeur hexadécimale en mode RGB (Red Green Blue : chaque valeur va de 00 à FF) ex.: document.write("JavaScript".fontcolor("red")) = JavaScript Cela correspond au tag : <FONT COLOR="red">JavaScript</FONT> document.write("JavaScript".fontcolor("coral")) = JavaScript Cela correspond au tag : <FONT COLOR="coral">JavaScript</FONT> document.write("JavaScript".fontcolor("FF00FF")) = JavaScript Cela correspond au tag : <FONT COLOR="FF00FF">JavaScript</FONT>
fontsize	Affiche une string avec une certaine taille. **Syntaxe** : fontsize(taille) **Paramètres** : taille : un entier ou une string représentant un entier de 1 à 7. ex.: document.write("JavaScript".fontsize(5)) = JavaScript document.write("JavaScript".fontsize("2")) = JavaScript Cela correspond au tag : <FONT SIZE=taille>JavaScript</FONT> String.big, String..small
fromCharCode	Renvoie une string créée à partir des valeurs ASCII des lettres. **Syntaxe** : String.fromCharCode(n1 [, n2, ..., nN]) **Paramètres** : n1, ..., nN : une entier > 1 ex.: String.fromCharCode(97, 98, 99) renvoie "abc"
indexOf	Renvoie l'index de la première occurence d'une sous-chaîne dans une chaîne. -1 si pas trouvé. **Syntaxe** : indexOf(chaîne[, index]) **Paramètres** : • chaîne : c'est la chaîne à rechercher • index : index à partir de où rechercher Ex.: "JavaScript de Netscape".indexOf("Java") renovie 0 "JavaScript de Netscape".indexOf("Script", 2) renvoie 4 "JavaScript de Netscape".indexOf("Script", 5) renvoie -1 "JavaScript de Netscape".indexOf(" ") renvoie 10 Cette méthode fait la distinction entre les majuscules et les minuscules : "JavaScript de Netscape".indexOf("script", 5) renvoie -1

	String.charAt, String.lastIndexOf, String.split
italics	Affiche le texte en italic. **Syntaxe** : italics() **Paramètres** : aucun Ex.: document.write("Javascript".italics) = *JavaScript* Cela correspond au tag : <I>JavaScript</I> String.blink, String.bold, String.strike
lastIndexOf	Renvoie l'index de la dernière occurence d'une sous-chaîne dans une chaîne. -1 si pas trouvé. La sous-chaîne est recherchée de la fin de la chaîne au début. **Syntaxe** : lastIndexOf(chaîne[, index]) **Paramètres** : • chaîne : c'est la chaîne à rechercher • index : index à partir de où rechercher "JavaScript de Netscape".lastIndexOf("Java") renovie 0 "JavaScript de Netscape".lastIndexOf("Script", 2) renvoie -1 "JavaScript de Netscape".lastIndexOf("Script", 10) renvoie 4 "JavaScript de Netscape".lastIndexOf(" ") renvoie 13 "JavaScript de Netscape".lastIndexOf("a") renvoie 19 Cette méthode fait la distinction entre les majuscules et les minuscules : "JavaScript de Netscape".indexOf("script", 5) renvoie -1 String.charAt, String.indexOf, String.split
link	Crée un lien hypertexte vers une URL. **Syntaxe** : link(lienHREF) **Paramètres** : lienHREF : une chaîne qui contient une URL. ex.: document.write("Aller chez Netscape".link("http://home.netscape.com")) correspond au tag : <A HREF="http://home.netscape.com">Aller chez Netscape</A>
match	Utilisé pour rechercher une expression régulière dans une chaîne de caractères. **Syntaxe** : match(expression) **Paramètres** : expression : une expression régulière. Peut être un nom ou un littéral. **Description** : si vous voulez effectuer une recherche globale ou en ignorant les majuscules et minsucules, mettez g ou i. match renvoie tous les caractères trouvés dans la chaîne et renvoie null si il n'a pas trouvé. ex.: Je recherche tous les "s" dans la chaîne : s = "JavaScript de Netscape" resultat = s.match(/s/gi) document.write(resultat) donne : S,s Je recherche tous les "sc" an minuscule : s = "JavaScript de Netscape" resultat = s.match(/sc/g) document.write(resultat) donne : sc
replace	Cherche une correspondance entre une expression régulière et une chaîne et remplace la sous-chaîne trouvée par la nouvelle sous-chaîne. **Syntaxe** : • replace(expression, sous-chaîne) • replace(expression, fonction) **Paramètres** : • expression : une expression régulière. Peut être un nom ou un littéral.

	• sous-chaîne : la chaîne qui va remplacer l'ancienne. • fonction : une fonction qui sera appelée après que la correspondance soit trouvée. **Description** : si vous voulez effectuer une recherche globale ou en ignorant les majuscules et minsucules, mettez g ou i. ex.: document.write("JavaScript de Netscape".replace(/a/g,"A")) donne : JAvAScript de NetscApe
search	Recherche dans une chaîne la correspondance avec une expression régulière. Renvoie l'index de l'expression trouvée dans la chaîne, renvoie -1 si pas trouvé. **Syntaxe** : search(expression) **Paramètres** : expression : une expression régulière. Peut être un nom ou un littéral. ex.: "JavaScript de Netscape".search("hello") renvoie -1
slice	Extrait une partie de la chaîne de caractères. **Syntaxe** : slice(indexdebut[, indexfin]) **Paramètres** : • indexdebut : index de début de la recherche • indexfin : index de fin de la recherche ex.: document.write("JavaScript de Netscape".slice(2,-4)) renvoie "vaScript de Nets" document.write("JavaScript de Netscape".slice(10)) renvoie "de Netscape"
small	Affiche une string en petite police. **Syntaxe** : small() **Paramètres** : aucun ex.: document.write("JavaScript".small()) : JavaScript Cela correspond au tag : <SMALL>JavaScript</SMALL> String.big, String.fontsize
split	Divise une string en un tableau de strings. **Syntaxe** : split([separateur][,limite]) **Paramètres** : • separateur : le caractère à utiliser pour séparer la string. • limite : donne une limite au nombre de sous-chaîne à séparer ex.: s = "lundi,mardi,mercredi,jeudi,vendredi,samedi,dimanche" tab = s.split(",") for(i = 0;i < tab.length; i++) document.write(tab[i] + " - ") donne : lundi - mardi - mercredi - jeudi - vendredi - samedi - dimanche -
strike	Affiche la string barrée. **Syntaxe** : strike() **Paramètres** : aucun ex.: document.write("JavaScript".strike()) : ~~JavaScript~~ Cela correspond au tag : <STRIKE>JavaScript</STRIKE> String.blink, String.bold, String.italics
sub	Affiche la string en indice. **Syntaxe** : sub() **Paramètres** : aucun ex.: document.write("JavaScript".strike()) : _{JavaScript}

	Cela correspond au tag : <STRIKE>JavaScript</STRIKE> String.sup
substr	Renvoie une sous-chaîne commencant à l'index choisi et contenant un certain nombre de caractères. **Syntaxe** : substr(debut, longueur) **Paramètres** : • debut : index de début à partir duquel on extrait la chaîne • longueur : nombre de caractères à extraire ex.: document.write("JavaScript".substr(1,2)) donne : av document.write("JavaScript".substr(-1,2)) donne : Ja document.write("JavaScript".substr(4)) donne : Script document.write("JavaScript".substr(15,2)) donne : String.substring
substring	Renvoie une sous-chaîne d'une string **Syntaxe** : substring(indexA, indexB) **Paramètres** : indexA, indexB : index entre 0 et longueur de la string - 1 ex.: document.write("JavaScript".substring(2,5)) donne : vaS document.write("JavaScript".substring(5,3)) donne : aS document.write("JavaScript".substring(0,15)) donne : JavaScript String.substr
sup	Affiche la string en exposant. **Syntaxe** : sup() **Paramètres** : auncun ex.: document.write("JavaScript".strike()) : JavaScript Cela correspond au tag : ^{JavaScript} String..sub
toLowerCase	Convertit la string en minuscules. **Syntaxe** : toLowerCase() **Paramètres** : aucun String.toUpperCase
toSource	Renvoie une string représentant le code source de l'objet. **Syntaxe** : toSource() **Paramètres** : aucun Cette méthode est appelée en interne dans le JavaScript et non dans le code
toString	Renvoie une string représentant l'objet. **Syntaxe** : toString() **Paramètres** : aucun
toUpperCase	Convertit la string en majuscules. **Syntaxe** : toUpperCase() **Paramètres** : aucun String.toLowerCase
valueOf	Renvoie la valeur primitive de l'objet **Syntaxe** : valueOf() **Paramètres** : aucun C'est équivalent à la méthode toString() String.toString

11.3 Les caractères spéciaux

Passage à la ligne suivante	\n
Tabulation horizontale	\t
Retour en arrière	\b
Retour chariot	\r
Saut de page	\f

Chapitre 12 L'objet "navigator"

Grâce aux objets du navigateur, on peut récolter des informations sur le navigateur de la personne qui vient visiter le site. Cela peut être utile lors de l'emploi de certaines méthodes du JavaScript qui ne sont implémentées qu'à partir d'une certaine version du navigateur ou encore pour pouvoir faire un traitement sur un navigateur et pas un autre. En effet, il existe des méthodes reconnues par Netscape et pas par Internet Explorer, et vice-versa.

Votre navigateur est	**Microsoft Internet Explorer**
Votre version est	**4.0**
Le code du navigateur est	**Mozilla**
Votre platforme est	**Win32**
Java activé?	**oui**
JavaScript	**1.3**
Votre OS	**Windows 98**

12.1 Propriétés

Propriétés	**Description**
appCodeName	Donne le code du navigateur. Ici, **Mozilla**
appName	Donne le nom du navigateur. Ici, **Microsoft Internet Explorer**
appVersion	Donne la version du navigateur. Ici, **4.0 (compatible; MSIE 5.0; Windows 98; DigExt)**
language	Indique la langue du navigateur. Ici, **undefined**
mimeTypes	Renvoie un tableau contenant tous les "mimes" supportés. L'objet mime a 4 propriétés : • description : description du mime. • enabledPlugin : sert à voir si il y a un plugin configuré pour le type de mime. • suffixes : liste des extensions pour le mime. ex.: "mpeg, mpg, mpe, mpv, vbs, mpegv" • type : le nom du type de mime. ex.: "video/mpeg" Liste des mimes
platform	Indique le type de machine pour lequel le navigateur a été compilé. Ici, **Win32**
plugins	Renvoie un tableau contenant tous les "plugins" installés.

	L'objet plugin a 4 propriétés : • description : description du plug-in. • filename : nom du plug-in sur le disque. • length : nombre d'éléments dans le tableau de plug-in de l'objet mime. • name : le nom du type du plug-in. Liste des plugins
userAgent	Donne l'en-tête du "user-agent" du navigateur. Ici, **Mozilla/4.0 (compatible; MSIE 5.0; Windows 98; DigExt)**

12.2 Méthodes

Méthodes	**Description**
javaEnabled	Teste si le Java est activé.
plugins.refresh	Rends les nouveau plug-ins installés disponibles. ex.: navigator.plugins.refresh(true)
preference	Permet à un script de prendre et de mettre certaines préférences du navigateur. **Syntaxe** : preference(nom[, valeur]) **Paramètres** : • nom : une string représentant le nom de la préférence voulue. • valeur : string, nombre ou booléen Liste des préférences disponibles

12.3 Liste des préférences disponibles

uniquement pour Netscape

But	**Préférences**	**Valeurs**
Automatically load images	general.always_load_images	true ou false
Enable Java	security.enable_java	true ou false
Enable JavaScript	javascript.enabled	true ou false
Enable style sheets	browser.enable_style_sheets	true ou false
Enable SmartUpdate	autoupdate.enabled	true ou false
Accept all cookies	network.cookie.cookieBehavior	0
Accept only cookies that get sent back to the originating server	network.cookie.cookieBehavior	1
Disable cookies	network.cookie.cookieBehavior	2
Warn before acception cookies	network.cookie.warnAboutCookies	true ou false

Exemple 1

Script qui permet de voir si on peut jouer un fichier midi (pour Netscape)

```
<script LANGUAGE="JavaScript">
<!--
mimetype = navigator.mimeTypes["audio/x-midi"]
if (mimetype) {
//Oui. Teste si le plug-in est disponible
plugin = mimetype.enabledPlugin
if (plugin)
//Oui
document.writeln("Voici un fichier midi: <EMBED SRC='fichier.mid' HEIGHT=60 WIDTH=150>")
else
//Non
document.writeln("<A HREF='fichier.mid'>Cliquez ici</A>.")
}
else {
//Non, petit message
document.writeln("Désolé, vous ne pouvez pas l'écouter.")
}
//-->
</script>
```

Exemple 2

Script de détection du navigateur

```
<script LANGUAGE="JavaScript">
<!--
function VersionNavigateur(Netscape, Explorer) {
//vérifie la version du navigateur
//usage VersionNavigateur(3.0,4.0)
if ((navigator.appVersion.substring(0,3) >= Netscape &&
navigator.appName == 'Netscape') ||
(navigator.appVersion.substring(0,3) >= Explorer &&
navigator.appName.substring(0,9) == 'Microsoft'))
return true;
else return false;
}
function isExplorer() {
//renvoie true si le navigateur est MSIE
return navigator.appName.indexOf("Explorer") != -1;
}
function isNetscape() {
//renvoie true si le navigateur est Netscape
return navigator.appName.indexOf("Netscape") != -1;
}
//-->
</script>
```

Chapitre 13 Exemples

13.1 Accéder aux propriétés du document

Le titre de ce document est "Accéder aux propriétés du document"

Cette page a été modifiée le 05/30/2000 19:03:18
L'URL de cette page est file:///D:/Mes documents/info/javascript/wphilippe/js_document.html
La page ou vous étiez avant :
La couleur de fond est #ffffff
La couleur du texte est #000000
La couleur des liens est #0000ff
La couleur des liens visités est #800080
La couleur du lien actif est #0000ff

```
<script LANGUAGE="JavaScript">
<!--
document.write("Le titre de ce document est \"" + document.title + "\"<BR>")
document.write("Cette page a été modifiée le " + document.lastModified + "<BR>")
document.write("L'URL de cette page est " + unescape(location.href) + "<BR>")
document.write("La page ou vous étiez avant : " + document.referrer + "<BR>")
document.write("La couleur de fond est " + document.bgColor + "<BR>")
document.write("La couleur du texte est " + document.fgColor + "<BR>")
document.write("La couleur des liens est " + document.linkColor + "<BR>")
document.write("La couleur des liens visités est " + document.vlinkColor + "<BR>")
document.write("La couleur du lien actif est " + document.alinkColor + "<BR>")
//-->
</script>
```

On peut aussi changer ces propriétés :

Par exemple, le fond d'écran :

```
document.bgColor='DEB887';
ou
document.bgColor='cyan';
```

13.2 Afficher la date du jour

Aujourd'hui, nous sommes le 10 Juillet 2001, il est 16:30

```
<script LANGUAGE="JavaScript">
<!--
var d = new Date();

//un tableau contenant les noms des mois
var tabMois = new Array("Janvier", "Fevrier", "Mars", "Avril", "Mai", "Juin", "Juillet", "Août",
"Septembre", "Octobre", "Novembre", "Decembre");

document.write("Aujourd'hui, nous sommes le " + d.getDate() + " " + tabMois[d.getMonth()] + " "
+ d.getFullYear());
document.write(", il est " + d.getHours() + ":" + d.getMinutes());
//-->
</script>
```

13.3 Afficher un message en alternance avec d'autres

...permet d

```
Erreur! Signet non défini.<html>
<head>
<title> </title>
<script LANGUAGE="JavaScript">
<!--
//nombre de messages
var nbMsg = 5;
//tableau contenant les messages
var messages = new Array(nbMsg);
messages[0] = "Un message de ce type...";
messages[1] = "...permet de faire passer des idées...";
messages[2] = "...ou des informations...";
messages[3] = "...d'une autre façon que le scrolling de texte...";
messages[4] = "...qui n'est pas toujours facile à lire!";
//numéro du message actuellement affiché
var numMsg = 0;
//temps d'afichage de chaque messages
var pause = 2500;
var timeID = null;
function AfficheMsg() {
document.txtAlter.msg.value = messages[numMsg++];
if (numMsg >= nbMsg) numMsg = 0;
timerID = setTimeout("AfficheMsg()", pause);
}
function StopTimer() {
clearTimeout(timerID);
}
```

```
//-->
</script>
</head>

<body onLoad="AfficheMsg()" onUnLoad="StopTimer()">

<center>
<form name="txtAlter">
<input type="text" size="50" name="msg" style="font-family: 'Comic Sans MS', 'Courrier
New';font-size: 12pt; color: #000000; background-color: #00ffff">
</form>
</center>

</body>
</html>
```

13.4 Afficher un message en fonction de l'heure

Afficher un message en fonction de l'heure de la journée : " C'est l'après-midi "

```
<script LANGUAGE="JavaScript">
<!--
var d = new Date();
//tableau contenant les messages
var tab = new Array();
tab[0] = "Il est minuit passé";
tab[1] = "C'est le matin, je vous souhaite une bonne journée";
tab[2] = "Midi, bon appétit"
tab[3] = "C'est l'après-midi"
tab[4] = "Message pour la nuit"&

var heure = d.getHours();

if ((heure >= 0) && (heure <= 6)){
document.write(tab[0]);
}
else if ((heure >= 7) && (heure < 12)){
document.write(tab[1]);
}
else if (heure == 12) then{
document.write(tab[2]);
}
else if ((heure > 12) && (heure < 19)){
document.write(tab[3]);
}
else document.write(tab[4]);
//-->
</script>
```

13.5 Afficher un message en fonction du jour de la semaine

Le message du jour : " Message pour le Mardi "

```
<script LANGUAGE="JavaScript">
<!--
var d = new Date();

//tableau contenant les messages
var tab = new Array(7);
tab[0] = "Message pour le Dimanche";
tab[1] = "Message pour le Lundi";
tab[2] = "Message pour le Mardi";
tab[3] = "Message pour le Mercredi";
tab[4] = "Message pour le Jeudi";
tab[5] = "Message pour le Vendredi";
tab[6] = "Message pour le Samedi";

document.write(tab[d.getDay()]);
//-->
</script>
```

13.6 Afficher une image aléatoire

Le script ci-dessous est prévu pour 10 images dont les noms vont de "b0" à "b9".

```
<html>
<head>
<title>Images aléatoires</title>
</head>
<body>
<script LANGUAGE="JavaScript">
<!--
var n = parseInt(Math.random()*10);
//avec Netscape, Math.random renvoie parfois NaN (Not a number)
//le but de la boucle est de recommencer afin d'obtenir un nombre si n est égal à NaN
while (isNaN(n)) {
n = parseInt( Math.random() * 10)
}

document.write("<img src='images/b" + n + ".gif' width=15 height=15 alt='Image aléatoire'>");
//-->
</script>

</body>
</html>
```

- **parseInt** renvoie un nombre entier à partir du nombre renvoyé par Math.random()
- **Math.random()*10** permet de choisir un nombre aléatoire entre 0 et 10.

```
<html>
<head>
```

```
<title>Images aléatoires</title>
</head>
<script LANGUAGE="JavaScript">
<!--
document.write("<body>");
var tab = new Array("aaa.gif","bbb.gif","ccc.gif");
//remplacez aaa, bbb et ccc par vos images

var n = parseInt( Math.random() * tab.length)
while (isNaN(n)) {
n = parseInt( Math.random() * tab.length)
}

document.write("<img src='images/" + tab[n] + "' alt='Image aléatoire'>");
//-->
</script>

</body>
</html>
```

Il est aussi possible de changer aléatoirement l'image de fond de la page.
Il suffit alors de remplacer dans le script **document.write("<body>");** par
document.write("<body background='nom_image.gif'>");

13.7 Afficher une barre avec un dégradé de couleurs

Degrade(255,0,0,0,0,255,50,'50%');

Degrade(255,0,0,0,255,0,50,'200');

Degrade(255,230,80,255,255,255,50,'500');

Degrade(100,100,50,200,255,0,50,'100%');

```
<html>
<head>
<title>Texte multi-couleur</title>
<script LANGUAGE="JavaScript">
<!--
//construction du tableau des valeurs hexadécimales
var hexa = new Array(16);
for(var i = 0; i < 10; i++) hexa[i]=i;
hexa[10] = "A";
hexa[11] = "B";
```

```
hexa[12] = "C";
hexa[13] = "D";
hexa[14] = "E";
hexa[15] = "F";
function toHexa(n) {
//fonction de conversion de décimal à hexadécimal
if (n < 0) return "00";
if (n > 255) return "FF";
return "" + hexa[Math.floor(n/16)] + hexa[Math.floor(n%16)];
}
function Degrade(dr,dg,db,fr,fg,fb,nb, taille) {
//paramètres :
// dR est la couleur rouge de début
// dG est la couleur verte de début
// dB est la couleur bleue de début
// fR est la couleur rouge de fin
// fG est la couleur verte de fin
// fB est la couleur bleue de fin
// nb est la précision du dégradé.
// Au plus le nombre est élevé, au plus le dégradé sera précis.
// taille : taille de la barre : en pourcentage (par rapport à la fenêtre) ou un nombre
cr = dr;
cg = dg;
cb = db;
sr = (fr - dr) / nb;
sg = (fg - dg) / nb;
sb = (fb - db) / nb;
document.write('<table BORDER="0" CELLPADDING="0" CELLSPACING="0" WIDTH="' +
taille + '" COLS="1">');
document.write('<tr>');
for (var x = 0; x <= nb; x++) {
document.write('<TD WIDTH="1" BGCOLOR="#' + toHexa(cr) + toHexa(cg) + toHexa(cb) +
'"> </TD>');
cr += sr; cg += sg; cb += sb;
}
document.write('</tr>');
document.write('</table>');
}
Degrade(100,100,50,200,255,0,50);
//-->
</script>
</head>

<body>

</body>
</html>
```

Mettre des barres trop longues ralentit la vitesse d'affichage de la page.

13.8 Un bouton pour imprimer une page

Imprimer la

```
Erreur! Signet non défini.<html>
<head>
<script LANGUAGE="JavaScript">
<!--
function printit(){
if (NS) {
window.print();
} else {
var WebBrowser = '<OBJECT ID="WebBrowser1" WIDTH=0 HEIGHT=0
CLASSID="CLSID:8856F961-340A-11D0-A96B-00C04FD705A2"></OBJECT>';
document.body.insertAdjacentHTML('beforeEnd', WebBrowser);

WebBrowser1.ExecWB(6, 2);//Use a 1 vs. a 2 for a prompting dialog box
WebBrowser1.outerHTML = "";
}
}
//-->
</script>
</head>

<body>

<script language="Javascript">
<!--
var NS = (navigator.appName == "Netscape");
var VERSION = parseInt(navigator.appVersion);

if (VERSION > 3) {
document.write('<form><input type=button value="Imprimer la page" name="Print"
onClick="printit()"></form>');
}
//-->
</script>

</body>
</html>
```

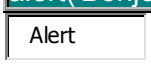
13.9 Boîtes de dialogue

Il y a 3 boîtes de dialogue :

Alert

Affiche un simple message. L'utilisateur n'a qu'un seul choix : appuyer sur "OK"
alert('votre texte')

```
alert('Bonjour');
```



Il y a moyen de formatter le texte à afficher :
passer à la ligne : \n

```
alert('Bonjour\n'+
'ceci est un exemple\n'+
'pour passer à la ligne');
```

Alert

---

mettre une tabulation : \t

```
alert('Bonjour\tceci est un exemple\tde tabulation');
```

Alert

Confirm

A utiliser quand l'utilisateur est confronté 2 choix.
confirm('votre texte')

```
if (confirm('Exemple de confirmation')){
alert('Vous avez appuyé sur OK');
} else {
alert('Vous avez appuyé sur CANCEL');
}
```

Confirm

Prompt

A utiliser quand l'utilisateur doit entrer du texte.
prompt('votre texte','valeur par défaut')
Si vous annulez la commande, prompt renvoie null.

```
nom = prompt('Entrez votre nom', 'Votre nom');
```

Prompt

```
s = prompt('texte à afficher', 'chaîne par défaut');
alert('vous avez tapé \'" + s + '\"');
if (s == null) alert('Vous ne vouliez pas me croire quand je disais que c\'était null.');
```

Prompt

```
s = 'Hello';
s = prompt('texte à afficher', s);
if (s == null){
alert('vous avez appuyé sur cancel');
} else {
alert('vous avez tapé \'" + s + '\"');
}
```

13.10 Calculer le temps de lecture d'une page

```
<html>
<head>
<title>Temps de lecture d'une page</title>
<script LANGUAGE="JavaScript">
<!--
var Debut = new Date();

function Affiche() {
var Fin = new Date();
//nombre de millisecondes entres le chargement de la page
//et la fin de la lecture
var mmsec = Fin.valueOf() - Debut.valueOf();
mmsec = Math.round(mmsec / 1000);

//transformation des millisecondes en minutes et secondes
var mm, ss;
mm = Math.floor(mmsec / 60);
ss = mmsec - mm * 60;

//construction du message
var temps = "Vous avez passé ";
temps += (mm == 1? mm + " minute ":mm + " minutes ");
temps += (ss == 1? ss + " seconde ":ss + " secondes ");
temps += " sur cette page...";
alert(temps);
}

//-->
</script>
</head>

<body onUnload="Affiche()">

</body>
</html>
```

13.11 Les cookies

Qu'est-ce que c'est?

Les cookies permettent au browser de stocker des informations de manière à pouvoir les récupérer lors de la prochaine viste.

Il faut bien se rappeler que les cookies sont stockés sur votre ordinateur et pas sur le serveur.

Exemples d'éléments à mettre dans un cookie :

- garder le nom de l'utilisateur
- garder la langue de l'utilisateur
- garder le mot de passe

...

Comment cela fonctionne?

L'objet qui permet de lire les cookies est "document.cookie". Si un cookie est présent, son contenu sera automatiquement mis dans cet objet.

Les cookies ont un paramètre obligatoire : **nom=valeur**.

Par défaut, le cookie dure le temps de la session sauf si vous mettez une date d'expiration : **expires=date**. La date doît être convertie en utilisant l'heure de Greenwich grâce à la méthode **toGMTString**.

Tous les paramètres d'un cookie sont séparés par un point-virgule (;).
Exemple : nom=valeur;expires=date

Le code
Pour utiliser les cookies, vous devez avoir 3 fonctions :

1. une fonction qui crée le cookie : setCookie
2. une fonction qui lit les informations du cookie : getCookie
3. une fonction qui efface le cookie : delCookie

```
<script LANGUAGE="JavaScript">
<!--
function setCookie(nom, valeur, jours){
//nom est le nom du cookie
//valeur est la valeur a stocker
//jours est le nombre de jours avant l'expiration du cookie

var expireDate = new Date();
expireDate.setTime(expireDate.getTime() + (jours * 24 * 3600 * 1000));
//création du cookie
document.cookie = nom + "=" + escape(valeur) + ";expires=" +
expireDate.toGMTString();
}
function getCookie(nom){
//on vérifie si il y a un cookie
if (document.cookie.length > 0){
debut = document.cookie.indexOf(nom + "=");
//on vérifie si la valeur qu'on recherche est dans le cookie
if (debut != -1) //!= veut dire différent
{
debut += nom.length + 1;
fin = document.cookie.indexOf(";", debut);
if (fin == -1) fin = document.cookie.length;
return unescape(document.cookie.substring(debut, fin));
}
}
return null;
//la valeur n'a pas été trouvée...
}
function delCookie(nom){
if (getCookie(nom)){
document.cookie = nom + "=;expires=Thu, 01-Jan-70 00:00:01 GMT";
//en mettant cette date, le cookie sera désactivé
}
```

```
}
//-->
</script>
```

13.12 Date de dernière modification

Afficher la date de dernière modification peut permettre de faire savoir aux visiteurs quand le site a été mis à jour.

Vous pouvez évidemment taper à chaque fois cette date, mais c'est un peu fastidieux.

JavaScript vous permet de le faire une fois pour toutes.

Dernière date de mise à jour : 30 Mai 2000 à 19:03

Comment cela fonctionne?

Le JavaScript permet de récupérer la date de dernière modification d'un fichier grâce à la fonction "document.lastModified". C'est cette date que nous allons utiliser.

Le code

Ce code est à inclure dans le code de la page, à l'endroit ou vous voulez mettre la date.

```
<script LANGUAGE="JavaScript">
<!--

//on prend la date du fichier
var lastMod = document.lastModified;

//un tableau contenant les noms des mois
var tabMois = new Array("Janvier", "Fevrier", "Mars", "Avril", "Mai", "Juin", "Juillet", "Août",
"Septembre", "Octobre", "Novembre", "Decembre");

//on construit un objet de type date avec la date du fichier
var lastDate = new Date(lastMod);
var annee = lastDate.getFullYear();
var hh = lastDate.getHours();
var mm = lastDate.getMinutes();

var heure = (hh > 9? hh:"0" + hh);
heure += ":" + (mm > 9? mm:"0" + mm);

document.write("<center>Dernière date de mise à jour : ");
document.write(lastDate.getDate() + " ");
document.write(tabMois[lastDate.getMonth()] + " ");
document.write(annee + " ");
document.write(" à " + heure + "</center>");
//-->
</script>
```

13.13 Déterminer si le navigateur supporte le JavaScript

Votre navigateur supporte le JavaScript...

Enlevez le support JavaScript de votre navigateur et rechargez la page pour voir la différence.

Votre navigateur ne supporte pas le JavaScript...

```
<html>
<head>
<title>Déterminer si le navigateur supporte le JavaScript</title>
</head>

<body>
<script LANGUAGE="JavaScript">
<!--
document.writeln("<font color=#FF0000>Votre navigateur supporte le
JavaScript...</font><br>");
document.writeln("Enlevez le support JavaScript de votre navigateur et rechargez la page");
//-->
</script>

<noscript>
<font color=#FF0000>Votre navigateur ne supporte pas le JavaScript...</font>
</noscript>

</body>
</html>
```

<noscript> et **</noscript>** : c'est entre ces 2 tags que vous devez mettre un traitement (comme la redirection vers une autre page) pour les navigateurs qui ne supportent pas le JavaScript.

13.14 Détecter la version du JavaScript

Connaître la version du JavaScript peut se révéler utile afin de prévoir un traitement alternatif si les fonctions utilisées ne peuvent être exécutées.

```
<script language="JavaScript">
<!--
var JSVersion=1.1;
//-->
</script>
<script language="JavaScript1.2">
<!--
JSVersion=1.2;
//-->
</script>
<script
language="JavaScript1.3">
<!--
JSVersion=1.3;
```

```
//-->
</script>
<script language="JavaScript1.4">
<!--
JSVersion=1.4;
//-->
</script>

<script language="JavaScript">
<!--
document.writeln("La version de votre JavaScript est : " + JSVersion);
//-->
</script>
```

13.15 Ecrire dans la barre de status

Ecrire dans la barre de status est utile lorsqu'on veut mettre un texte explicatif pour un lien ou une image.

Exemple **:**

Mettez le curseur de la souris sur le lien et regardez dans la barre de status pour voir « Votre descriprtion du lien ».
ex : votre lien

```
<a href="page.html" onMouseOver="window.status='Votre description du lien'; return true;"
onMouseOut="window.status=''; return true;">votre lien</a>
```

- **onMouseOver** est l'événement qui intervient lors du passage de la souris sur le lien.
- **onMouseOut** est l'événement qui intervient lorsque vous quittez le lien. C'est ici que vous devez effacer le contenu de la barre de status.
- **window.status** permet d'écrire dans la barre de status.
- **return true;** est indispensable.

13.16 Quelle est la résolution de l'écran?

Valable pour la version 4.0 et plus du navigateur et pour la version 1.2 du JavaScript.

```
<script language="JavaScript">
<!--
document.write("<p>Votre écran est en ",screen.width," x ",screen.height,"<p>");
//-->
</script>
```

13.17 Stopper les erreurs du JavaScript

(Pour Netscape) Cela permet d'effectuer un traitement lorsqu'une erreur survient dans le code. On peut afficher un texte par exemple. C'est utile pour vérifier le code de vos fonctions afin de voir s'il y a une erreur, mais l'inconvénient est qu'on n'a aucune informations sur l'endroit d'où l'erreur est partie.

```
<script language="JavaScript">
<!--
window.onError = stopErreur;
function stopErreur() {
//votre code
return true;
}
//-->
</script>
```

13.18 Mettre le code JavaScript dans un fichier externe

Mettre le code dans un fichier externe a plusieurs avantages :

- cela permet de pouvoir mettre à jour le code une seule fois et de ne pas devoir le modifier dans chaque page ou vous l'avez mis.
- cela permet aussi de réduire la taille des fichiers HTML car si vous utilisez une même fonction sur plusieurs pages, le code ne sera écrit qu'une seule fois et ne sera plus présent dans les fichiers HTML.
-

Le code doit être mis dans un fichier ayant l'extension **.JS**

Une fois le code tapé, il suffit de rajouter le tag **src="fichier.js"** dans le tag **<script>** et d'appeler les fonctions comme d'habitude.

Fichier.JS

```
var variable1;
var variable2;
function fonction1() {
//code
}
function fonction2() {
//code
}
```

Page HTML

```
<script LANGUAGE="JavaScript" SRC="fichier.js"></script>
<script LANGUAGE="JavaScript">
<!--

//appel de la fonction "fonction1"
variable2 = fonction1();

//-->
</script>
```

13.19 Ajouter la page dans les favoris

(Pour MSIE seulement)

```
Ajoutez mon site dans vos <a
href="javascript:window.external.AddFavorite('http://adresse_de_votre_site','Titre')">favorits</a
>
```

Chapitre 14 Les événements

Afin de fournir plus d'interactivité entre l'utilisateur et le navigateur, il existe une série d'événements.
Ils permettront de fournir un traitement approprié en fonction du choix de l'utilisateur.

14.1 Généralités

Avec les événements et surtout leur gestion, nous abordons le côté "magique" de Javascript.

En Html classique, il y a un événement que vous connaissez bien. C'est le clic de la souris sur un lien pour vous transporter sur une autre page Web. Hélas, c'est à peu près le seul. Heureusement, Javascript va en ajouter une bonne dizaine.

Les événements Javascript, associés aux fonctions, aux méthodes et aux formulaires, ouvrent la porte pour une réelle interactivité.

14.2 Les événements

Les différents événements implémentés en Javascript sont...

Description	Evénement
Lorsque l'utilisateur clique sur un bouton, un lien ou tout autre élément.	Clik
Lorsque la page est chargée par le browser ou le navigateur.	Load
Lorsque l'utilisateur quitte la page.	Unload
Lorsque l'utilisateur place le pointeur de la souris sur un lien ou tout autre élément.	MouseOver
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).	MouseOut
Lorsque un élément de formulaire a le focus c-à-d devient la zone d'entrée active.	Focus
Lorsque un élément de formulaire perd le focus c-à-d que l'utilisateur clique hors du champs et que la zone d'entrée n'est plus active.	Blur
Lorsque la valeur d'un champ de formulaire est modifiée.	Change
Lorsque l'utilisateur sélectionne un champ dans un élément de formulaire.	Select
Lorsque l'utilisateur clique sur le bouton Submit pour envoyer un formulaire.	Submit

14.3 Les gestionnaires d'événements

Pour être efficace, il faut qu'à ces événements soient associées les actions prévues par vous. C'est le rôle des gestionnaires d'événements. La syntaxe est

onévénement="fonction()"

Par exemple, onClick="alert('Vous avez cliqué sur cet élément')".
De façon littéraire, au clic de l'utilisateur, ouvrir une boite d'alerte avec le message indiqué.

onclick

Evénement classique en informatique, le clic de la souris.

Le code de ceci est :

```
<FORM>
<INPUT TYPE="button" VALUE="Cliquez ici" onClick="alert('Vous avez bien cliqué ici')">
</FORM>
```

onLoad et onUnload

L'événement Load survient lorsque la page a fini de se charger. A l'inverse, Unload survient lorsque l'utilisateur quitte la page.

Les événements onLoad et onUnload sont utilisés sous forme d'attributs de la balise <BODY> ou <FRAMESET>. On peut ainsi écrire un script pour souhaiter la bienvenue à l'ouverture d'une page et un petit mot d'au revoir au moment de quitter celle-ci.

```
<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()'>
Html normal
</BODY>
</HTML>
```

onmouseOver et onmouseOut

L'événement onmouseOver se produit lorsque le pointeur de la souris passe au dessus (sans cliquer) d'un lien ou d'une image. Cet événement est fort pratique pour, par exemple, afficher des explications soit dans la barre de statut soit avec une petite fenêtre genre infobulle.

L'événement onmouseOut, généralement associé à un onmouseOver, se produit lorsque le pointeur quitte la zone sensible (lien ou image).
Notons qui si onmouseOver est du Javascript 1.0, onmouseOut est du Javascript 1.1.
En clair, onmouseOut ne fonctionne pas avec Netscape 2.0 et Explorer 3.0.

onFocus

L'événement onFocus survient lorsqu'un champ de saisie a le focus c.-à-d. quand son emplacement est prêt à recevoir ce que l'utilisateur à l'intention de taper au clavier. C'est souvent la conséquence d'un clic de souris ou de l'usage de la touche "Tab".

onBlur

L'événement onBlur a lieu lorsqu'un champ de formulaire perd le focus. Cela se produit quand l'utilisateur ayant terminé la saisie qu'il effectuait dans une case, clique en dehors du champ ou utilise la touche "Tab" pour passer à un champ. Cet événement sera souvent utilisé pour vérifier la saisie d'un formulaire.

Le code est :
```
<FORM>
<INPUT TYPE=text onBlur="alert('Ceci est un Blur')">
</FORM>
```

onchange

Cet événement s'apparente à l'événement onBlur mais avec une petite différence. Non seulement la case du formulaire doit avoir perdu le focus mais aussi son contenu doit avoir été modifié par l'utilisateur.

onselect

Cet événement se produit lorsque l'utilisateur a sélectionné (mis en surbrillance ou en vidéo inverse) tout ou partie d'une zone de texte dans une zone de type text ou textarea.

Gestionnaires d'événement disponibles en Javascript

Voici la liste des objets auxquels correspondent des gestionnaires d'événement bien déterminés.

Objets	Gestionnaires d'événement disponibles
Fenêtre	onLoad, onUnload
Lien hypertexte	onClick, onmouseOver, on mouseOut
Elément de texte	onBlur, onChange, onFocus, onSelect
Elément de zone de texte	onBlur, onChange, onFocus, onSelect
Elément bouton	onClick
Case à cocher	onClick
Bouton Radio	onClick
Liste de sélectionon	Blur, onChange, onFocus
Bouton Submit	onClick
Bouton Reset	onClick

14.4 La syntaxe de onmouseOver

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

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

Ainsi, lorsque l'utilisateur passe avec sa souris sur le lien, la fonction action() est appelée. L'attribut HREF est indispensable. Il peut contenir l'adresse d'une page Web si vous souhaitez que le lien soit actif ou simplement des guillemets si aucun lien actif n'est prévu. Nous reviendrons ci-après sur certains désagréments du codage HREF="".

Voici un exemple. Par le survol du lien "message important", une fenêtre d'alerte s'ouvre.

Le code est :
```
<BODY>
...
<A HREF="" onmouseOver="alert('Coucou')">message important</A>
...
<BODY>
```

ou si vous préférez utiliser les balises <HEAD>
```
<HTML>
<HEAD>
<SCRIPT language="Javascript">
function message(){
alert("Coucou")
}
</SCRIPT>
</HEAD>
<BODY>
<A HREF="" onmouseOver="message()">message important</A>
</BODY>
</HTML>
```

14.5 La syntaxe de onmouseOut

Tout à fait similaire à onmouseOver, sauf que l'événement se produit lorsque le pointeur de la souris quitte le lien ou la zone sensible.

Au risque de nous répéter, si onmouseOver est du Javascript 1.0 et sera donc reconnu par tous les browsers, onmouseOut est du Javascript 1.1 et ne sera reconnu que par Netscape 3.0 et plus et Explorer 4.0 et plus (et pas par Netscape 2.0 et Explorer 3.0)

On peut imaginer le code suivant :
```
<A HREF="" onmouseOver="alert('Coucou')" onmouseOut="alert('Au revoir')">message important</A>
```

Les puristes devront donc prévoir une version différente selon les versions Javascript.

Problème! Et si on clique quand même...

Vous avez codé votre instruction onmouseOver avec le lien fictif <A HREF=""... >, vous avez même prévu un petit texte, demandant gentiment à l'utilisateur de ne pas cliquer sur le lien et comme de bien entendu celui-ci clique quand même.
Horreur, le browser affiche alors l'entièreté des répertoires de sa machine ou de votre site. Ce qui est un résultat non désiré et pour le moins imprévu.

Pour éviter cela, prenez l'habitude de mettre l'adresse de la page encours ou plus simplement le signe # (pour un ancrage) entre les guillemets de HREF. Ainsi, si le lecteur clique quand même

sur le lien, au pire, la page encours sera simplement rechargée et sans perte de temps car elle est déjà dans le cache du navigateur.
Prenez donc l'habitude de mettre le code suivant <A HREF="#" onmouseOver="action()"> lien </A>.

Changement d'images

Avec le gestionnaire d'événement onmouseOver, on peut prévoir qu'après le survol d'un image par l'utilisateur, une autre image apparaisse (pour autant qu'elle soit de la même taille).
le code est relativement simple.

```
<HTML>
<HEAD>
<SCRIPT LANGUAGE="Javascript1.1">
function lightUp() {
document.images["homeButton"].src="button_hot.gif"
}
function dimDown() {
document.images["homeButton"].src="button_dim.gif"
}
</SCRIPT>
</HEAD>
<BODY>
<A HREF="#" onmouseOver="lightUp();" onmouseOut="dimDown();">
<IMG SRC="button_dim.gif" name="homeButton" width=100 height=50 border=0> </A>
</BODY>
</HTML>
```

Compléter toujours en Javascript les attributs width=x height=y de vos images.

Il n'y a pas d'exemple ici pour la compatibilité avec les lecteurs utilisant explorer 3.0 en effet, non seulement onmouseOut mais aussi image[] est du Javascript 1.1.

Voir : images Rollover - Dreamweaver

14.6 L'image invisible

On peut prévoir une image invisible de la même couleur que l'arrière plan (même transparente). On la place avec malice sur le chemin de la souris de l'utilisateur et son survol peut ,à l'insu de l'utilisateur, déclencher un feu d'artifice d'actions de votre choix. Magique le Javascript ?

```
Erreur! Signet non défini.<form onSubmit="alert('Evénement onSubmit');return false;">
<p>
<img src="../images/lamp.gif" width="16" height="16"
alt="événement onMouseOver et onMouseOut" onMouseOver="alert('Evénement
onMouseOver')" onMouseOut="alert('Evénement onMouseOut')" onClick="alert('Evénement
onClick')">
<input type="text" name="T1" value="Sélectionnez-moi" size="20" onFocus="alert('Evénement
onFocus')" onBlur="alert('Evénement onBlur')" onChange="alert('Evénement onChange')"
onSelect="alert('Evénement onSelect')">
<input type="submit" value="Submit" name="B1">
<input type="reset" value="Reset" name="B2" onClick="alert('Evénement onCLick')">
</p>
</form>
```

Il est conseillé de mettre les traitements sur les événements dans fonctions entre les tags <script> et </script> et de faire appel à ces fonctions lors de l'événement. Ceci afin de faciliter la mise à jour et la lisibilité des sources

14.7 Fade (dégradé du fond d'écran)

Le fading, c'est faire apparaître progressivement le fond d'écran d'une page à partir d'une couleur pour finir à la couleur définitive de la page. Cela fonctionne si on ne met **pas** d'image de fond sur la page.

Le code

```
<html>
<head>
<title>Exemple de fading</title>
</head>
<body>
<script LANGUAGE="JavaScript">
<!--
//construction d'un tableau qui contiendra les caractères hexadécimaux
var hexa = new Array(16);
for(var i = 0; i < 10; i++) hexa[i]=i;
hexa[10] = "A";
hexa[11] = "B";
hexa[12] = "C";
hexa[13] = "D";
hexa[14] = "E";
hexa[15] = "F";
function toHex(i) {
//convertit une valeur décimale en valeur hexadécimale
if (i < 0) return "00";
if (i > 255) return "FF";
return "" + hexa[Math.floor(i/16)] + hexa[Math.floor(i%16)];
}
function setbgColor(R,G,B) {
//fonction qui met à jour la couleur de fond de la page
var r = toHex(R);
var g = toHex(G);
var b = toHex(B);

document.bgColor = "#"+r+g+b;
}
function fade(dR,dG,dB,fR,fG,fB,pas) {
//paramètres :
// dR est la couleur rouge de début
// dG est la couleur verte de début
// dB est la couleur bleue de début
// fR est la couleur rouge de fin
// fG est la couleur verte de fin
// fB est la couleur bleue de fin
// pas est la vitesse, il varie de 2 (vite) à 255 (lent)

for( var i = 0; i <= pas; i++) {
```

```
setbgColor(
Math.floor(dR * ((pas-i)/pas) + fR * (i/pas)),
Math.floor(dG * ((pas-i)/pas) + fG * (i/pas)),
Math.floor(dB * ((pas-i)/pas) + fB * (i/pas)));
}
}

//appel de la fonction fade
fade(0,0,0,255,255,255,40);
//-->
</script>
</body>
</html>
```

Ne vous inquiétez pas si ça clignote un peu lors du test, c'est normal : c'est parce que le navigateur doit réafficher la page à chaque passage dans la boucle. Lorsque le fade a lieu au chargement de la page, le navigateur ne doit réafficher qu'une page blanche, ce qui est plus rapide.

14.8 Faire pleuvoir

```
<html>
<head>
<title></title>
</head>
<body>
<script language="JavaScript">
<!--
var no = 50;
var speed = 1;
var ns4up = (document.layers) ? 1 : 0;
var ie4up = (document.all) ? 1 : 0;
var s, x, y, sn, cs;
var a, r, cx, cy;
var i, doc_width = 800, doc_height = 600;

if (ns4up) {
doc_width = self.innerWidth;
doc_height = self.innerHeight;
} else
if (ie4up) {
doc_width = document.body.clientWidth;
doc_height = document.body.clientHeight;
}

x = new Array();
y = new Array();
r = new Array();
cx = new Array();
cy = new Array();
s = 8;
```

```
for (i = 0; i < no; ++ i) {
initRain();
if (ns4up) {
if (i == 0) {
document.write("<layer name=\"dot"+ i +"\" left=\"1\" ");
document.write("top=\"1\" visibility=\"show\"><font color=\"blue\">");
document.write(",</font></layer>");
} else {
document.write("<layer name=\"dot"+ i +"\" left=\"1\" ");
document.write("top=\"1\" visibility=\"show\"><font color=\"blue\">");
document.write(",</font></layer>");
}
}
else
if (ie4up) {
if (i == 0) {
document.write("<div id=\"dot"+ i +"\" style=\"POSITION: ");
document.write("absolute; Z-INDEX: "+ i +"; VISIBILITY: ");
document.write("visible; TOP: 15px; LEFT: 15px;\"><font color=\"blue\">");
document.write(",</font></div>");
}
else {
document.write("<div id=\"dot"+ i +"\" style=\"POSITION: ");
document.write("absolute; Z-INDEX: "+ i +"; VISIBILITY: ");
document.write("visible; TOP: 15px; LEFT: 15px;\"><font color=\"blue\">");
document.write(",</font></div>");
}
}
}

function initRain() {
a = 6;
r[i] = 1;
sn = Math.sin(a);
cs = Math.cos(a);
cx[i] = Math.random() * doc_width + 1;
cy[i] = Math.random() * doc_height + 1;
x[i] = r[i] * sn + cx[i];
y[i] = cy[i];
}

function makeRain() {
r[i] = 1;
cx[i] = Math.random() * doc_width + 1;
cy[i] = 1;
x[i] = r[i] * sn + cx[i];
y[i] = r[i] * cs + cy[i];
}

function updateRain() {
r[i] += s;
x[i] = r[i] * sn + cx[i];
y[i] = r[i] * cs + cy[i];
}
```

```
function raindropNS() {
for (i = 0; i < no; ++ i) {
updateRain();
if ((x[i] <= 1) || (x[i] >= (doc_width - 20)) || (y[i] >= (doc_height - 20))) {
makeRain();
doc_width = self.innerWidth;
doc_height = self.innerHeight;
}
document.layers["dot"+i].top = y[i];
document.layers["dot"+i].left = x[i];
}
setTimeout("raindropNS()", speed);
}

function raindropIE() {
for (i = 0; i < no; ++ i) {
updateRain();
if ((x[i] <= 1) || (x[i] >= (doc_width - 20)) || (y[i] >= (doc_height - 20))) {
makeRain();
doc_width = document.body.clientWidth;
doc_height = document.body.clientHeight;
}
document.all["dot"+i].style.pixelTop = y[i];
document.all["dot"+i].style.pixelLeft = x[i];
}
setTimeout("raindropIE()", speed);
}

if (ns4up) {
raindropNS();
}
else
if (ie4up) {
raindropIE();
}
//-->
</script>

</body>
</html>
```

14.9 Fenêtres

Les fenêtres peuvent servir à afficher un texte à l'utilisateur mais en ne voulant pas qu'il quitte la page qu'il est en train de voir.

Vous pouvez ouvrir une fenêtre lorsque vouz appuyez sur un bouton ou sur un lien. Si vous désirez d'utiliser un lien et de rester sur la page courante, vous devez créer un lien avec le tag **HREF="javascript://"** ou **HREF="#"**. Cela sert à dire au browser que quand l'utilisateur clique sur le lien, le broswer doit rester sur la même page.

Comment cela fonctionne?

L'événement **onClick** déclenché lorsqu'on appuye sur le lien.
Ouverture de la fenêtre
On ouvre une fenêtre grâce à la commande **window.open()**.
Voici le code pour ouvrir une fenêtre de base :

```
mafenetre = window.open("fichier.html","Nom_de_la_fenetre");
```

Les paramètres de window.open()

- width = valeur □ hauteur de la fenêtre en pixels
- height = valeur □ largeur de la fenêtre en pixels
- toolbar = yes | no □ affiche ou non la bar d'outils
- location = yes | no □ affiche ou non la de déroulement
- directories = yes | no □ affiche ou non la barre des répertoires
- status = yes | no □ affiche ou non la barre de status
- menubar = yes | no □ affiche ou non le menu
- scrollbars = yes | no □ affiche ou non les barres de défilement
- resizeable = yes | no □ permet à la fenêtre d'être redimensionnée

Note : vous pouvez utiliser 1 | 0 ou yes | no dans les paramètres.

Fermer une fenêtre

On peut fermer une fenêtre de 2 façons :

1. mafenetre.close() si vous avez ouvert plusieurs fenêtres ou si vous voulez la fermer à partir d'une autre fenêtre.

0	
Ouvrir	Fermer

2. window.close() pour fermer la fenêtre courante.

Exemples

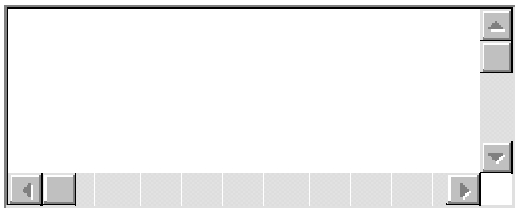
```
mafenetre = window.open("fichier.html","Nom_de_la_fenetre","toolbar=no,location=yes");
```

mafenetre

Créez votre propre fenêtre :

Erreur! Signet non défini.Paramètres de la fenêtre				
toolbar =	yes	V1	no	V2
location =	yes	V1	no	V2
directories =	yes	V3	no	V2
status =	yes	V1	no	V2
menubar =	yes	V1	no	V2
scrollbars =	yes	V1	no	V2
resizable =	yes	V1	no	V2
width =	600			
height =	300			
	Tester			

Voici le code de la fenêtre :



Créer une fenêtre et écrire dedans

Le code

```
<html>
<head>
<script LANGUAGE="JavaScript">
<!--
function fenDynamique() {
var win = window.open('','exemple','resizable=yes');
with (win.document) {
writeln('<html>');
writeln('<head>');
writeln('<title>Exemple de fenêtre</title>');
writeln('</head>');
writeln('<body background=\"images/bg_gr.gif\">');
writeln('<center><h2>Exemple de fenêtre</h2></center><hr>');
writeln('<p>Cette fenêtre est entièrement créée à l\'exécution<br>');
writeln('Elle peut servir à afficher les résultats d\'une forme par exemple</p>');
writeln('</body>');
writeln('</html>');
}
}
//-->
</script>
</head>
<body>
<form method=post>
<p><input type="button" value="Exemple" name="B1" onClick="fenDynamique()"</p>
</form>
</body>
</html>
```

On ouvre une fenêtre en l'associant à une variable (ici, *win*). Après, il suffit d'écrire dedans grâce à l'instruction **win.document.writeln('votre texte')**.

L'instruction **with (win.document)** permet de ne pas toujours devoir taper *win.document* devant chaque writeln.

14.10 Une fenêtre qui grandit

```
<html>
<head>
<script LANGUAGE="JavaScript">
<!--
var maPage = "http://www.domaine.com/mapage.html"
var winHauteur = 100
var winLargeur = 100
var pasX = 10
var pasY = 10
var speed = 1

function go(){
win2 = window.open("","","scrollbars")
if (!document.layers && !document.all){
win2.location = maPage
return true
}
win2.resizeTo(winLargeur,winHauteur)
win2.moveTo(0,0)
go2()
}

function go2(){
if (winHauteur >= screen.availHeight-3) {
pasY = 0
//si la fenêtre dépasse, on la remet à la taille de l'écran
win2.resizeTo(winLargeur,screen.availHeight-3)
}
if (winLargeur >= screen.width-4) {
pasX = 0
//si la fenêtre dépasse, on la remet à la taille de l'écran
win2.resizeTo(screen.width-4,winHauteur)
}
win2.resizeBy(pasX,pasY)
winHauteur += pasY
winLargeur += pasX
// fin de l'agrandissement
if ((winLargeur >= screen.width-4) && (winHauteur >= screen.availHeight-3)) {
win2.location = maPage
// Remise des valeurs de début
winHauteur = 100
winLargeur = 100
pasX = 50
pasY = 5
return true
}
setTimeout("go2()",speed)
}
//-->
```

```
</script>
</head>
<body>
<a href="javascript:go()">Cliquez ici !</a>
</body>
</html>
```

14.11 Les horloges

Les horloges les plus courantes se retrouvent dans un objet de type "one-line text box". Ce sont les plus faciles car elles ne nécessitent pas d'images.

17:02:54

Comment cela fonctionne?

Pour afficher l'heure, il faut utiliser la fonction **setTimeout(fonction(), duree);**
fonction() est la fonction à utiliser pour afficher l'heure
duree est la durée en millisecondes du chronomètre. Ici, on fait appel toutes les 1000 millisecondes à la fonction "fonction()"

Le code

```
<html>
<head>
<script LANGUAGE="JavaScript">
<!--
//variable qui permet d'identifier le timer afin de pouvoir l'arrêter
var timerID = null;
var timerActif = false;

function stopClock() {
if (timerActif) clearTimeout(timerID);
timerActif = false;
}

function startClock() {
stopClock();
showtime();
}

function showtime() {
var now = new Date();
var hour = now.getHours();
var min = now.getMinutes();
var sec = now.getSeconds();
heure = (hour > 9? hour:"0" + hour);
heure += ":" + (min > 9? min:"0" + min);
heure += ":" + (sec > 9? sec:"0" + sec);
document.Clock.Horloge.value = heure;
//pour mettre l'horloge dans la barre de status :
//window.status = heure;
timerID = setTimeout("showtime()",1000);
```

```
timerActif = true;
}
//-->
</script>

</head>
<body onLoad="startClock()" onUnLoad="stopClock()">
<form name="Clock">
<center>
<p><input type="text" name="Horloge" size="8" style="font-family: Courier New"></p>
</center>
</form>
</body>
</html>
```

- **onLoad="startClock();"** signifie que lorsqu'on ouvre la fenêtre, on démarre l'horloge.
- **onUnLoad="startClock();"** signifie que lorsqu'on ferme la fenêtre, on arrête l'horloge.
-

Il est aussi possible d'utiliser des images pour afficher l'heure.

Le code

```
<html>
<head>
<script LANGUAGE="JavaScript">
<!--
//variable qui permet d'identifier le timer afin de pouvoir l'arrêter
var timerID = null;
var timerActif = false;

if(document.images){
chiffre = new Array(10);
chiffre[0] = new Image(); chiffre[0].src = "images/r0.gif";
chiffre[1] = new Image(); chiffre[1].src = "images/r1.gif";
chiffre[2] = new Image(); chiffre[2].src = "images/r2.gif";
chiffre[3] = new Image(); chiffre[3].src = "images/r3.gif";
chiffre[4] = new Image(); chiffre[4].src = "images/r4.gif";
chiffre[5] = new Image(); chiffre[5].src = "images/r5.gif";
chiffre[6] = new Image(); chiffre[6].src = "images/r6.gif";
chiffre[7] = new Image(); chiffre[7].src = "images/r7.gif";
chiffre[8] = new Image(); chiffre[8].src = "images/r8.gif";
chiffre[9] = new Image(); chiffre[9].src = "images/r9.gif";
Blanc = new Image(); Blanc.src = "images/rblanc.gif";
}
function stopClock() {
if (timerActif) clearTimeout(timerID);
timerActif = false;
}

function startClock() {
stopClock();
showtime();
}
```

```
function showtime() {
var now = new Date();
var hour = now.getHours();
var min = now.getMinutes();
var sec = now.getSeconds();

affiche(hour,0);
affiche(min,3);
affiche(sec,6);
timerID = setTimeout("showtime()",1000);
timerActif = true;
}

function affiche(nombre, rang) {
var unites = nombre % 10
var dizaines = Math.floor(nombre / 10)
document.images[rang+1].src = chiffre[unites].src;
if (dizaines == 0 && rang == 0)
document.images[rang].src = Blanc.src;
else
document.images[rang].src = chiffre[dizaines].src ;
}
//-->
</script>
</head>
<body onLoad="startClock()" onUnLoad="stopClock()">
<p align="center">
<img src="images/rblanc.gif" width="15" height="20">
<img src="images/rblanc.gif" width="15" height="20">
<img src="images/rled.gif" width="15" height="20">
<img src="images/rblanc.gif" width="15" height="20">
<img src="images/rblanc.gif" width="15" height="20">
<img src="images/rled.gif" width="15" height="20">
<img src="images/rblanc.gif" width="15" height="20">
<img src="images/rblanc.gif" width="15" height="20">
</p>

</body>
</html>
```

Lers fichiers contenant les différentes images que vous pouvez utiliser sont joints

14.12 Des infos bulles sur les liens

La couleur du fond est tout ce que vous pouvez changer pour modifier l'apparence de la bulle.

```
<html>
<head>
```

```
<script LANGUAGE="JavaScript">
<!--
function showtip(current,e,text){
if (document.all){
document.all.tooltip.innerHTML='<marquee style="border:1px solid blue">'+text+'</marquee>'
document.all.tooltip.style.pixelLeft=event.clientX + document.body.scrollLeft + -75
document.all.tooltip.style.pixelTop=event.clientY + document.body.scrollTop + 20
document.all.tooltip.style.visibility="visible"
}
else if (document.layers){
document.tooltip.document.nstip.document.write('<b>'+text+'</b>')
document.tooltip.document.nstip.document.close()
document.tooltip.document.nstip.left=0
currentscroll=setInterval("scrolltip()",100)
document.tooltip.left=e.pageX + -75
document.tooltip.top=e.pageY + 20
document.tooltip.visibility="show"
}
}

function hidetip(){
if (document.all)
document.all.tooltip.style.visibility="hidden"
else if (document.layers){
clearInterval(currentscroll)
document.tooltip.visibility="hidden"
}
}

function scrolltip(){
if (document.tooltip.document.nstip.left >=
-document.tooltip.document.nstip.document.width)
document.tooltip.document.nstip.left-=3 //vitesse pour NS
else
document.tooltip.document.nstip.left=150
}
//-->
</script>
</head>

<body>
<div id="tooltip" style="font-family:verdana ; font-size:10px ; clip:rect(0px 150px 40px 0px) ;
background-color:yellow ; color:black ; position:absolute ; visibility:hidden ; width:150px">
<layer bgColor="white" width="1024" name="nstip"></layer>
</div>
<p align="center">
<a href="page.html" onmouseout="hidetip()" onmouseover="showtip(this,event,'Votre texte ici')"
target="_self">Texte</a>
</p>

</body>
</html>
```

Remarque: les couleurs ne fonctionnent qu'avec MSIE.

14.13 Une image qui reste en bas de la page

```
<html>
<head>

<DIV CLASS="jsbrand" ID="jsbrand" STYLE="position:absolute;top:1;visibility:hide;"
zIndex="1000" ALIGN="right">
<A HREF="#">
<IMG SRC="images/diffmade.gif" alt="Texte" BORDER="1"></A>
</DIV>
<script language="javascript" src="wtrmrk.js"></script>
</body>
</html>
```

Le script est disponible

Il reste à changer l'image, le lien vers lequel elle pointe et quelques variables dans le script.

14.14 Liens dans une liste déroulante

A la place d'une série de liens, on peut mettre ces liens dans une liste déroulante. Cela a comme avantage de réduire la place utilisée pour mettre votre liste.
Comment cela fonctionne?

On peut utiliser 2 façons de faire :

1. choisir un lien et cliquer sur un bouton. On utilise alors l'événement ***onClick***.
2. aller directement sur le lien choisit dès qu'on en choisit un grâce à ***onChange***.

Dans les 2 cas, le code est le même.

Le code

```
<html>
<head>
<title>Exemple de liens dans une liste déroulante</title>
<script LANGUAGE="JavaScript">
<!--
function change(){
if (window.document.formListe.liste.selectedIndex != 0)
//la première valeur est, ici, une valeur bidon qu'on utilise
//pour donner un titre à la liste déroulante
window.location = window.document.formListe.liste.options
[document.formListe.liste.selectedIndex].value
}
// -->
</script>
</head>
<body>
<form NAME="formListe">
<select NAME="liste" onChange="change()" size="1">
<option value="aucun"> Choisissez un lien </option>
<option value="home.html">Home</option>
```

```
<option value="contact.html">Contact</option>
<option value="javascript.html">JavaScript</option>
<option value="http://www.altavista.com">Altavista</option>
<option value="http://www.yahoo.com">Yahoo</option>
</select>
</form>
</body>
</html>
```

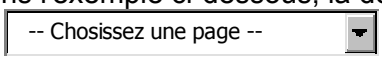
Voici un peu plus d'explications :

window.location : on attribue à la fenêtre une nouvelle URL. Ici, la destination va s'afficher dans la frame. Pour afficher la nouvelle page dans l'entièreté de la fenêtre, il suffit de mettre ceci : ***window.parent.location***

<option value="page_de_destination"> nom qui va apparaître dans la liste</option>

Dans ***value***, vous mettez la page ou le site de destination.

Dans l'exemple ci-dessous, la destination va s'afficher dans une boîte de dialogue.



14.15 L'objet screen

L'objet Screen contient des informations sur l'écran tel que les couleurs, la taille, ...
Cet objet est disponible depuis la version 1.2 du JavaScript.

Propriétés

Propiétés	Description
availHeight	Renvoie la hauteur de l'écran en pixels moins l'interface utilisateur affichée par l'os (comme la barre des tâches pour Windows). *Pour les utilisateurs de windows, agrandissez ou réduisez la barre des tâches ou déplacez la et rechargez la page, vous verrez la valeur changer.*
availWidth	Renvoie la largeur de l'écran en pixels moins l'interface utilisateur affichée par l'os (comme la barre des tâches pour Windows). *Pour les utilisateurs de windows, agrandissez ou réduisez la barre des tâches ou déplacez la et rechargez la page, vous verrez la valeur changer.*
colorDepth	Renvoie le nombre le nombre de bits pour la couleur. exemples : 256 couleurs : 8 65000 couleurs : 16 ...
height	Hauteur de l'écran
pixelDepth	Renvoie la résolution de l'écran (bits par pixel)
width	Largeur de l'écran

Configuration de votre écran

screen.availHeight	836
screen.availWidth	1152
screen.colorDepth	32 bits ou 4294967296 couleurs
screen.height	864
screen.pixelDepth	undefined
screen.width	1152

14.16 Manipulation d'images

Il y a sur le web beaucoup de pages avec des images, qui lorsque vous passez le curseur de la souris, changent d'aspect. Ce genre d'images servent à mettre dans des menus par exemple, afin de pouvoir se diriger vers d'autres pages.

Passez le curseur de la souris sur l'image désignée et vous la verrez se transformer en une autre, qui peut être porteuse d'un lien html.

Comment cela fonctionne?
2 événements liés à la souris permettent de contrôler l'apparence de l'image.

1. ***onMouseOver*** est l'événement qui est envoyé lorsque le curseur se trouve sur l'image.
2. ***onMouseOut*** intervient lorsque le curseur quitte l'image.

Le code

Mettons une image et donnons lui un nom :

```
<P><A HREF="page.html">
<IMG SRC="images/off.gif" NAME="Bouton1" BORDER="0" WIDTH="32" HEIGTH="48">
</A></P>
```

Le tag **NAME="Bouton1"** doît être mis afin de référencer l'image.
Les tags **BORDER** (taille du bord de l'image), **WIDTH**(largeur de l'image) et **HEIGHT**(hauteur de l'image) ne sont pas obligatoires.

Rajoutons maintenant les événements onMouseOver et onMouseOut

```
<P><A HREF="page.html" onMouseOver="Bouton1.src='images/on.gif'"
onMouseOut="Bouton1.src='images/off.gif'">
<IMG SRC="images/off.gif" NAME="Bouton1" BORDER="0" WIDTH="32" HEIGTH="48">
</A></P>
```

Notes : les 2 événements doivent être ajoutés dans le tag <a href>
Vous pouvez aussi utiliser des fonctions, cela permet de faciliter la mise à jour du code en cas de changement.

On fait appel aux fonctions dans les événements.

```
<P><A HREF="page.html" onMouseOver="fnctMouseOver('Bouton1')"
onMouseOut="fnctMouseOut('Bouton1')">
<IMG SRC="images/off.gif" NAME="Bouton1" BORDER="0" WIDTH="32" HEIGTH="48">
```

```
</A></P>
```

Voici comment définir les fonctions.

```
<head>
<script LANGUAGE="JavaScript">
<!--
//déclaration des images
if(document.images){
boutonOn = new Image(); boutonOn.src = "images/on.gif";
boutonOff = new Image(); boutonOff.src = "images/off.gif";
}

function fnctMouseOver(nom){
if (nom=="Bouton1")
document.Bouton1.src = boutonOn.src;
}

function fnctMouseOut(nom){
if (nom=="Bouton1")
document.Bouton1.src = boutonOff.src;
}

-->
</script>
</head>
```

Voici le code complet pour une liste de liens.

```
<html>
<head>
<title>JavaScript : Images</title>
<script LANGUAGE="JavaScript">
<!--
//je déclare 2 images
//listeon lorsque le curseur de la souris est sur le lien
//listeoff lorsque le curseur quitte le lien
listeon = new Image();
listeon.src = "images/red.gif";
listeoff = new Image();
listeoff.src = "images/ltgrey.gif";
function listeIn(imgName){
//imgName est le nom de l'image que vous avez donné
document[imgName].src = listeon.src;
}
function listeOut(imgName){
//imgName est le nom de l'image que vous avez donné
document[imgName].src = listeoff.src;
}
// -->
</script>

</head>
```

```
<body>
<img src="images/ltgrey.gif" name="L1" width="20" height="19">
<a href="contact.html" onMouseOver="listeIn('L1')"
onMouseOut="listeOut('L1')">contact</a><br>
<img src="images/ltgrey.gif" name="L2" width="20" height="19">
<a href="download.html" onMouseOver="listeIn('L2')"
onMouseOut="listeOut('L2')">download</a><br>
<img src="images/ltgrey.gif" name="L3" width="20" height="19">
<a href="javascript.html" onMouseOver="listeIn('L3')"
onMouseOut="listeOut('L3')">javascript</a><br>
<img src="images/ltgrey.gif" name="L4" width="20" height="19">
<a href="liens.html" onMouseOver="listeIn('L4')" onMouseOut="listeOut('L4')">liens</a><br>
</body>
</html>
```

Je n'utilise pas "document.nom_image.src" mais plutôt "document[nom_image].src". On peut utiliser les 2 façons de faire.

Note : Dreamweaver permet de programmer ce genre d'effet en quelques secondes via la fonction « insertion image rollover » :
Une image de base, affichée en permanence,
Une image alternative, affichée seulement quand on passe la souris sur l'image de base,
Eventuellement, un lien sur l'image alternative.

14.17 Un message d'accueil pour votre site

Ce qui est en jaune est à adapter selon votre désir.

```
<head>
<script language="JavaScript">
<!--
message = "Bienvenue sur mon site";
//Un seul texte qui est divisé en plusieurs parties en fonction des espaces.
//Chaque partie étant affichée l'une après l'autre.
colours = new Array('000099','ff6600')
siZe = 20;
message = message.split(' ');
timer = null;
clrPos = 0;
msgPos = 0;
jog = 1;
currentStep = 10;
step = 8;
ns = (document.layers)?1:0;
viz = (document.layers)?'hide':'hidden';
if (ns)
document.write("<div id='T' style='position:absolute'></div><br>");
else {
document.write("<div style='position:absolute'>");
document.write("<div align='center' style='position:relative'>");
document.write("<div id='T' style='position:absolute;width:0;height:0;font-family:Arial;font-
size:0'>kurt</div>");
document.write("</div></div><br>");
```

```
}
function Message() {
var pageHeight = (document.layers)?window.innerHeight:window.document.body.offsetHeight;
var pageWidth = (document.layers)?window.innerWidth:window.document.body.offsetWidth;
if (ns) {
ypos = pageHeight / 2;
var Write = '<div align="center" style="width:0px;height:0px;font-family:Arial,Verdana;font-
size:'+currentStep/4+'px;color:'+colours[clrPos]+'">'+message[msgPos]+'</div>';
document.T.top = ypos + -currentStep / 8 + window.pageYOffset;
document.T.document.write(Write)
document.T.document.close();
}
else {
ypos = pageHeight / 2;
xpos = pageWidth / 2;
T.style.width = currentStep;
T.style.pixelTop = ypos + -currentStep / 16 + document.body.scrollTop;
T.style.pixelLeft = (xpos - 20)+ -currentStep / 2;
T.style.fontSize = currentStep / 8;
T.innerHTML = message[msgPos];
T.style.color = colours[clrPos];
}
if (ns)step += 5;
else step += 15;
currentStep += step
if (ns) {
if (currentStep > pageWidth) {
currentStep = 10;
step = 8;
msgPos += jog;
clrPos += jog;
}
if (clrPos >= colours.length) clrPos = 0;
}
else {
if (currentStep > pageWidth * siZe) {
currentStep = 10;
step = 8;
msgPos += jog;
clrPos += jog;
}
if (clrPos >= colours.length) clrPos = 0;
}
if (msgPos >= message.length) {
clearTimeout(timer);
if (ns) document.T.visibility = viz;
else T.style.visibility = viz;
}
timer = setTimeout("Message()",40)
}
//-->
</script>

</head>
```

```
<body onLoad="Message()">
```

14.18 Ouvrir 2 fenêtres avec un lien

Insérez cette ligne où vous voulez :

```
<a href="page1.html" onClick="javascript:window.open('page2.html')">lien</a>
```

14.19 Rafraîchir automatiquement une page

Cette fonction est identique à la fonction recharger d'un navigateur. C'est idéal pour une page qui accueille une webcam ou qui doit actualiser une image toutes les X secondes.
C'est la même façon de faire que pour le tag HTML "***<META HTTP-EQUIV="REFRESH" CONTENT="10">***"

L'exemple donné ci-dessous rechargera la page toutes les 10 secondes.

```
<html>
<head>
<title>Rafraîchir automatiquement une page</title>
</head>
<body onLoad="window.setTimeout('history.go(0)', 10000)">
</body>
</html>
```

history.go(0) signifie qu'on recharge la page en cours. Mais avec cette méthode, je crois que la page est reprise du cache du navigateur.

Pour recharger la page, on peut aussi utiliser **self.location = "URL"** ou encore **self.location=self.location** ou enfin **location.reload();**

De cette manière, que l'on soit sur la fenêtre principale ou dans une frame, la fenêtre à partir de laquelle ce code est appelé (la fenêtre active) sera rechargée.
Pour recharger une autre frame que la frame courante,

- **window.parent.nom_de_la_frame.location = "URL"**
- **window.parent.frames["nom_de_la_frame"].location = "URL"**

Pour recharger la fenêtre entière lorsqu'il y a des frames :
window.parent.location=window.parent.location ou **window.parent.location.reload();**

14.20 Jouer un son en JavaScript

Il ne faut pas oublier qu'il existe plusieurs navigateurs avec leurs propres tags HTML.
Il faut prévoir la correspondance pour les autres navigateurs sinon certains entendront un son, d'autres pas.

Netscape utilise le tag "**<embed>**"
Internet Explorer, quand à lui, utilise "**<bgsound>**"

La première chose à faire est donc de détecter quel navigateur le visiteur utilise, ceci grâce à la fonction **navigator.appName**.

Une fois celui-ci connu, on sait quel tag il faut utiliser.
Voici un script qui joue un son au chargement de la page en fonction du navigateur :

```
<html>
<head>
<title> </title>
<script LANGUAGE="JavaScript">
<!--
var MSIE = navigator.appName=="Microsoft Internet Explorer";
if (!MSIE) {
document.write('<embed src="FichierSon" autostart=true hidden=true>');
} else {
document.write('<bgsound src="FichierSon" loop="0">');
}
//-->
</script>
</head>

<body>

</body>
</html>
```

Il est aussi possible de démarrer ou d'arrêter un son à volonté (nécessite un navigateur 4.0 et plus)

Choisissez une musique : 


```
Erreur! Signet non défini.<html>
<head>
<title> </title>
<script LANGUAGE="JavaScript">
<!--
var MSIE = navigator.appName=="Microsoft Internet Explorer";
function PlaySound(URL) {
if (parseInt(navigator.appVersion) <4) return false;
if (MSIE) {
document.all.sound1.src = (URL);
} else {
document.sound1.play(false, URL);
}
}

function StopSound() {
if (parseInt(navigator.appVersion) <4) return false;
if (MSIE) {
document.all.sound1.src = ("");
} else {
document.sound1.stop();
}
}
```

```
if (!MSIE) {
document.write('<embed src="letitbe.mid" autostart=false hidden=true name="sound1"
mastersound>');
} else {
document.write('<bgsound id="sound1" loop="0">');
}
//-->
</script>
</head>

<body>
<form name="midiForm">
Choisissez une musique : <select name="list">
<option value="letitbe.mid">Let it be
<option value="goldorak.mid">Goldorak
<option value="starwars.mid">Starwars
<option value="panther.mid">La panthère rose
</select>
<input type=button value="Jouer"
onClick="PlaySound(midiForm.list.options[midiForm.list.selectedIndex].value)">
<input type=button value="Stop" onClick="StopSound()">
</form>
</body>
</html>
```

14.21 Textes défilants (scrolling)

Comment cela fonctionne?

Il suffit juste, après un traitement du message en fonction du type de scrolling désiré, de l'afficher. On peut faire défiler du texte dans 3 endroits différents :

1. Dans la barre de status.
   Grâce à l'instruction **window.status="Texte"**
2. Dans un formulaire.
   Grâce à l'instruction **window.document.nom_formulaire.nom_champs.value="Texte"**
3. Dans le titre du navigateur (Pour Internet Explorer).
   Grâce à l'instruction **window.document.title="Texte"**

Il faut lancer le scrolling au chargement de la page et ne pas oublier d'enlever le timer losrqu'on quitte la page.

```
<head>
<script LANGUAGE="JavaScript">
<!--
//identificateur du timer
var timerID = null;
var delaiScroll = 100;
```

```
function startScroll() {
//le code pour le scrolling
timerID = setTimeout("startScroll()", delaiScroll);
}

function stopScroll() {
clearTimeout(timerID);
window.status = '';
//ne pas oublier si vous écrivez dans la barre de status sinon le texte restera dans l'état où il
était...
}

//-->
</script>
</head>
<body onLoad="startScroll()" onUnLoad="stopScroll()">
</body>
```

Exemples

- La machine à écrire
- Texte défilant
- Texte qui rebondit

14.21.1 Scrolling machine à écrire

Le texte à

```
Erreur! Signet non défini.<html>
<head>
<title>Scrolling machine à écrire</title>
<script LANGUAGE="JavaScript">
<!--
var texte = 'Le texte à faire défiler est à mettre ici...'
var timerID = null;
var pos = 0;
//position de la lettre a afficher
var delaiScroll = 50;
//vitesse du scroll
var pause = 2000;
//effectue une pause quand le texte est entièrement affiché
//ici, 2 secondes

function startScroll() {
var msg;

if (pos >= texte.length) {
pos = 0;
}

if (pos == 0) {
msg = texte;
```

```
timerID = setTimeout("startScroll()", pause);
}
else {
msg = texte.substring(0, pos);
timerID = setTimeout("startScroll()", delaiScroll);
}
pos++;
window.document.formScroll.textScroll.value = msg;
//ou dans la barre de status
// window.status = msg;
}

function stopScroll() {
clearTimeout(timerID);
window.status = '';
}
//-->
</script>
</head>

<body onLoad="startScroll()" onUnLoad="stopScroll()">

<h1 align="center">Scrolling machine à écrire</h1>

<hr>

<form name="formScroll">
<p><input type="text" name="textScroll" size="50"></p>
</form>
</body>
</html>
```

14.21.2 Texte défilant

```
<html>
<head>
<title>Texte défilant</title>
<script LANGUAGE="JavaScript">
<!--
var texte = 'Le texte à faire défiler est à mettre ici...'
var i = 0;
//sert à afficher le texte tout à fait à droite
while (i ++ < 100)
texte = " " + texte;

var timerID = null;
var pos = 0;
var delaiScroll = 75;

function startScroll() {
window.status = texte.substring(pos++, texte.length);
if (pos == texte.length)
pos = 0;
```

```
timerID = setTimeout("startScroll()", delaiScroll);
}

function stopScroll() {
clearTimeout(timerID);
window.status = '';
}
//-->
</script>
</head>

<body onLoad="startScroll()" onUnLoad="stopScroll()">

<h1 align="center">Texte défilant</h1>

<hr>

</body>
</html>
```

14.21.3 Texte qui rebondit

```
Erreur! Signet non défini.<html>
<head>
<title>Texte qui rebondit</title>
<script LANGUAGE="JavaScript">
<!--
var texte = "Le texte à faire défiler est à mettre ici...";
var nbEspaces = 20;
var i = 0;
while (i++ < nbEspaces) texte = " " + texte;
var timerID = null;
var pos = nbEspaces;
var direction = -1; //+1 = droite, -1 = gauche
var delaiScroll = 75;

function startScroll() {
var msg;

if (pos == nbEspaces) direction = -1;
if (pos == 0) direction = 1;
pos += direction;
msg = texte.substring(pos,texte.length);
window.status = msg;
window.document.formScroll.textScroll.value = msg;
timerID = setTimeout("startScroll()", delaiScroll);
}

function stopScroll() {
clearTimeout(timerID);
window.status = '';
```

```
}

//-->
</script>
</head>

<body onLoad="startScroll()" onUnLoad="stopScroll()">

<h1 align="center">Texte qui rebondit</h1>

<hr>

<form name="formScroll">
<p><input type="text" name="textScroll" style="font-family: Courier New" size="64"></p>
</form>

</body>
</html>
```

14.22 Afficher un texte avec un dégradé de couleurs

Voici un exemple de texte multi-couleur...
Degrade(0,200,0,255,0,100,"Voici un exemple de texte multi-couleur...");

```
<html>
<head>
<title>Texte multi-couleur</title>
<script LANGUAGE="JavaScript">
<!--
//construction du tableau des valeurs hexadécimales
var hexa = new Array(16);
for(var i = 0; i < 10; i++) hexa[i]=i;
hexa[10] = "A";
hexa[11] = "B";
hexa[12] = "C";
hexa[13] = "D";
hexa[14] = "E";
hexa[15] = "F";
function toHexa(n) {
//fonction de conversion de décimal à hexadécimal
if (n < 0) return "00";
if (n > 255) return "FF";
return "" + hexa[Math.floor(n/16)] + hexa[Math.floor(n%16)];
}
function Degrade(dr,dg,db,fr,fg,fb,texte) {
//paramètres :
// dR est la couleur rouge de début
// dG est la couleur verte de début
// dB est la couleur bleue de début
// fR est la couleur rouge de fin
// fG est la couleur verte de fin
// fB est la couleur bleue de fin
```

```
// texte : texte à afficher en couleur

var steps = texte.length;
cr = dr;
cg = dg;
cb = db;
sr = (fr - dr) / steps;
sg = (fg - dg) / steps;
sb = (fb - db) / steps;
for (var x = 0; x <= steps; x++) {
document.write('<FONT COLOR="#' + toHexa(cr) + toHexa(cg) + toHexa(cb) + '">');
document.write(texte.charAt(x));
document.write('</FONT>');
cr += sr; cg += sg; cb += sb;
}
}
Degrade(255,100,0,100,0,255,"Voici un exemple de texte multi-couleur...");
//-->
</script>
</head>

<body>

</body>
</html>
```

Mettre un texte trop long ralentit la vitesse d'affichage de la page.

14.23 Texte qui zoome et dézoome

```
<head>
<script language=javaScript>
<!--
// Vous pouvez modifier
var maxfontsize=40
var textcolor="AAAAAA"
var textfont="Arial"
var message="Votre texte"
// fin des modifications
var thissize=0
var step=1

function stretch() {
if (thissize<0) {step=1; thissize=0}
if (thissize < maxfontsize) {
if(document.all) {
zoomer.innerHTML="<span style='font-family:"+textfont+";font-
size:"+thissize+"px;color:"+textcolor+"'>"+message+"</span>"
}

if(document.layers) {
document.zoomer.document.write("<span style='font-family:"+textfont+";font-
```

```
size:"+thissize+"px;color:"+textcolor+"'>"+message+"</span>")
document.close()
}
step++
thissize=thissize+step
var timer=setTimeout("stretch()",50)
}
else {
clearTimeout(timer)
var intermezzo=setTimeout("shrink()",1000)
}
}

function shrink() {
if (thissize > -0) {
if(document.all) {
zoomer.innerHTML="<span style='font-family:"+textfont+";font-
size:"+thissize+"px;color:"+textcolor+"'>"+message+"</span>"
}

if(document.layers) {
document.zoomer.document.write("<span style='font-family:"+textfont+";font-
size:"+thissize+"px;color:"+textcolor+"'>"+message+"</span>")
document.close()
}
if (step >= 2) {step=step-1} else{step=1}

thissize=thissize-step
var timer=setTimeout("shrink()",50)
}
else {
clearTimeout(timer)
var intermezzo=setTimeout("stretch()",1000)
}
}
// -->
</script>
</head>

<body onLoad="stretch()">

<hr>
<div id="zoomer" style="position:absolute;visibility:visible;top:10px; left:10px"></div>
</body>
</html>
```

14.24 Bande de couleur derriere le texte

Insérer dans la partie <head> du document html

```
<STYLE TYPE="text/css">
.couleur {background-color: #FF8888;}
</STYLE>
```

Insérer dans la partie <body> du document html

```
<P CLASS="couleur">Votre texte</P>
```

14.25 Jeu du pendu

Code à insérer entre les balises Body

```
<P align=left><!--webbot bot="HTMLMarkup" startspan -->
                        <SCRIPT language=javascript>

                        /*
                        Submitted by Mike McGrath http://website.lineone.net/~mike_mcgrath
                        Featured on Website Abstraction (http://wsabstract.com)
                        For this and over 400+ free scripts, visit http://wsabstract.com
                        */

                        var alpha=new Array();
                        var alpha_index=0;

                        var bravo=new Array();
                        var bravo_index=0;

                        var running=0;
                        var failnum=0;
                        var advising=0;

                        function pick()
                        {
                        var choice="";
                        var blank=0;

                        for (i=0; i<words[index].length; i++)
                        {
                        t=0;
                        for(j=0; j<=alpha_index; j++)
                        if(words[index].charAt(i)==alpha[j] ||
words[index].charAt(i)==alpha[j].toLowerCase()) t=1;

                        if (t) choice+=words[index].charAt(i)+" ";
                        else
                        {
                        choice+="_ ";
                        blank=1;
                        }
                        }

                        document.f.word.value=choice;

                        if (!blank)
                        {
                        document.f.tried.value=" === BRAVO ! ===";
```

```
document.f.score.value++;
running=0;
}
}


function new_word(form)
{
if(!running)
{
running=1;
failnum=0;
form.lives.value=failnum;
form.tried.value="";
form.word.value="";
index=Math.round(Math.random()*10000) % 100;
alpha[0]=words[index].charAt(0);
alpha[1]=words[index].charAt(words[index].length-1);
alpha_index=1;
bravo[0]=words[index].charAt(0);
bravo[1]=words[index].charAt(words[index].length-1);
bravo_index=1;
pick();
}
else advise("Le mot est en place pour jouer !");
}

function seek(letter)
{
if (!running) advise(".....Cliquez sur GO !");
else
{
t=0;
for (i=0; i<=bravo_index; i++)
{
if (bravo[i]==letter || bravo[i]==letter.toLowerCase()) t=1;
}

if (!t)
{
document.f.tried.value+=letter+" "
bravo_index++;
bravo[bravo_index]=letter;

for(i=0;i<words[index].length;i++)
if(words[index].charAt(i)==letter || words[index].charAt(i)==letter.toLowerCase())
t=1;

if(t)
{
alpha_index++;
alpha[alpha_index]=letter;
}
else failnum++;
```

```
                    document.f.lives.value=failnum;
                    if (failnum==6)
                    {
                    document.f.tried.value="Perdu, moins un point";
                    document.f.word.value=words[index];
                    document.f.score.value--;
                    running=0;
                    }
                    else pick();
                    }
                    else advise("Cette lettre a déjà été utilisée");
                    }
                    }

                    function advise(msg)
                    {
                    if (!advising)
                    {
                    advising=-1;
                    savetext=document.f.tried.value;
                    document.f.tried.value=msg;
                    window.setTimeout("document.f.tried.value=savetext; advising=0;",1000);
                    }
                    }

                    var words = new Array("voiture","vache","reveil","radar","yoyo","stylo",
                    "repondeur","graveur","imprimante","scanner","disque","boite","lapin","amour",
                    "lumiere","spot","lampe","alien","hollywood","poster","affiche","fromage",
                    "oiseaux","mangeoir","horizon","ouest","magie","carte","triche","feuille",
                    "virgule","mario","allemagne","belgique","france","malabar","portable",
                    "violon","piano","guitare","fraiche","violet","jaune","rouge","orange",
                    "pomme","poire","helicoptere","livre","clavier","or","chaine","redaction",
                    "patin","skate","tamtam","disquette","ecran","enceinte","armoire",
                    "ordinateur","foire","menteur","tapis","produit","assassin","telephone",
                    "pekin","osier","tatouage","pain","journal","fleur","sucette","truite",
                    "esquimaux","ski","portique","pendule","fruit","tomate","carotte",
                    "allumette","papa","maman","menteur","tambour","javascript","cacahuete",
                    "camembert","vache");

                    </SCRIPT>

                    <FORM name=f>
                    <TABLE bgColor=#c0c0c0 border=1>
                    <TBODY>
                    <TR>
                    <TD align=right colSpan=4>Score : <INPUT name=score onfocus=score.blur();
                    size=2 value=0> <BR>Erreur (6): <INPUT name=lives onfocus=lives.blur();
                    size=2 value=0> </TD>
                    <TD align=middle colSpan=7><<INPUT name=word onfocus=word.blur();
size=25
                    value="Le jeu du pendu"> <BR><INPUT name=tried onfocus=tried.blur();
                    size=25 value="Go pour commencer."> </TD>
                    <TD align=middle colSpan=2><INPUT onclick=new_word(this.form);
type=button value=" GO ">
                    </TD></TR>
```

```
                    <TR>
                    <TD><INPUT onclick="seek('A');" type=button value=" A "></TD>
                    <TD><INPUT onclick="seek('B');" type=button value=" B "></TD>
                    <TD><INPUT onclick="seek('C');" type=button value=" C "></TD>
                    <TD><INPUT onclick="seek('D');" type=button value=" D "></TD>
                    <TD><INPUT onclick="seek('E');" type=button value=" E "></TD>
                    <TD><INPUT onclick="seek('F');" type=button value=" F "></TD>
                    <TD><INPUT onclick="seek('G');" type=button value=" G "></TD>
                    <TD><INPUT onclick="seek('H');" type=button value=" H "></TD>
                    <TD><INPUT onclick="seek('I');" type=button value=" I "></TD>
                    <TD><INPUT onclick="seek('J');" type=button value=" J "></TD>
                    <TD><INPUT onclick="seek('K');" type=button value=" K "></TD>
                    <TD><INPUT onclick="seek('L');" type=button value=" L "></TD>
                    <TD><INPUT onclick="seek('M');" type=button value=" M "></TD></TR>
                    <TR>
                    <TD><INPUT onclick="seek('N');" type=button value=" N "></TD>
                    <TD><INPUT onclick="seek('O');" type=button value=" O "></TD>
                    <TD><INPUT onclick="seek('P');" type=button value=" P "></TD>
                    <TD><INPUT onclick="seek('Q');" type=button value=" Q "></TD>
                    <TD><INPUT onclick="seek('R');" type=button value=" R "></TD>
                    <TD><INPUT onclick="seek('S');" type=button value=" S "></TD>
                    <TD><INPUT onclick="seek('T');" type=button value=" T "></TD>
                    <TD><INPUT onclick="seek('U');" type=button value=" U "></TD>
                    <TD><INPUT onclick="seek('V');" type=button value=" V "></TD>
                    <TD><INPUT onclick="seek('W');" type=button value=" W "></TD>
                    <TD><INPUT onclick="seek('X');" type=button value=" X "></TD>
                    <TD><INPUT onclick="seek('Y');" type=button value=" Y "></TD>
                    <TD><INPUT onclick="seek('Z');" type=button value=" Z
"></TD></TR></TBODY></TABLE></FORM><!--webbot
                    bot="HTMLMarkup"
                    endspan --></P>
```

14.26 Affiche une horloge simple

```
<html>
<head>
<title>Untitled Document</title>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">

                    <Script Language="javascript" Type = "text/javascript">
                    <!--
                    // Helpers for JSI page...
                    // Navigation - Start
                    function goback(){
                    alert("Good Bye!");
                    history.go(-1);
                    }
                    // Navigation - Stop
                    // Netscapes Clock - Start
                    // this code was taken from Netscapes JavaScript documentation at
                    // www.netscape.com on Jan.25.96

                    var timerID = null;
```

```
                var timerRunning = false;
                function stopclock (){
                if(timerRunning)
                clearTimeout(timerID);
                timerRunning = false;
                }

                function startclock () {
                // Make sure the clock is stopped
                stopclock();
                showtime();
                }

                function showtime () {
                var now = new Date();
                var hours = now.getHours();
                var minutes = now.getMinutes();
                var seconds = now.getSeconds()
                var timeValue = " " + ((hours >23) ? hours -23 :hours)
                timeValue += ((minutes < 10) ? "h0" : "h") + minutes
                document.clock.face.value = timeValue;
                // you could replace the above with this
                // and have a clock on the status bar:
                // window.status = timeValue;
                timerID = setTimeout("showtime()",1000);
                timerRunning = true;
                }
                // Netscapes Clock - Stop

                // end Helpers
                // -->

                </Script>







</head>

<body onload="startclock();">
<form name="clock">
                <input type="text" size="7" name="face" align="left">
                </form>
</body>
</html>
```

14.27 Recherche à l'intérieur d'une page

```
<html>
<head>
<title>Untitled Document</title>
```

```
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">

		<Script Language="javascript" Type = "text/javascript">
		<!-- Begin
		var NS4 = (document.layers);
		var IE4 = (document.all);

		var win = this;
		var n = 0;

		function findInPage(str) {
		var txt, i, found;
		if (str == "")
		return false;
		if (NS4) {
		if (!win.find(str))
		while(win.find(str, false, true))
		n++;
		else
		n++;
		if (n == 0) alert(str + " n\'a pas été trouvé dans cette page.");
		}
		if (IE4) {
		txt = win.document.body.createTextRange();
		for (i = 0; i <= n && (found = txt.findText(str)) != false; i++) {
		txt.moveStart("character", 1);
		txt.moveEnd("textedit");
		}
		if (found) {
		txt.moveStart("character", -1);
		txt.findText(str);
		txt.select();
		txt.scrollIntoView();
		n++;
		}
		else {
		if (n > 0) {
		n = 0;
		findInPage(str);
		}
		else
		alert(str + " n\'a pas été trouvé dans cette page.");
		}
		}
		return false;
		}
		// End -->

		</Script>


</head>

<body bgcolor="#FFFFFF">
<form name=search onSubmit="return findInPage(this.string.value);">
```

```
<p align="center">Rechercher : <input name=string type=text size=15 onChange="n = 0;">
</form>
</body>
</html>
```

14.28 Moteur de recherche

Entre <HEAD> et </HEAD>

```
<script language="JavaScript">
<!--
// Déclaration de la base de données pour la recherche
var page=new Array;     // page est un tableau qui contient l'adresse des pages
var m=new Array;      // mot est un tableau qui contient les mots clés associés aux pages

page[0]="page1.html";
m[0]="mot_clé_1,mot_clé_2,mot_clé_3,et_ainsi_de_suite";
page[1]="Page 2.html";
m[1]="mot_clé_1,mot_clé_2,mot_clé_3,et_ainsi_de_suite";


function go(txt) {
// txt contient le texte de la recherche

var n=m.length;
var indice=-1;
if (txt=="") {alert("Entrez un mot pour la recherche"); }
else
{ for (i=0; i<n; i++)
{if (m[i].toUpperCase().indexOf(txt.toUpperCase(),0)!="-1") {indice=i;}
}
if (indice>=0) {window.location=page[indice];}     // Recherche fructueuse
else {window.location="not_found.html";}         // Redirection vers la page NOT FOUND
}
}
//-->
</script>
```

Dans le tag <BODY>

```
Rien
```

Entre <BODY> et </BODY>

```
<form method=get>
Vous recherchez :<input name="search" size=35 maxlength=35><br>
<input type="button" value="Lancer la recherche..." onClick="go(document.forms[0].elements[0].value)">
</form>
```

Chapitre 15 Les formulaires

15.1 Généralités

Avec Javascript, les formulaires Html prennent une toute autre dimension. N'oublions pas qu'en Javascript, on peut accéder à chaque élément d'un formulaire pour, par exemple, y aller lire ou écrire une valeur, noter un choix auquel on pourra associer un gestionnaire d'événement...

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

15.2 Déclaration d'un formulaire

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

Une erreur classique en Javascript est, emporté par son élan, d'oublier de déclarer la fin du formulaire </FORM> après avoir incorporé un contrôle.

15.3 Le contrôle ligne de texte

La zone de texte est l'élément d'entrée/sortie par excellence de Javascript. La syntaxe Html est <INPUT TYPE="text" NAME="nom" SIZE=x MAXLENGTH=y> pour un champ de saisie d'une seule ligne, de longueur x et de longueur maximale de y.

L'objet text possède trois propriétés :

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

Lire une valeur dans une zone de texte

Voici un exemple :

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

Le script complet est celui-ci :

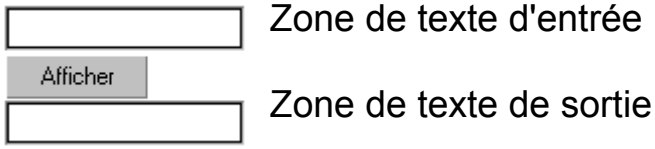
```
<HTML>
<HEAD>
<SCRIPT LANGUAGE="javascript">
function controle(form1) {
var test = document.form1.input.value;
alert("Vous avez tapé : " + test);
```

```
}
</SCRIPT>
</HEAD>
<BODY>
<FORM NAME="form1">
<INPUT TYPE="text" NAME="input" VALUE=""><BR>
<INPUT TYPE="button" NAME="bouton" VALUE="Contrôler" onClick="controle(form1)">
</FORM>
</BODY>
</HTML>
```

Lorsqu'on clique le bouton "contrôler", Javascript appelle la fonction controle() à laquelle on passe le formulaire dont le nom est form1 comme argument.
Cette fonction controle() définie dans les balises <HEAD> prend sous la variable test, la valeur de la zone de texte. Pour accéder à cette valeur, on note le chemin complet de celle-ci (voir le chapitre "Un peu de théorie objet"). Soit dans le document présent, il y a l'objet formulaire appelé form1 qui contient le contrôle de texte nommé input et qui a comme propriété l'élément de valeur value. Ce qui donne document.form1.input.value.

Ecrire une valeur dans une zone de texte

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



Voici le code :

```
<HTML>
<HEAD>
<SCRIPT LANGUAGE="javascript">
function afficher(form2) {
var testin =document. form2.input.value;
document.form2.output.value=testin
}
</SCRIPT>
</HEAD>
<BODY>
<FORM NAME="form2">
<INPUT TYPE="text" NAME="input" VALUE=""> Zone de texte d'entrée <BR>
<INPUT TYPE="button" NAME="bouton" VALUE="Afficher" onClick="afficher(form2)"><BR>
<INPUT TYPE="text" NAME="output" VALUE=""> Zone de texte de sortie
</FORM>
</BODY>
</HTML>
```

Lorsqu'on clique le bouton "Afficher", Javascript appelle la fonction afficher() à laquelle on passe le formulaire dont le nom est cette fois form2 comme argument.
Cette fonction afficher() définie dans les balises <HEAD> prend sous la variable testin, la valeur de la zone de texte d'entrée. A l'instruction suivante, on dit à Javascript que la valeur de la zone de texte output comprise dans le formulaire nommé form2 est celle de la variable testin. A

nouveau, on a utilisé le chemin complet pour arriver à la propriété valeur de l'objet souhaité soit en Javascript document.form2.output.value.

15.4 Les boutons radio

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

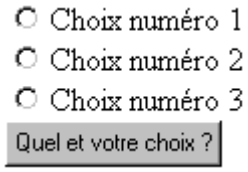
Propriété	Description
name	indique le nom du contrôle. Tous les boutons portent le même nom.
index	l'index ou le rang du bouton radio en commençant par 0.
checked	indique l'état en cours de l'élément radio
defaultchecked	indique l'état du bouton sélectionné par défaut.
value	indique la valeur de l'élément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

Prenons un exemple :

```
<HTML>
<HEAD>
<SCRIPT language="javascript">
function choixprop(form3) {
if (form3.choix[0].checked) { alert("Vous avez choisi la proposition " + form3.choix[0].value) };
if (form3.choix[1].checked) { alert("Vous avez choisi la proposition " + form3.choix[1].value) };
if (form3.choix[2].checked) { alert("Vous avez choisi la proposition " + form3.choix[2].value) };
}
</SCRIPT>
</HEAD>
<BODY>
Entrez votre choix :
<FORM NAME="form3">
<INPUT TYPE="radio" NAME="choix" VALUE="1">Choix numéro 1<BR>
<INPUT TYPE="radio" NAME="choix" VALUE="2">Choix numéro 2<BR>
<INPUT TYPE="radio" NAME="choix" VALUE="3">Choix numéro 3<BR>
<INPUT TYPE="button"NAME="but" VALUE="Quel et votre choix ?" onClick="choixprop(form3)">
</FORM>
</BODY>
</HTML>
```

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

Entrez votre choix :



Dans le formulaire nommé form3, on déclare trois boutons radio. Notez que l'on utilise le même nom pour les trois boutons. Vient ensuite un bouton qui déclenche la fonction choixprop().
Cette fonction teste quel bouton radio est coché. On accède aux boutons sous forme d'indice par rapport au nom des boutons radio soit choix[0], choix[1], choix[2]. On teste la propriété checked

du bouton par if(form3.choix[x].checked). Dans l'affirmative, une boite d'alerte s'affiche. Ce message reprend la valeur attachée à chaque bouton par le chemin form3.choix[x].value.

15.5 Les boutons case à cocher (checkbox)

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

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

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

☐ Choix numéro 1
☐ Choix numéro 2
☐ Choix numéro 3
☐ Choix numéro 4
Corriger

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

Dans le formulaire nommé form4, on déclare quatre cases à cocher. Notez que l'on utilise un nom différent pour les quatre boutons. Vient ensuite un bouton qui déclenche la fonction reponse().

Cette fonction teste quelles cases à cocher sont sélectionnées. Pour avoir la bonne réponse, il faut que les cases 1, 2 et 4 soient cochées. On accède aux cases en utilisant chaque fois leur nom. On teste la propriété checked du bouton par (form4.nom_de_la_case.checked). Dans l'affirmative (&& pour et logique), une boite d'alerte s'affiche pour la bonne réponse. Dans la négative, une autre boite d'alerte vous invite à recommencer.

15.6 Liste de sélection

Le contrôle liste de sélection vous permet de proposer diverses options sous la forme d'une liste déroulante dans laquelle l'utilisateur peut cliquer pour faire son choix. Ce choix reste alors affiché. La boite de la liste est crée par la balise <SELECT> et les éléments de la liste par un ou plusieurs tags <OPTION>. La balise </SELECT> termine la liste.

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

Un petit exemple comme d'habitude :

Entrez votre choix :


```
<HTML>
<HEAD>
<script language="javascript"> function liste(form5) {
alert("L\'élément " + (form5.list.selectedIndex + 1)); }
</SCRIPT>
</HEAD>
<BODY>
Entrez votre choix : <FORM NAME="form5">
<SELECT NAME="list">
<OPTION VALUE="1">Elément 1
<OPTION VALUE="2">Elément 2
<OPTION VALUE="3">Elément 3
</SELECT>
<INPUT TYPE="button"NAME="b" VALUE="Quel est l'élément retenu?" onClick="liste(form5)">
</FORM>
</BODY>
</HTML>
```

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

15.7 Le contrôle textarea

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

La syntaxe Html est :
<FORM>
<TEXTAREA NAME="nom" ROWS=x COLS=y>
texte par défaut
</TEXTAREA>
</FORM>

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

L'objet textarea possède plusieurs propriétés :

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

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

En Javascript, on utilisera \r\n pour passer à la ligne.
Comme par exemple dans l'expression document.Form.Text.value = 'Check\r\nthis\r\nout'.

15.8 Le contrôle Reset

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

la syntaxe Html est :

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

Une seule méthode est associée au contrôle Reset, c'est la méthode onClick(). Ce qui peut servir, par exemple, pour faire afficher une autre valeur que celle par défaut.

15.9 Le contrôle Submit

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

la syntaxe Html est :

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

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

15.10 Le contrôle Hidden (caché)

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

la syntaxe Html est :

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

Chapitre 16 Les frames

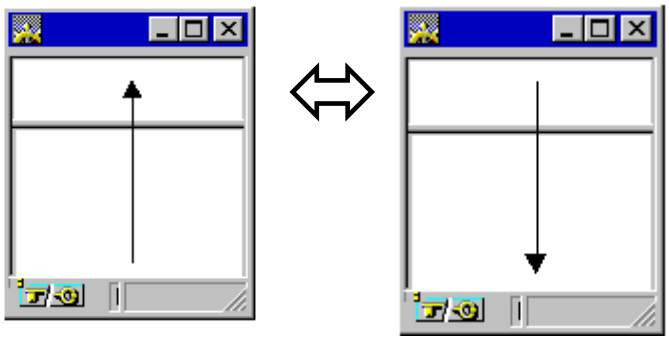
16.1 Généralités

Utiliser des frames vous permet de diviser la fenêtre affichant votre page HTML en plusieurs cadres (parties distinctes) indépendants les uns des autres. Vous pouvez alors charger des pages différentes dans chaque cadre.

En Javascript, nous nous intéresserons à la capacité de ces cadres à interagir. C'est à dire à la capacité d'échanger des informations entre les frames.

En effet, la philosophie du Html veut que chaque page composant un site soit une entité indépendante. Comment pourrait-on faire alors pour garder des informations tout au long du site ou tout simplement passer des informations d'une page à une autre page. Tout simplement (sic), en utilisant des frames.

Le schéma suivant éclairera le propos :



16.2 Propriétés

Propriétés	Description
length	Retourne le nombre de frames subordonnés dans un cadre "créateur".
parent	Synonyme pour le frame "créateur".

16.3 Echange d'informations entre frames

Par l'exemple suivant, nous allons transférer une donnée introduite par l'utilisateur dans une frame, dans une autre frame.

La page "créatrice" des frames

```
<HTML>
<HEAD>
</HEAD>
<FRAMESET ROWS="30%,70%">
<FRAME SRC="enfant1.htm" name="enfant1">
<FRAME SRC="enfant2.htm" name="enfant2">
</FRAMESET>
</HTML>
```

La page "créatrice" contient deux frames subordonnées "enfant1" et "enfant2".

Le fichier enfant1.htm

```
<HTML>
<BODY>
<FORM name="form1">
<INPUT TYPE="TEXT" NAME="en" value=" ">
</FORM>
</BODY>
</HTML>
```

Le fichier enfant2.htm

```
<HTML>
<HEAD>
<SCRIPT LANGUAGE="Javascript">
<!--
function affi(form) {
parent.enfant1.document.form1.en.value=document.form2.out.value
}
// -->
</SCRIPT>
</HEAD>
<BODY>
Entrez une valeur et cliquez sur "Envoyer".
<FORM NAME="form2" >
<INPUT TYPE="TEXT" NAME="out">
<INPUT TYPE="button" VALUE="Envoyer" onClick="affi(this.form)">
</FORM>
</BODY>
</HTML>
```

La donnée entrée par l'utilisateur se trouve par le chemin document.form2.out.value. Nous allons transférer cette donnée dans la zone de texte de l'autre frame. Pour cela, il faut en spécifier le chemin complet. D'abord la zone de texte se trouve dans la frame subordonnée appelée enfant1. Donc le chemin commence par parent.enfant1. Dans cette frame se trouve un document qui contient un formulaire (form1) qui contient une zone de texte (en) qui a comme propriété value. Ce qui fait comme chemin document.form1.en.value. L'expression complète est bien :
parent.enfant1.document.form1.en.value=document.form2.out.valu