L'Altruiste : Le guide des langages Web

Le Javascript

Sommaire

1 / Introduction

Les programmes Javascript permettent de rendre dynamique un site internet développé par le langage HTML. Comme nous avons pû le constater auparavant, le langage HTML crée des pages statiques dont les carences dans le domaine de l'interactivité peuvent désormais provoquer une désaffection du public pour votre site.

Créer un site interactif et dynamique peut constituer un atout indéniable pour **améliorer ou conforter votre audience** mais aussi afin de permettre à l'utilisateur **une navigation plus aisée** et de fournir **un aspect attractif** à votre site.

Le langage Javascript permet de développer de véritables applications fonctionnant exclusivement dans le cadre d'Internet. Néanmoins, plus le degré de complexité de ces applications est important, plus la puissance de calcul du processeur sera sollicité jusque dans ses confins. Ainsi, **un programme Javascript requiert une machine puissante et rapide chez l'usager**.

Contrairement à un applet Java qui est un programme compilé, les scripts écrits en Javascript sont interprétés, cela explique en partie la caractéristique précitée.
De la même manière de sorte à ne pas confondre les différents programmes : **le Java, représenté par un ou plusieurs fichiers autonomes** dont l'extension sera **.class* ou **.jar*, est invoqué par une balise HTML spécifique, alors que **le JavaScript est écrit directement au sein du document HTML** sous forme d'un script encadré par des balises HTML spéciales.

Cependant, **le Javascript est un langage de script simplifié orienté objet dont la syntaxe est basée sur celle du Java** qui lui-même prend sa source dans les langages de développement C et C++.

Le langage Javascript a été initialement élaboré par Netscape en association avec Sun Microsystem.

Plus tard, Microsoft développera son propre langage Javascript officiellement connu sous le nom de *JScript*.

Tout comme, le langage HTML ou les feuilles de style, **le Javascript est standardisé par un comité spécialisé**, en l'occurrence l'ECMA (European Computer Manufactures Association).

Afin de ne favoriser aucun des concepteurs précités, **cette association a décrété que ce langage de script porterait le nom commun de *ECMAScript*** dont les références se trouvent dans le standard ECMA-262. Ce dernier regroupe la totalité des fonctionnalités du langage Javascript aussi bien présentes dans le langage de Netscape ou de celui de Microsoft mais aussi d'autres qui ne seraient implémentées ni dans l'un, ni dans l'autre.

Enfin, **les programmeurs avertis ne ressentiront guère de difficultés à se familiariser à ce langage** et **les débutants devraient par une bonne connaissance du langage HTML et certainement par un bon investissement personnel réussir à rapidement maîtriser cet outil**. Dans tous les cas, le jeu en vaut la chandelle !

Voici quelques uns des éditeurs Javascript ou HTML offrant un module Javascript:

- WebExpert intègre un module Javascript en plus de ces fonctonnailté HTML,
- Frontpage accueille naturellement son propre langage JScript,
- HotMetal Pro offre également la possibilité d'écrire des scripts,
- DreamWeaver se distingue dans la création de page dynamique,
- Golive propose des fonctionnalités équivalentes.

2 / La structure

La syntaxe définit des règles d'écritures sans lesquelles un programme Javascript ne pourrait fonctionner correctement.

La syntaxe du langage Javascript s'appuie sur le modèle du Java, ressemblant lui-même à celui du C ou du C++.

Ainsi, les programmeurs ne rencontreront aucunes difficultés à se familiariser avec cette structure. Les dilettantes n'y verront pas plus d'obstacles.

2.1 / La casse

En ce qui concerne la casse, le langage Javascript y est extrêmement sensible. Ainsi, le strict respect des majuscules et minuscules est une condition sine qua non au bon fonctionnement de vos scripts.

En conséquence soyez vigilant sur la casse des variables et instructions.

Par exemple, écrivez systématiquement les instructions et les variables en lettres minuscules et les constantes en lettres majuscules.

```
var START = "début";

var Start = "un";

var start = 0;

var log = LOG10E;

var racine = SQRT2;
```

2.2 / Les espaces

L'insertion des espaces peut s'éffectuer n'importe où dans le script comme lors d'une rédaction habituelle.

C'est-à-dire, des espaces peuvent séparer les valeurs des opérateurs et eux-mêmes des identificateurs, etc..

```
//Dans ce cas les deux méthodes sont bonnes
var identificateur="Valeur";
var identificateur = "Valeur";

//Par contre ici le mot-clé var.
var initialisation = i + 1;

//ne peut être accoché à l'identificateur.
var initialisation=i+1;

//Cette méthode est correct.
document.write("Chapitre");

//Celle-ci l'est également.
document.write ( "Chapitre" );

//Ici ce n'est pas bon !
document . write ( "Chapitre" );
```

2.3 / Le point-virgule

Chaque commande doit être terminée par un point-virgule (;).

Bien que cela ne soit pas obligatoire, il est préférable pour de bonnes règles de programmation de respecter systématiquement cette règle.

```
var increment = i+1;

var jour = "Lundi";

if (choix == "non")
{
  document.write('Votre choix est l\'abandon');
}
```

2.4 / Le point

Dans les opérations mathématiques, un nombre à virgule flottante est séparé en France par une virgule, mais celle-ci a une signification particulière dans le langage Javascript.

C'est pourquoi, ces nombres doivent être séparés par un point (**.**) comme le montre ces exemples :

```
var PI = 3.141592654;

var prix = 2.50;

nb = 5.3 * 14.025;
```

2.5 / Les sauts de ligne

Les sauts de ligne peuvent être placés partout où se trouve un espace, y compris au sein d'une commande.

Néanmoins, afin d'éviter toutes ambiguités, il vaudra mieux éviter de découper sur plusieurs lignes une instruction car si les sauts de ligne ne sont pas correctement interpréter, un point-virgule pourrait être inséré automatiquement en fin de ligne rendant la commande inopérante.

```
var texte = "ce texte est bien trop long pour tenir sur une seule ligne,
c\'est pourquoi un saut de ligne sera nécessaire";

var init = 0;

document.write("document.write("<html><head>"
   + "<title>Rémunération sur Internet</title>
   + "</head><body bgcolor='#D5DECC'>"
   + "<a href='http://www.domaine.com/' target='_blank'>
   + "<img src='images/image.gif' border="0"></a>");
```

2.6 / Les commentaires

Les commentaires permettent de rendre votre code lisible et surtout d'en faciliter ultérieurement la maintenance.

En général, l'insertion de commentaire se fait soit en fin de ligne, soit sur une nouvelle ligne mais en aucun cas au sein d'une ligne de commande.

Il existe deux méthodes permettant d'intégrer des commentaires à vos scripts.

La première consiste à placer un double slash (//) devant le texte comme dans l'exemple ci-dessous :

```
var image = "fond.gif"; //image de fond

if (x=2) //redirection vers la seconde page
{
  url = ("page2.html"); //action...
}
else
{ //sinon retourne à la page d'accueil
  url = ("accueil.html");
}
```

La seconde solution est d'encadrer le texte par un slash suivi d'une étoile (/*) et la même séquence inversée (*/) comme le montre l'exemple suivant :

```
/*Voici un commentaire*/

/*
Encore un autre commentaire
*/
/*Ce commentaire est écrit
sur deux lignes*/
```

2.7 / Les types de valeurs

Le langage Javascript reconnaît plusieurs types de valeurs :

- **Les nombres entiers ou à virgule flottante** comme "42" ou "3.14159".
- **Les valeurs logiques** (Booléennes), *true* (vrai) et *false* (faux).
- **Les caractères** comme 'a', '5' '.', etc..
- **Les chaînes de caractères** comme "Bonjour !".
- *Null*, **un mot-clé spécial symbolisant une valeur nulle**; le *null* est aussi une valeur primitive. Parce que JavaScript est sensible à la casse, *null* n'est pas le même comme le *Null*, le *NULL*, ou une autre variante.
- *Undefined*, est **une propriété de niveau supérieur dont la valeur est non définie**; *undefined* est aussi une valeur primitive.

Ce jeu relativement restreint de types de valeurs ou de types de données, permet néanmoins d'exécuter des fonctions utiles avec vos applications.

Il n'y a aucune distinction explicite entre l'entier et les nombres réels.

Il n'y a également pas de type de données sous forme de dates explicites dans Javascript. Cependant, vous pouvez utiliser l'objet de Date et ses méthodes afin de manipuler des dates.

3 / Les variables

Vous utilisez des variables comme identificateurs pour affecter des valeurs qui peuvent être sollicitées n'importe où dans le script.

Ces variables peuvent contenir des littéraux composés soit de valeurs numériques, soit de chaînes de caractères.

Par ailleurs, elles doivent se conformer à certaines règles, comme d'éviter de donner à la variable un nom identique à celui d'un mot clé.

3.1 / Déclaration et affectation

Le mot-clé *var* permet de déclarer une ou plusieurs variables. Après la déclaration de la variable, il est possible de lui affecter une valeur par l'intermédiaire du signe d'égalité (=).

Si une valeur est affectée à une variable sans que cette dernière ne soit déclarée, alors Javascript la déclare automatiquement. Cependant, la lecture d'une variable non déclarée provoque une erreur risquant de perturber votre programme.

D'autre-part, une variable correctement déclarée mais dont aucune valeur n'est affectée, est indéfinie (undefined). Ainsi, il est préférable de sytématiquement après la déclaration d'une variable de lui associée une valeur.

```
//Déclaration de i, de j et de k.
var i, j, k;

//Affectation de i.
i = 1;

//Déclaration et affectation de prix.
var prix = 0;

//Déclaration et affectation de caractere
var caractere = ["a", "b", "c"];
```

3.2 / La portée

Dans le langage Javascript, les variables peuvent être globales ou locales.

Une variable globale est déclarée en début de script et est accessible à n'importe quel endroit du programme.

Une variable locale est déclarée à l'intérieur d'une fonction et n'est utilisable que dans la fonction elle-même. Dans certain cas, une variable n'a de portée qu'au sein des accolades au sein de laquelle elle a été déclarée.

Une variable globale peut être appelé au sein d'une fonction par l'intermédiaire du mot-clé *this*.

```
//Variables globales.
var i = 0;
j = 64;

function() {
  //Variables locales.
  var i = 1;
  j = 128;
  document.write(Valeurs de i et j : " + i + " " + j);
  ...
}
  ...
  //La variable k est locale et //n'est utilisable que dans la boucle for.
  for(k = 0; k < 10; k++=){
    document.write("valeur de k : " + k);
  }
  ...
  //Variable globale.
  var x = 72;
  //paramètre de la fonction.
  function affiche(x){
  //Appel de la variable globale à l'aide du mot-clé this.
    document.write("Valeurs : " + x + " " + this.x);
  }
  affiche(12);
```

3.3 / Affectation du type

Une particularité du langage Javascript est d'avoir des variable sans type (untyped), c'est-à-dire que le type comme les nombres entiers ou à virgule flottante, est automatiquement affecté à la variable.

```
//i est du type entier.
var i = i + 1;

//taux est du type à virgule flottante
var taux = 0.66;

//texte est du type string
var texte = "Une chaîne de caractère quelconque";
```

3.4 / Les littéraux

Les littéraux sont employés pour représenter des valeurs dans JavaScript.

Ceux-ci permettent de fixer des valeurs et non-pas des variables, que vous fournissez littéralement dans vos scripts.

"une chaîne de caractères"

true

2.789

'une autre chaîne de caractères'

0x26

3.5 / Les tableaux de littéraux

Un littéral de tableau est une liste de zéro ou plusieurs expressions, dont chacune représente un élément du tableau, entouré par des parenthèses carrées ([]).

Lors de la création d'un tableau employant un littéral de tableau, il est initialisé avec les valeurs indiquées comme ses éléments et sa longueur est calculé selon le nombre d'arguments indiqués.

```
semaine ["lundi","mardi","mercredi","jeudi",
         "vendredi","samedi","dimanche"]

nom ['Marc',,'Ludovic',,,'Angélique']
```

Dans le premier exemple, le littéral de tableau comporte sept éléments et une longueur de sept.

Dans le second, le littéral comporte trois éléments ainsi que trois vides. La longueur est ici de six avec :

```
nom[0] = 'Marc';
nom[2] = 'Ludovic';
nom[5] = 'Angélique'
```

3.6 / Les littéraux booléens

Le type booléen a deux valeurs littérales : true (vrai) et false (faux).

Ne confondez pas les valeurs primitives booléennes *true* et *false* avec les valeurs de l'objet *Boolean()*.

L'objet *Boolean()* est un emballage autour du type de données primitif Booléen.

```
var ouverte = true;
var fermee = false;
porte [ouverte, fermee]

var on = true;
var off = false;
lumiere [on, off]
```

3.7 / Les littéraux entiers

Les entiers peuvent être exprimés en décimal (base 10), hexadécimal (base 16) et octal (base 8). Un littéral entier décimal consiste en une séquence de chiffres situé entre 0 et 9.

Des entiers hexadécimaux peuvent inclure des chiffres de 0 à 9 et les lettres a-f et A-F. Des entiers octaux peuvent inclure seulement les chiffres 0-7.

Littéraux décimaux	Littéraux hexadécimaux	Littéraux octaux
0	0x0	00
1	0x1	01
2	0x2	02
3	0x3	03
4	0x4	04
5	0x5	05
6	0x6	06
7	0x7	07
8	0x8	010
9	0x9	011
10	0xA	012
11	0xB	013
12	0xC	014
13	0xD	015
14	0xE	016
15	0xF	017

3.8 / Les littéraux à virgule flottante

Un littéral de virgule flottante peut avoir les parties suivantes : un entier décimal, un point (.) une fraction sous forma d'un autre nombre décimal et un nombre avec exposant dont la partie exposant est constituée par un *e* ou *E* suivi d'un entier, qui peut être précédé par *+* ou -.

Un littéral de virgule flottante doit avoir au moins un chiffre et/ou bien un point décimal et/ou bien un *e* ou *E*.

1.2165

-3.5E9

2e-14

125.245E12

3.9 / Les littéraux objets

Un littéral objet est une liste de zéro ou plusieurs de couples de noms de propriété et les valeurs associées d'un objet, entouré par des accolades ({}).

Vous ne devez pas employer un littéral objet au début d'une déclaration. Cela conduira à une erreur.

```
var monnaie = "deutschmark";
function numeraire(nom)
{
  if(nom == "euro")
    return nom;
  else
    return "Désolé, la monnaie euopéenne n'est pas " + nom + ".";
}
nouveau = {france: "franc", europe: numeraire("euro"),
                allemagne: monnaie, angleterre: "livre" };

document.write(nouveau.france); // franc
document.write(nouveau.allemagne); // deutschmark
document.write(nouveau.europe); // euro
document.write(nouveau.angleterre); // livre
```

L'exemple suivant est un exemple de littéral objet. Le premier élément de l'objet de *nouveau* définit une propriété, *france*; le deuxième élément, la propriété europe, invoque une fonction (numeraire ("euro")); le troisième élément, la propriété *allemagne*, emploie la variable *monnaie* et enfin le dernier couple *angleterre* possède la valeur *livre*.

3.10 / Les littéraux chaînes de caractères

Un littéral chaîne de caractères est composé de zéro à plusieurs caractères entourés de guillemets doubles (") ou simples (').

Une chaîne de caractères doit être délimitée par les guillemets du même type, c'est-à-dire soit deux guillemets simples, soit deux guillemets doubles.

D'autre part, si la chaîne comporte un type de guillemets, il suffit alors de l'entourer par des guillemets de l'autre type.

```
"Disque de platine"
"Retour vers la page d'accueil"
'Disquette 3.5"'
```

Caractère	Description
\b	Un caractère en arrière
\f	Saut de page
\n	Saut de ligne
\r	Retour chariot
\t	Tabulation
\'	Apostrophe ou guillemet simple
\"	Guillemet double
\\	Backslash (\)
XXX	Séquence octale
\x*XX*	Séquence hexadécimale
\u*XXXX*	Séquence Unicode.

```
"Obtenir \"une récompense\"
pour l\'originalité de votre site"

"Décompactage dans : \n \t C:\\temp\\"

'Attention ! \r Un problème est survenu !
\r Relancez votre navigateur !'
```

3.11 / Les littéraux

Unicode est une norme universelle codant les caractères pour l'échange et l'affichage des principales langues écrites.

Il couvre les langues d'Amériques, d'Europe, du Moyen-Orient, d'Afrique, de l'Inde, de l'Asie et du Pacifique, aussi bien que des scripts historiques et des symboles techniques. Unicode tient compte de la conversion, le traitement et l'affichage de textes multilingues, aussi bien que l'utilisation de symboles communs techniques et mathématiques. Il espère résoudre les problèmes d'internationalisation de calcul multilingue, comme des différents standards de caractères nationaux.

Cependant, tous les scripts modernes ou archaïques ne sont pas actuellement soutenus.
Le jeu de caractère Unicode peut être employé pour tout les codages connus. Unicode a été créé après le jeu de caractère ASCII (American Standard Code for Information Interchange). Il utilise une valeur numérique et un nom pour chaque caractère.

Le codage de caractères spécifie l'identité du caractère et sa valeur numérique (la position du code), aussi bien que la représentation de cette valeur en bit. La valeur numérique su 16 bits (la valeur du code) est définie par un nombre hexadécimal et un préfixe U, par exemple, U+0041 représente A. Le nom unique pour cette valeur est la MAJUSCULE LATINE A.

Les fonctionnalités Unicode ne sont reconnus dans Javascript qu'à partir de la version 1.3.

Unicode est compatible avec les caractères ASCII et est supporté par beaucoup de programmes. Le 128 premier caractères Unicode correspondent aux caractères ASCII et ont la même valeur d'octet. Les caractères Unicode de U+0020 jusqu'à U+007E sont équivalents des caractères ASCII de 0x20 jusqu'à 0x7E. À la différence de l'ASCII, qui supporte l'alphabet latin et utilise le jeu de caractère à 7 bits, Unicode emploie une valeur de 16 bits pour chaque caractère. Il prend en compte des dizaines de milliers de caractères. La version 2.0 Unicode contient 38 885 caractères. Il supporte aussi un mécanisme d'extension, l'UTF (Unicode Transformation Format), nommé UTF-16, qui permet l'encocodage de plus d'un million de caractères en employant des couples de caractère 16 bits. UTF tourne le codage aux bits réelles.

3.11.1 / Affichage des caractères

Le langage Javascript utilise des séquences d'échappement d'Unicode diffèrente du Java. Dans JavaScript, la séquence d'échappement n'est jamais interprété comme un caractère spécial.

Par exemple, une séquence d'échappement fin de ligne à l'intérieur d'une chaîne de caractère ne la termine pas avant qu'elle ne soit interprété par la fonction.

JavaScript ignore n'importe quelles séquences d'échappement si elles sont employées dans des commentaires. Dans le Java, si une séquence d'échappement est employée dans une seule ligne de commentaire, elle est interprétée comme un caractère Unicode.

Pour un littéral chaîne de caractères, le compilateur de Java interprète les séquences d'échappement en premier. Par exemple, si un le caractère d'échappement fin de ligne (\u000A) est employé dans Java, il termine le littéral chaîne de caratère.

Dans le Java, cela mène à une erreur, parce qu'on ne permet pas de fins de ligne dans des littéraux chaînes de caractères.

Vous devez employer *n* pour un retour à la ligne dans un littéral chaîne de caractères. En Javascript, la séquence d'échappement fonctionne de la même manière qu'un saut de ligne *n*.

Vous pouvez utiliser Unicode pour afficher des caractères de langues différentes ou des symboles techniques. Pour que des caractères soient montrés correctement, une plateforme cliente comme le navigateur de Netscape ou de Microsoft doit supporter Unicode. De plus, une police Unicode appropriée doit être disponible chez le client et la plateforme cliente doit supporter la technologie Unicode.

Souvent, les polices Unicode ne montrent pas tous les caractères Unicode. Quelques plateformes, comme celle de Windows 95, fournissent un appui partiel pour Unicode.

Pour recevoir l'entrée du caractère non-ASCII, le client doit l'envoyer comme Unicode. L'utilisation d'une norme a étendu le clavier, le client ne peut pas facilement saisir des caractères supplémentaires supportés par Unicode. Souvent, la seule manière d'entrée des caractères Unicode est en utilisant des séquences d'échappement Unicode. La spécification Unicode, toutefois, n'exige pas l'utilisation de séquences d'échappement. Unicode trace une méthode pour une interprétation des caractères spéciaux Unicode employant un caractère composé. Il indique l'ordre des caractères qui peuvent être employés pour créer un caractère composé, où le caractère de base vient d'abord, suivi par unes ou plusieurs marques de non-espacement.

Cependant, les mises en oeuvre communes d'Unicode incluant la mise en oeuvre JavaScript ne soutiennent pas cette option. JavaScript n'essaye pas la représentation de l'Unicode des ordres se combinant. Autrement dit, une entrée d'un *a* et ′ ne produit pas *à*. JavaScript interprète un ′ comme deux caractères distincts 16 bits Unicode. Vous devez employer un séquence d'échappement Unicode ou un littéral Unicode pour le caractère *à*.

Voir les caractères Unicode disponibles.

3.11.2 / Les séquences d'échappement

Vous pouvez employer la séquence d'échappement d'Unicode dans des littéraux chaînes de caractères. L'ordre d'échappement comporte six caractères ASCII : u et un nombre hexadécimal à quatre chiffres. Par exemple, u00A9 représente le symbole de droit d'auteur.

Chaque ordre d'échappement d'Unicode dans Javascript est interprété comme un caractère.

Catégorie	Valeur Unicode	Description	Nom de format
Espace blanc	u0009	Tabulation	<TAB>
	u000B	Tabulation verticale	<VT>
	u000C	Saut de page	<FF>
	u0020	Espace	<SP>
Valeur de fin de ligne	u000A	Retour à la ligne	<LF>
	u000D	Retour chariot	<CR>
Valeurs Unicode complémentaires	u000b	Espace arrière	<bS>
	u0009	Tabulation horizontale	<hT>
	u0022	Double guillemet	"
	u0027	Single guillemet	'
	u005C	Backslash	\

```
var texte="Voici un exemple concret !"
        + "u000AUn saut de ligne a été inséré.";

var chaine="u0009 Maintenant cu0027est au tour"
          + "d'une tabulation et d'un simple guillemet.";

var phrase="u0022Des guillemets doubles "
          + "entourent cette phraseu0022";
```

3.11.3 / compatibilité avec le code ASCII et ISO

Unicode est entièrement compatible avec la Norme Internationale ISO/IEC 10646-1; 1993, qui est un sous-ensemble d'ISO 10646 et soutient l'ISO UCS-2 (Universal Caractere Set) qui utilise deux octets (soit 16 bits).

Javascript et les navigateurs supportent pour Unicode signifie que vous pouvez employer des caractères non-latins, internationaux et particulier, plus des symboles techniques spéciaux dans des programmes Javascript. Unicode fournit une méthode standard pour encoder le texte multilingue. Depuis qu'Unicode est compatible avec ASCII, les programmes peuvent utiliser des caractères ASCII.

Il est désormais possible d'utiliser des caractères Unicode non-ASCII dans les commentaires et les littéraux chaînes de caractères de Javascript.

3.12 / Les identificateurs

Un identificateur Javascript est un nom symbolique affecté d'une variable ou d'une fonction.

Il est composé de lettres, de chiffres de *0* à *9*, du caractère de soulignement ainsi que du caractère dollar *$* et doit commencer soit par une lettre, soit par un caractère de soulignement (_).

Javascript est sensible à la casse, les lettres incluses sont les caractères majuscules de *a* jusqu'à *Z* et les caractères minuscules de *a* jusqu'à *z*.

Les lettres accentuées sont exclus dans la composition d'un identificateur et à fortiori d'une variable.

Si le caractère de soulignement est autorisé en début de la chaîne de caractère, il est recommandé de ne pas en utiliser deux d'affilés, car cette séquence étant utilisée par plusieurs fonctions Javascript peut générer des bogues dans vos scripts.

```
var Identificateur = "littéral";

var increment_1 = 0;

var _texte = "Une chaîne de caractère quelconque";

//Cet identificateur est très mal construit !
var __chaîne = "Un double problème";
```

3.13 / Les mots réservés

Plusieurs mots sont réservés à un emploi bien précis dans Javascript.

C'est pourquoi, ces mots clés ne peuvent être utlisés pour dénommer un identificateur quelconque sinon des risques de dysfonctionnement peuvent apparître et rendre votre programmation inopérante.

Mots réservés			
abstract	else	instanceof	switch
boolean	enum	int	synchronized
break	export	interface	this
byte	extends	long	throw
case	false	native	throws
catch	final	new	transient
char	finally	null	true
class	float	package	try
const	for	private	typeof
continue	function	protected	var
debugger	goto	public	void
default	if	return	volatile
delete	implements	short	while
do	import	static	with
double	in	super	

```
/*Formules à éviter imérativement !*/
var true = "vrai";

var goto = "Retour à la page d'accueil";

var final = dernier - 1;
```

4 / Les instructions

Les instructions Javascript sont contenues dans chaque ligne d'un programme. En général, elles sont fermées par un point-virgule afin que le compilateur Javascript les identifie précisément.

Les instructions les plus simples sont notamment les déclarations de variables.

```
//Cette ligne est une instruction.
var identificateur="Valeur";

//Celle-ci en est une autre.
var init = i + 1;

//En voila une dernière.
document.write("Chapitre");
```

4.1 / Les groupes

En entourant un bloc d'instructions par des accolades, on crée un regroupement d'instructions utlisable au sein, notamment, d'une commande conditionnelle telle que *if... else* qui les éxecutera de la même manière qu'une unique instruction.

```
if (choix == "oui")
{
  i = i + 1;
  document.write("incrément = " + i);
}
```

4.2 / Les labels

Les labels permettent de dénommer une instruction ou un groupe d'instructions.

L'affectation d'un nom peut donner l'occasion à une autre instruction ou à une fonction d'y faire référence et donc de les éxécuter.

Un label est simplement appelé par son identificateur.

```
fiche_livre:
{
  var numero = "000000";
  var auteur = "nom";
  var titre = "titre";
  var editeur = "editeur";
}

function librairie()
{
  fichier_livre;
  //Instructions;
}
```

4.3 / Les instructions conditionnelles

Une instruction conditionnelle est un jeu de commandes qui s'exécute si une condition indiquée est vraie.

Le JavaScript supporte deux déclarations conditionnelles : *if... else* et *switch*.

```
if (condition)
{
  //première_instruction
}
else
{
  //seconde_instruction
}
```

4.3.1 / If... else

La condition *if* est utilisée dans le but d'exécuter certaines instructions si une condition logique est vraie; (*true*).

Si cette dernière est fausse (false), l'instruction facultative *else* exécute d'autres instructions.

Les commandes *if* et *else* peuvent contenir une ou plusieurs instructions délimitées par des accolades.

D'autre-part, il est autorisé d'utiliser l'imbrication de l'instruction *if* autant de fois qu'il est nécessaire dans l'énumération des conditions et instructions associées.

```
if (condition)
{
  //première_instruction
}
else
{
  //seconde_instruction
}
if (condition)
{
  //premières_instructions
}
else if (Nième_condition)
{
  //instructions
}
  ...
else
{
  //instructions
}
```

4.3.2 / With

L'instruction *with* définit un objet par défaut. Cette objet est ensuite utilisé dans le bloc d'instructions attaché à *with*.

```
with(objet)
{
  //Instructions;
}

/*L'instruction with permet de travailler directement avec une propriété d'un objet.*/
var resultat, x, y;
with (Math)
{
  x = 2;
  y = 10;
  //sans with resultat = Math.pow(x, y)
  resultat = pow(x, y);
}
```

4.3.3 / Switch... case

Cette instruction permet de vérifier le contenu d'une variable à partir de nombreuses valeurs.

L'instruction évalue successivement les différentes valeurs des labels jusqu'à ce qu'il trouve la bonne contenue entre ses paranthèses.

S'il échoue dans sa comparaison, alors, il exécute les instructions contenues dans le label *default:*.

Enfin, une instruction *break* termine les instructions de *case* et saute à la fin de *switch*.

```
switch (Expression)
{
 case Premier_label:
   Premières_instructions;
  break;
  ...
 case Nième_label:
    //Nièmes_instructions;
   break;
 default:
   //Instruction_par_défaut
  break;
}
```

4.4 / Les boucles

Une boucle est un bloc de programme qui s'exécute à plusieurs reprises jusqu'à ce qu'un état indiqué soit réuni.

Le langage JavaScript supporte *for*, *do while*, *while*, et des boucles de *label*.

Cette dernière n'est pas elle-même une instruction de boucle, mais est fréquemment utilisée avec elle.

En outre, les instructions *break* et *continue* sont utilisées généralement à l'intérieur des boucles.

Une autre instruction, *for... in*, peut exécuter des commandes à plusieurs reprises mais est couramment utilisé pour la manipulation d'objet.

4.4.1 / For

La boucle *for* exécute une ou plusieurs instructions jusqu'à ce que la condition spécifiée retourne *false*.

L'instruction *for* contient une donnée initiale, une condition et une commande d'incrémentation ou de décrémentation, le tout séparé par des points-virgules.

La boucle *for* fonctionne littéralement comme suit : "Pour une donnée initiale, jusqu'à ce que la condition soit atteinte et par pas de... exécuter les instructions suivantes".

D'ailleurs, il faut noter que la donnée initiale n'est valable que pour le démarrage de la boucle. Par la suite cette donnée n'est plus prise en compte.

```
for (Initialisation; Condition; Incrémentation)
{
  //Instructions;
}
/*La boucle for permet d'afficher à l'écran les nombres de 0 à 10*/
for (i = 0; i <= 10; i++)
{
  document.write('i = ' + i + '\n');
}
```

4.4.2 / Do... while

Cette boucle exécute un groupe d'instructions jusqu'à ce que la condition renvoie *false*.

La boucle *do... while* est toujours exécutée au moins une fois malgré l'état de la condition spécifiée, car cette dernière n'est évaluée qu'après son activation étant donné la position de la condition.

```
do
{
  //Instructions;
}
while(Condition);

/*La boucle do... while s'exécutera jusqu'à ce que l'utilisateur saisira un nom différent de nom*/
var Nom = 'nom';
do
{
   Nom = prompt('Entrez votre nom','Votre nom ici');
}
while(Nom == 'nom');
```

4.4.3 / While

Cette boucle exécute un groupe d'instructions jusqu'à ce que la condition renvoie *false*.

Contrairement à la précédente, la boucle *while* est exécutée seulement si la condition spécifiée n'est pas remplie.

```
while(Condition)
{
  //Instructions;
};

/*La boucle while exécutera une soustraction jusqu'à ce que i atteigne 11*/
var calc = 0, i = 0;
while(i <= 10)
{
  i++;
  calc = 10-i;
  document.write('10 - ' + i + ' = ' + calc + '\n');
}
```

4.4.4 / Continue

L'instruction *continue* permet d'interrompre également le déroulement d'une boucle à l'exception qu'elle ne la termine pas immédiatement mais la court-circuite.

En fait elle permet de sauter un groupe d'instructions pour reprendre, par la suite, le fonctionnement normal de la boucle.

```
continue;

/*L'instruction continue court-circuitera la boucle while
lorsque x sera égal à 5 provoquant la non-impression de ce chiffre.*/
var x = 0;
//La boucle effectuera ses dix itérations
while (x >= 10)
{
  x++;
  if (x == 5)
  {
    continue;
  }
  document.write(i + '\n');
}
```

4.4.5 / Break

L'instruction *break* permet d'interrompre le déroulement d'une boucle.

```
break;

while (x >= 100)
{
  if (x == 50)
  {
    break;
  }
  document.write(i + '\n');
}
```

4.4.6 / For... in

L'instruction *for... in* déplace une variable indiquée dans toutes les propriétés d'un objet.

Pour chacune des propriétés de l'objet, la boucle *for... in* exécutera les instructions indiquées.

La variable indiquée dans l'instruction est utilisée pour stocker le nom de la propriété en cours d'exploitation.

```
for(Variable in Objet)
{
  //Instructions;
}
/*La variable a parcourt le tableau Tab en retenant
le nom de la propriété permettant de restituer le contenu du tableau par les instructions entre des accolades*/
var Tab = new Array("un", "deux", "trois", "quatre", "cinq");
var a;
var i = 0;
for( a in Tab)
{
  i++;
  document.write('Contenu (' + i + ') = ' + Tab[a] + '\n');
}
```

4.5 / La gestion des exceptions

Le langage Javascript offre aux programmeurs la possibilité de gérer plus élégamment les exceptions, des anomalies pouvant apparaître subrepticement dans le code.

A cet effet, plusieurs instructions permettent de traiter des erreurs, comme *throw* dont la fonction est de lancer une exception, *try... catch* qui la récupère et la manipule, ou encore *try... finally* qui libère une ressource.

4.5.1 / Throw

L'instruction *throw* est utilisé pour lancer un exception.

Lorsqu'une exception est lancée, une expression contenant la valeur de l'exception doit être spécifiée.

```
throw expression;

throw "erreur202"; //Une chaîne de caractères

throw 400; // Un nombre

throw null; //La valeur null
```

4.5.2 / Try... catch

L'instruction *try... catch* désigne un bloc de commandes à essayer et spécifie qu'une réponse doit lancer une exception. Si une exception est lancée, l'instruction *try... catch* la saisira.

L'instruction *try* permet de tester un bloc d'instructions dont l'une est *throw*. Si cette dernière est activée, alors l'instruction *catch* stockera le nom de l'exception par l'intermédiaire de son identificateur et transmettra, à l'aide de ses instructions, l'objet d'exception vers le gestionnaire d'erreurs.

```
try
{
  //Instructions à essayer;
};
catch (Identificateur)
{
  //Instructions;
}

/*Si x est inférieur à 0, l'exception erreur est lancée dans try.*/
try
{
  if (x>0)
  {
    resultat=Math.sqrt(x);
  }
  else
  {
    throw "erreur";
  };
}
//ex stocke erreur
catch (ex)
{
  //retourne l'expression indiquée
  resultat="Opération Impossible";
  //transmet l'objet d'exception au gestionnaire d'erreurs
  logMyErrors(ex);
}
/*catch saisit l'exception erreur et la traite en fonction de ses instructions.*/
```

4.5.3 / Try... finally

Le bloc *finally* contient des instructions à exécuter après l'exécution du bloc *try... catch* mais avant les instructions suivant le bloc précité.

Le bloc *finally* est exécuté si une exception est lancée, même si aucune instruction *catch* ne traîte l'exception.

Cette instruction peut être utilisée pour libérer une ressource par exemple.

```
try
{
  //Instructions;
}
catch
{
  //Instructions;
}
finally
{
  //Instructions;
}

try
{
  OuvertureFichier(); // attache une ressource
  EcritureFichier(donnee);
}
catch
{
  CaptureErreur();
}
finally
{
   FermetureFichier(); //ferme la ressource
}
```

5 / Les expressions et opérateurs

Les expressions permettent d'évaluer, par l'intermédiaire des opérateurs Javascript, différentes valeurs ou variables littérales.

Le langage Javascript possède un jeu complet d'opérateurs permettant de multiples combinaisons d'expressions.

5.1 / Les expressions

Une expression est n'importe quel ensemble valide de littéraux, de variables, d'opérateurs, et d'expressions qui correspond à une valeur simple.

Cette valeur peut être un nombre (42), une chaîne de caractères (olivier), ou une valeur logique (true).

Conceptuellement, il y a deux types d'expressions : celles qui assignent une valeur à une variable, et celles qui ont simplement une valeur.

Par exemple, l'expression *i = 0* est une expression qui affecte à *i* la valeur *zéro*. Cette expression elle-même correspond à *zéro*.

Ce genre d'expressions fait appel à des opérateurs d'affectation. D'autre part, l'expression *1 + 5* est évaluée logiquement à *6* soit le résultat de l'addition. Dans ce cas, elle n'exécute pas une affectation. Les signes utilisés dans de telles expressions sont désignés sous le nom d'opérateurs.

Type d'expression	Description	Exemple
Arithmétique	Correspond à un nombre réel ou entier	i + 1; prix = 20.50;
Logique	Correspond aux valeurs logiques *true* et *false*	choix = true; faux = false;
Chaîne de caractères	Correspond à une séquence de lettres ou de chiffres.	adresse = "1 av. Raspail"; cp = "75000";

5.2 / Les opérateurs

Le langage JavaScript a différents types d'opérateurs. Cette section décrit les opérateurs et contient des informations sur la priorité d'opérateur.

Les opérateurs peuvent être binaires ou unaires ou encore ternaire. Un opérateur binaire a besoin de deux opérandes, une avant et une après l'opérateur.

Un opérateur unaire n'a besoin que d'une unique opérande avant ou après l'opérateur.

Un opérateur ternaire, c'est-à-dire l'opérateur conditionnel, a besoin de trois opérandes.

5.3 / La priorité des opérateurs

La priorité des opérateurs détermine l'ordre qui leur est appliqué en évaluant une expression. Vous pouvez ignorer la priorité d'opérateur en utilisant des parenthèses.

La table suivante décrit la priorité des opérateurs, du moins élevé vers le plus élevé

Opérateur	Associativité
,	De gauche à droite
= += -= *= /= %= <<= >>= >>>= &= |= ^=	De gauche à droite
? :	De droite à gauche
||	De gauche à droite
&&	De gauche à droite
|	De gauche à droite
^	De gauche à droite
&	De gauche à droite
== !=	De gauche à droite
< <= > >= instanceof	De gauche à droite
<< >> >>>	De gauche à droite
+ -	De droite à gauche
* / %	De gauche à droite
(type) new	De droite à gauche
unaire + unaire - préfixe ++ préfixe -- ~ !	De gauche à droite
() [] . suffixe ++ suffixe --	De gauche à droite

5.4 / Les opérateurs d'affectation

Un opérateur d'affectation assigne la valeur de l'opérande gauche basé sur la valeur de l'opérande droite.

L'opérateur d'affectation de base est le signe d'égalité (=), qui assigne la valeur de son opérande droite à son opérande gauche. C'est-à-dire, *droit = gauche* assigne la valeur de *gauche* à *droit*.

Les autres opérateurs d'affectation sont sténographiés pour des exécutions standard, comme montré dans la table suivante.

Raccourci	Opération	Description
x = y		y est affecté à x
x += y	x = x + y	y est additionné à x
x -= y	x = x - y	y est soustrait de x
x *= y	x = x * y	x est multiplié par y
x /= y	x = x / y	x est divisé par y
x %= y	x = x % y	le reste de x/y est affecté à x
x <<= y	x = x << y	x est décalé de y bits vers la gauche
x >>= y	x = x >> y	x est décalé de y bits vers la droite
x >>>= y	x = x >>> y	x est décalé de y bits vers la droite avec extension de zéros
x &= y	x = x & y	Opération logique ET sur des entiers au niveau du bit
x ^= y	x = x ^ y	Opération logique OU EXCLUSIF sur des entiers au niveau du bit
x |= y	x = x | y	Opération logique OU sur des entiers au niveau du bit

5.5 / Les opérateurs arithmétiques

Les opérateurs arithmétiques prennent des valeurs numériques (des littéraux ou des variables) comme opérandes et renvoient une valeur numérique.

Les opérateurs arithmétiques standards sont l'addition (*+*), la soustraction (-), la multiplication (*), et la division (*/*).

Ces opérateurs travaillent de la même façon que la plupart des autres langages de programmation, excepté l'opérateur */* qui renvoie un résultat à virgule flottante dans JavaScript, et non pas un nombre tronqué comme dans certains langages, tels que le C ou le Java.

Opérateur	Description	Exemple
x **%** y	L'opérateur modulo retourne le reste de la division *x/y*	20 % 3 //retourne 2
x **++***	Cet opérateur unaire permet l'incrémentation de la valeur *x*	i++ //équivaut à i=i+1
x **--***	Cet opérateur unaire permet la décrémentation de la valeur *x*	i-- //équivaut à i=i-1

- x	Cet opérateur unaire retourne la valeur inverse à *x*	i = 1
		-i //renvoie donc -1
		i = -5
		-i // renvoie donc 5

* Si l'un de ses opérateurs est placé avant la variable, alors la valeur de la variable sera incrémentée (++) ou décrémentée (--) avant son utilisation. Par exemple pour *i=0*, *i++* donnera 0 et *++i* donnera 1 de même que *i--* donnera 0 et *--i* donnera -1.

5.6 / Les opérateurs de comparaisons

Ce type d'opérateur compare ses opérandes et renvoie une valeur logique en fonction du résultat. Si la comparaison est vraie, la valeur logique *true* est retournée.

Les opérandes peuvent être des valeurs numériques ou des chaînes de caractères.

Les chaînes de caractères sont comparées sur la base du standard lexicographique, en utilisant des valeurs d'Unicode.

Opérateur	Description	Exemples
x == y	Si la valeur *y* est égale à *x*, l'opérateur retourne *true*. Dans ce cas, si le type de donnée ne correspond pas alors Javascript tentera de convertir les opérandes dans le type approprié afin d'effectuer la comparaison.	if (choix == 1)...;
x != y	Si la valeur *y* est différente de *x*, l'opérateur retourne *true*	if (valeur != prix)
x === y	Si la valeur de *y* est strictement égale (valeur et type) à *x*, alors l'opérateur retourne *true*	if (paragraphe = texte)
x !== y	Si la valeur de *y* est strictement différente à *x*, l'opérateur retourne *true*	if (ref !== "A000000")
x > y	Si la valeur de *y* est supérieure à *x*, l'opérateur retourne *true*	if (montant > 1500)
x >= y	Si la valeur de *y* est supérieure ou égale à *x*, l'opérateur retourne *true*	if (hab >= pop)
x < y	Si la valeur de *y* est inférieure à *x*, l'opérateur retourne *true*	if (numero < page)
x <= y	Si la valeur de *y* est inférieure ou égale à *x*, l'opérateur retourne	if (fin <= premier)

5.7 / Les opérateurs au niveau du bit

Au niveau du bit les opérateurs traitent leurs opérandes comme ensemble de 32 bits (des zéros et ceux), plutôt qu'en tant que nombres décimaux, hexadécimaux, ou octals.

Par exemple, le numéro décimal neuf a une représentation binaire de 1001. Au niveau du bit les opérateurs exécutent leurs exécutions sur de telles représentations binaires, mais ils renvoient des valeurs numériques standard de JavaScript.

Opérateur	Usage	Description	Exemple

AND	***x & y***	Chaque position binaire de l'opérande *x* est comparée avec la position correspondante de l'opérande *y*. Dans ce cas, si un *1* est trouvé dans l'une ou l'autre des opérandes, alors l'opérateur retourne un *1*, sinon il renvoie un *0*.	1101 & 0111
			//retourne 0101
			0001 & 1000
			//retourne 0000
			1001 & 0001
			//retourne 0001

OR	***x | y***	Chaque position binaire de l'opérande *x* est comparée avec la position correspondante de l'opérande *y*. Dans ce cas, si un *1* est trouvé dans les deux opérandes soit dans l'une, soit dans l'autre, alors l'opérateur retourne un *1*, sinon il renvoie un *0*.	1101 | 0111
			//retourne 1111
			0001 | 1000
			//retourne 1001
			1001 | 0001
			//retourne 1001

XOR	***x ^ y***	Cette opérateur *OU EXCLUSIF* fonctionne comme le précédent à la seule exception que le *1* ne doit se trouver que dans l'une ou l'autre des opérandes pour produire le résultat *1* sinon il renvoie un *0*.	1101 ^ 0111
			//retourne 1010
			0001 ^ 1000
			//retourne 1001
			1001 ^ 0001
			//retourne 1000

NOT	***~ a***	L'opérateur unaire *NOT* retourne un *0* lorsque l'opérande contient un *1* et un *1* pour un *0*.	~1101
			//retourne 0010
			~0001
			//retourne 1110
			~1001
			//retourne 0110

Décalage à gauche	***x << n***	Cet opérateur déplace la représentation binaire de *x* de *n* bits à gauche. La valeur de ces *n* bits est de zéro.	15 << 2 //0000 1111 décalé de 2 bits produit 0011 1100 et donc retourne 60
Décalage à droite avec préservation du signe	***x >> n***	Cet opérateur déplace la représentation binaire de *x* de *n* bits à droite et conserve le signe.	15 >> 2 //0000 1111 décalé de 2 bits produit 0000 0011 et donc retourne 3 -15 >> 2 //1111 0001 décalé de 2 bits produit 1111 1100 et donc retourne -4
Décalage à droite avec extension de zéros	***x >>> n***	Cet opérateur décale la première opérande du nombre *n* indiqué de bits vers la droite. Des bits excédents décalés en dehors vers la droite sont rejetés. Les bits à zéro sont décalés vers la gauche. En fait, cet opérateur donne les mêmes résultats que le précédent (**>>**).	11 >>> 2 //0000 1011 décalé de 2 bits produit 0000 0010 et donc retourne 2

5.8 / Les opérateurs booléens

Les opérateurs logiques sont typiquement utilisés avec des valeurs booléennes; quand elles le sont, ils renvoient une valeur booléenne.

Cependant, les opérateurs *&&* et // renvoient réellement la valeur de l'une des opérandes indiquées, ainsi si ces opérateurs sont utilisés avec des valeurs non-booléennes, ils peuvent renvoyer une valeur non-booléenne.

Opérateur	Usage	Description	Exemple
AND	x **&&** y	L'opérateur renvoie *x*, s' il peut être converti en *false*; autrement, retourne *y*. Ainsi, une fois utilisé avec des valeurs booléennes, le *&&* renvoie *true* si les deux opérandes sont vraies; autrement, il retourne *false*.	(a < 10) && (b > 10) //Si a et b sont inférieures à 10, l'opérateur retourne true
OU	x || y	L'opérateur renvoie *x* s' il peut être converti en *true*; sinon il retourne *y*. Ainsi, une fois utilisé avec des valeurs booléennes, l'opérateur // renvoie *true* si l'une ou l'autre des opérandes est vraie ou si toutes les deux sont fausses, sinon il retourne *false*.	(a >= 1) || (b == "fin") //Si *a* est supérieur ou égal à 1 ou/et si *b* est égal à *fin* alors l'opérateur renvoie *true*, sinon il retourne *false*
NOT	!x	L'opérateur renvoie *false* si son unique opérande peut être convertie en *true*, sinon il retourne *false*.	!(a <= 100) //Si *a* est inférieur ou égal à 100 alors l'opérateur retourne *false*, sinon il renvoie *true*.

5.9 / Les opérateurs de concaténations

En plus des opérateurs de comparaison, qui peuvent être utilisés sur des valeurs de chaîne de caractères, l'opérateur d'enchaînement + permet d'enchaîner deux valeurs de chaîne de caractères ensemble, en renvoyant une autre qui est le résultat de la concaténation des deux opérande.

L'opérateur d'affectation + = peut également être utilisé pour enchaîner des chaînes de caractères. Le contenu de la variable est concaténé à l'opérande de droite.

```
"mon" + "programme";
//retourne "mon programme"
texte = "Un programme";
texte_2 = "Javascript"
texte += texte_2;
//retourne "Un programme Javascript"
```

5.10 / L'opérateur conditionnel

L'opérateur conditionnel est le seul opérateur du langage JavaScript qui utilise trois opérandes.

L'opérateur ternaire fonctionne comme une instruction *if... else*. Si la condition initiale renvoie true, alors la première instruction est éxécutée sinon la deuxième est activée.

```
Condition ? Instruction_1 : Instruction_2;

(i < 10) ? v = true : v = false;

(jour == "Lundi")
      ? (document.write("Bonne reprise du travail !"))
                : (document.write("Bon courage !"))

(Prix >= 100)
       ? (Diff_pos = Prix - 100)
                : (Diff_neg = 100 - Prix);
```

5.11 / Delete

L'opérateur *delete* efface un objet, la propriété d'un objet, ou un élément à un incrément donné dans un tableau.

là où objectName est le nom d'un objet, la propriété est une propriété existante, et l'incrément est un nombre entier représentant l'emplacement d'un élément dans un tableau.

La quatrième forme est légale seulement à l'intérieur d'une instruction *with*, pour effacer une propriété d'un objet.

Vous pouvez utiliser l'opérateur *delete* pour effacer des variables déclarées implicitement mais pas celles déclarées par l'intermédiaire de *var*.

Si l'opération d'effacement réussit, il place la propriété ou l'élément à *undefined* (non défini). L'opérateur *delete* renvoie *true* si l'exécution est possible, sinon il renvoie *false* si l'exécution n'est pas possible.

Lors d'un effacement d'un élément d'un tableau, la longueur du tableau n'est absolument pas affectée.

Par contre, les emplacement des éléments éffacés deviennent *undefined* et les éléments ne font plus partis du tableau.

```
//retourne true car objet a été défini par l'utilisateur
delete objet;

montant = 0;
//retourne true car montant a été déclaré implicitement sans l'instruction var
delete montant;

var i = 1;
//retourne false car i a été déclaré avec var
delete i;

objet.nom = "table";
//retourne true car la propriété objet.nom a été définie par l'utilisateur
delete objet.nom;

//retourne false car la propriété Math.cos(x) est prédéfinie dans Javascript
delete Math.cos(x);

Tab = ["un", "deux", "trois", "quatre", "cinq"];
//retourne true et le tableau conserve une longueur de 5
delete Tab[2];
```

5.12 / In

La propriété *in* retourne *true* si la propriété indiquée est dans l'objet donnée, sinon elle retourne *false*.

```
Tab = ["un", "deux", "trois", "quatre", "cinq"];

//retourne true car la propriété 3 est bien dans l'objet Tab
3 in Tab

//retourne true car la propriété acos(x) est bien dans l'objet Math
acos(x) in Math

//retourne false car le tableau ne contient pas plus de cinq cellules
6 in Tab
```

5.13 / Instanceof

Cette propriété retourne *true* si l'objet spécifié est du même type que l'opérande de droite.

En général, la propriété *instanceof* est utilisée lors d'un besoin de confirmation sur le type d'un objet afin d'éxécuter subséquemment une ou plusieurs instructions.

```
objet instanceof objet_type

/*anniversaire est du type date (année, mois, jour, heure, minute)*/
var anniversaire = new date(2000, 1, 1, 0, 0)

//retourne true puisque anniversaire est du type date
anniversaire instanceof date
```

5.14 / New

L'opérateur *new* est utilisé pour créer une nouvelle instance ou un nouveau type d'objet défini par l'utilisateur ou de l'un des types d'objets prédéfinis, *Array*, *Boolean*, *Date*, *Function*, *Image*, *Number*, *Object*, *Option*, *RegExp*, ou *String*.

Sur le serveur, il est possible de l'utiliser avec *DbPool*, *Lock*, *File*, ou *SendMail*.

```
nouvel_objet = new type_objet(litteral_1,...)

texte = new String("Une chaîne de caractère");
```

5.15 / This

Cette propriété est utilisé pour faire référence à un objet courant.

Ceci permet d'accéder à un objet sans passer par son identificateur.

```
this.nom_propriete

//dénomination complète de l'objet nom
window.document.forms[0].elements[1] nom

//raccourci vers l'élément nom
this.nom
```

5.16 / Typeof

L'opérateur *typeof* renvoie une chaîne de caractères indiquant quel est le type de l'opérande.

Les paranthèses sont optionnels.

```
typeof (opérande);

var i = 1;
typeof i; //retourne number

var titre="Les raisins de la colère";
typeof titre; //retourne string

var jour = new Date();
typeof jour; //retourne object

var choix = true;
typeof choix; //retourne boolean

var cas = null;
typeof cas; //retourne object
typeof parseFloat; //retourne function
typeof Math; //retourne object (IE 5.*, NS 6.*, NS 4.78, Opera 6.*, Opera 5.*
typeof Math; //retourne function NS 3.*, Opera 3.*
```

5.17 / La virgule

L'opérateur virgule , évalue simplement ses deux opérandes et renvoie la valeur de la deuxième.

Cet opérateur est principalement utilisé à l'intérieur d'un boucle *for*, pour permettre à des variables multiples d'être mises à jour chaque fois par l'intermédiaire de la boucle.

Par exemple, si "Tab" est un tableau à deux dimensions avec 10 éléments de chaque côté, le code suivant utilise l'opérateur virgule pour incrémenter deux variables en même temps. Le code imprime les valeurs des éléments diagonaux dans le tableau.

```
for (i=O , j=9 ; i<=9 ; i++, j--)
{
  document.write("Tab[" + i + "," + j + "]= " + Tab[i,j])
}
```

Tab[i,j]	0	1	2	3	4	5	6	7	8	9
0										0,9
1									1,8	
2								2,7		
3							3,6			
4						4,5				
5					5,4					
6				6,3						
7			7,2							
8		8,1								
9	9,0									

5.18 / Void

L'opérateur *void* indique une expression à évaluer sans renvoyer une valeur particulière si ce n'est *undefined* (non-définie).

En fait *void* accueille une opérande qui n'a aucune fonction puisque, quelque soit la valeur de cet argument, l'opérateur retournera toujour *undefined*.

Néanmoins, il est possible d'utiliser l'opérateur *void* pour spécifier une expression comme un lien hypertexte.

L'expression est évaluée mais n'est pas chargée à la place du document courant.

D'autre-part, l'opérateur *void* permet également de vérifier si une variable a été définie.

Enfin, il peut parfois être nécessaire de retirer une valeur de retour d'une fonction, et *void* sans acquitte parfaitement.

```
void(opérande);
<!-- Ce lien est inactif -->
<a href="javascript:void(0);">lien</a>

&lt;!-- Ce lien permet de soumettre un formulaire -->
<a href="javascript:void(document.form.submit())">Soumettre</a>

choix = void('non'); //retourne undefined

nombre = void(8); //retourne undefined
/*La condition if permet de vérifier si la variable est définie*/
if(var1 = void(0)) {document.write("var1 n'est pas défini !")}

/*Un mot est bien affiché mais la valeur de retour de la fonction sera undefined*/
void(document.write("Un mot"));
```

6 / Les expressions régulières

Les expressions régulières sont des modèles utilisés pour vérifier des combinaisons de caractère dans les chaînes de caractères.

Dans JavaScript, les expressions régulières sont également des objets.

Ces modèles sont utilisés avec les méthodes *exec* et *test* de *RegExp*, et avec les méthodes de chaîne de caractères *match*, *replace*, *search*, et *split*.

6.1 / Créations d'expressions régulières constantes

Les initialiseurs d'objets fournissent la compilation de l'expression régulière quand la séquence type est évaluée.

Quand l'expression régulière demeurera constante, utilisez ceci pour une meilleure exécution.

```
var modele = /Expression/;

var recherche = /Chaîne/;

var texte = /[Cc]ha[iî]ne/;
```

Exemple [voir]

```
<html>
<head>
  <script language="JavaScript" type="text/javascript">
    function valider(){
       var email = document.form.email.value;
       var modele = /^[a-z0-9\-_\.]+@[a-z0-9]+\.[a-z]{2,5}$/i;
       if (modele.test(email))
         alert("Votre adresse email est valide !")
       else
         alert("Votre adresse email est invalide !");
       return false;
    }
  </script>
<body>
    <form name="form">
    <input type="text" name="email" value="adresse@email.com"/>
    <input type="submit" value="Valider" onclick="valider();">
    </form>
</body>
</html>
```

6.2 / Création avec RegExp

L'utilisation de la fonction de constructeur fournit la compilation d'exécution de l'expression régulière.

Utilisez la fonction de constructeur quand vous savez que le modèle de l'expression régulière n'est pas fixé précisément, ou si vous ne connaissez pas le modèle dont l'obtention pourrait dépendre de l'entrée d'un utilisateur.

Une fois que vous avez une expression régulière définie, si l'expression régulière est utilisée dans toute la séquence type, et si sa source change, vous pouvez employer la méthode *compil* pour compiler une nouvelle expression régulière pour une réutilisation plus efficace.

```
var modele = new RegExp("Expression");

var recherche = new RegExp("html")

var texte = new RegExp("html|css")
```

Exemple [voir]

```
<html>
<head>
  <script language="JavaScript" type="text/javascript">
    function valider(){
        var nombre = document.form.nombre.value;
        var modele = new RegExp("^[0-9]*(\\.|,)?[0-9]+$");
        if (modele.test(nombre))
          alert("Le nombre est correct !");
        else
          alert("La valeur saisie n'est pas un nombre !")
        return false;
    }
  </script>
<body>
    <form name="form">
        <input type="text" name="nombre" value="0.123456789"/>
        <input type="submit" value="Valider" onclick="valider();"/>
    </form>
</body>
</html>
```

6.3 / Les attributs d'extensions de recherche

Il existe deux attributs *g* et *i* pour les expressions régulières. Ils doivent se placer derrière le second slash (/) de l'expression.

L'attribut *g* permet d'effectuer une recherche globale, c'est-à-dire que la totalité du texte doit être parcourue à la recherche de l'expression spécifiée.

Tandis que l'attribut *i* détermine que la casse ne doit pas être prise en compte dans la recherche, c'est-à-dire que la recherche s'effectuera sur un texte sans aucune distinction entre les minuscules ou les majuscules.

```
//recherche sans distinction de casse
var modele = /JAVASCRIPT/i;

//recherche globale
var modele = /balise/g;

//recherche globale et sans distinction de casse
var modele = new RegExp("mot", "gi");
```

Exemple [voir]

```
<html>
<head>
 <script language="JavaScript" type="text/javascript">
   function valider(){
      var email = document.form.email.value;
      var modele = /^[a-zA-Z0-9\.\-_]+@[a-zA-Z0-9]+\.[a-zA-Z]{2,5}$/i;
      if (modele.test(email))
        alert("Votre adresse email est valide !")
      else
        alert("Votre adresse email est invalide !");
      return false;
   }
 </script>
<body>
   <form name="form">
      <input type="text" name="email" value="mon.adresse@email.com"/>
      <input type="submit" value="Valider" onclick="valider();">
   </form>
</body>
</html>
```

6.4 / Les caractères spéciaux

Les caractères spéciaux permettent d'affiner une recherche, là où une expression régulière simple ne peut suffire.

Les caractères et autres symboles spéciaux représentent des caractères non-imprimables comme des sauts de lignes (\n), des tabulations (\t) ou des options particulières ou encore des plages de lettres ou de chiffres, etc.

Caractère	Description	Exemple
\\, \/, *, \+, \?, etc.	Symbolise les caractères correspondants	var modele=/2*8\+1/; //recherche la chaîne *2*8+1*
^	Symbolise le début d'une chaîne de caractères	var modele=/^Le/; //recherche *Le* en début de phrases
$	Symbolise la fin d'une chaîne de caractères	var modele=/\.$/; //recherche un *point* à la fin d'une chaîne de caractères
*****	Le caractère précédent peut intervenir de zéro à plusieurs fois	var modele=/10*/; //recherche les nombres commençant par *1* avec éventuellement des *zéros*
+	Le caractère précédent doit être trouvé de une à plusieurs fois	var modele=/\t+/; //recherche de une à plusieurs *tabulations*
?	Le caractère précédent est optionnel	var modele=/mots?/; //recherche *mot* au singulier ou au pluriel
.	Le caractère représente un caractère quelconque sauf un saut de ligne	var modele=/.n.?/; //recherche tous les mots de deux ou éventuellement de trois lettres contenant un *n* au centre comme *en* ou *une*
()	Les parenthèses divisent le modèle en plusieurs zones	var modele=/java(script)?/; //recherche le mot *javascript* ou *java*
x|y	Le pipe agît comme un OU, une des possibilités doit être trouvée	var modele=/html|css/; //recherche *html* ou *css* ou les deux
{n}	Le caractère précédent doit être trouvé un nombre *n* fois	var modele=/continu{2}m/; //recherche le mot *continuum*
{n,m}	Le caractère précédent doit être trouvé au moins *n* fois et au plus *m* fois	var modele=/10{3,9}/; //recherche d'un nombre de 10^3 ou de 10^9
{n, }	Le caractère précédent doit être trouvé au moins *n* fois et ou plus	var modele=/10{1,}/; //recherche un nombre de *10* à 10^n
[...]	Les crochets contiennent des caractères, dont l'un doit être trouvé	var modele=/parti[res]?/; //recherche l'un des mots suivants: *parti*, *partir*, *partie* ou *partis*
[...-...]	Les crochets contiennent une plage de caractères dont l'un doit être trouvé	var modele=/[a-b]/; //recherche un caractère faisant partie des lettres minuscules
[^...]	La recherche porte sur des caractères qui ne sont pas contenus entre les crochets	var modele=/exécut[^(ion)]/; //ne recherche pas le mot *exécution* mais *exécuta* ou *exécuter*, etc.
[\b]	Le caractère recherché est un espace blanc	var modele=/[\b]\./; //recherche un *espace* avant un *point*

\b	La recherche s'effectue sur les limites d'un mot comme un espace ou une nouvelle ligne	var modele=/x\b/; //recherche la lettre *x* suivie d'un espace ou d'un saut de ligne comme *peau**x***, *sceau**x*** mais pas *e**x**iger*
\B	La recherche s'effectue sur un espace qui n'est pas placé entre un mot et un espace	var modele=/x/B/; //recherche la lettre *x* qui n'est pas suivie d'un espace ou d'un saut de ligne comme ***x*ylophone** ou **a*x*e** mais pas **eu*x***
\c*X*	Où *X* est un caractère de contrôle. La recherche porte sur un caractère de contrôle dans une chaîne de caractères	var modele=/\cA/; //recherche un caractère de contrôle *control-A*
\d	La recherche s'effectue sur un chiffre quelconque de 0 à 9	var modele=/\d000/; //recherche un nombre comme *1000*, *2000*, etc.
\D	La recherche porte sur un caractère quelconque hormis un chiffre de 0 à 9	var modele=/10\D10/; //recherche une chaîne de caractère comme *10,10* ou *10.10* ou encore *10=10*
\f	La recherche porte sur un saut de page	var modele=/\f/; //recherche un saut de page
\n	La recherche porte sur un caractère de saut de ligne	var modele=/\.\n/; //recherche un *saut* *de ligne* après un *point*
\r	La recherche porte sur un retour charriot	var modele=/\.\r/; //recherche un *retour charriot* après un *point*
\s	Cette occurrence permet une recherche sur un espace, une tabulation, un saut de ligne, etc.	var modele=/\s/; //recherche un de ces caractères *[\f\n\r\t\v].*
\S	La recherche porte sur tous les caractères hormis, l'espace, la tabulation, le saut de ligne, etc.	var modele=/\S/; //recherche tous les caractères sauf ceux-ci *[^\f\n\r\t\v].*
\t	La recherche porte sur une tabulation	var modele=/\t/; //recherche une *tabulation*
\v	La recherche porte sur un tabulation verticale	var modele=/\v/; //recherche une *tabulation verticale*
\w	La recherche porte sur tous caractères faisant partie d'un mot	var modele=/**w**s[\b]/; //recherche la lettre *s* entre un caractère **[A-Za-z0-9_]** et un espace
\W	La recherche porte sur tous caractères ne faisant pas partie d'un mot	var modele=/\W/; //recherche un caractère sauf ceux-ci *[^A-Za-z0-9_]*
/(...)...*n*/	Le nombre *n* permet de répéter le contenu entre parenthèses à l'endroit où il se situe	var modele=/(["']réussite**1**)/; //recherche du mot *réussite* entre guillemets simples ou doubles devant et derrière
\o*octal* **\x*hex***	Cette formulation permet de trouver un caractère par son code octal ou hexadécimal	var modele=/\x0028ASCII\x0029/; //recherche le mot *ASCII* entre des parenthèses

6.5 / Les méthodes et les objets

Les expressions régulières sont utilisées avec les méthodes de *RegExp*, c'est-à-dire *test* et *exec* et avec les méthodes de chaîne de caractères *replace*, *match*, *search*, et *split*.

Méthode	Description
Exemple	
exec	Une méthode qui exécute une recherche pour une expression régulière dans une chaîne de caractères et retourne un tableau d'information.

```
var modele=/mot/;
var texte="Exécution d'une recherche sur un mot";
var tableau=modele.exec(texte);
document.write(tableau);
```

test	Une méthode qui teste l'expression régulière dans une chaîne de caractères et renvoie *true* ou *false*.

```
var modele=/mot/;
var texte="Cette méthode teste si un mot se trouve dans la chaîne de caractère";
if(modele.test(texte)) {
  document.write("Le test est positif !");
} else {
  document.write("Le test est négatif !"); };
```

match	Une méthode qui exécute une recherche pour une expression régulière dans une chaîne de caractères et retourne un tableau d'information ou la valeur *null* en cas d'échec.

```
var modele=/mot/;
var texte="Encore une exécution d'une recherche sur un mot";
document.write(modele.match(texte));
```

search	Une méthode qui teste l'expression régulière dans une chaîne de caractères et renvoie un index des expressions trouvées ou *-1* si la recherche échoue.

```
var modele=/mot/;
var texte="Un mot est recherché dans ce texte";
document.write(texte.search(modele));
```

replace	Une méthode qui exécute une recherche pour une expression régulière dans une chaîne de caractères et remplace une occurrence de la chaîne de caractère par une autre.

```
var modele=/expressions\srégulières/;
var texte="Comment remplacer un mot avec les expressions régulières";
document.write(texte.replace(modele, "regular expression"));
```

split	Une méthode qui utilise une expression régulière dans le but de fractionner une chaîne de caractères.

```
var mois="janvier, février, mars";
document.write(mois.split(/,\s/)[0]+ "\n");
document.write(mois.split(/,\s/)[1]+ "\n");
document.write(mois.split(/,\s/)[2]+ "\n");
```

6.6 / Les propriétés et les méthodes de RegExp

Cet objet contient le modèle d'une expression régulière et possède des méthodes et des propriétés dans le but d'utiliser une expression régulière dans une recherche ou un remplacement d'une partie d'une chaîne de caractères.

Propriété	Raccourci	Description	Type	Syntaxe
compile		Effectue la compilation d'une expression régulière pendant l'exécution d'un script.	méthode	RegExp.compile("modèle", attribut);
	$n	Représente une partie d'une chaîne de caractères, en fait un mot dans un texte.	propriété	modele= /(\w+)\s(\w+)\s(\w+)/; texte="un texte long"; modif= texte.replace(modele, "$1 $3 $2"); document.write(modif) //Ecrit *un long texte*
constructor		Spécifie la fonction qui crée un objet.	propriété	RegExp.constructor
input	$_	Représente la chaîne de caractères originale pour laquelle une correspondance a été trouvée avec l'expression régulière.	propriété	RegExp.input //ou RegExp.$_
lastIndex		Spécifie le point de départ de la recherche dans un texte. Elle est utilisable seulement en recherche globale, *g*.	propriété	RegExp.lastIndex
lastMatch	$&	Correspond aux derniers caractères d'un . $& est un autre nom pour la même propriété.	propriété	RegExp.lastMatch //ou RegExp.$&
lastParent	$+	Correspond à la dernière sous-chaîne entre parenthèses. $+ est un autre nom pour la même propriété.	propriété	RegExp.lastParent //ou RegExp.$+
leftContext	$`	Correspond à une sous-chaîne précédant une autre plus récente. $` est un autre nom pour la même propriété.	propriété	RegExp.leftContext //ou RegExp.$`
multiline	$*	définit si la recherche devra être effectuée sur une ou plusieurs lignes. $* est un autre nom pour la même propriété.	propriété	RegExp.multiline //ou RegExp.$*
prototype		Définit une propriété qui est partagée par tous les objets d'un type spécifié. Il faut utiliser cette propriété pour ajouter des propriétés aux objets créés avec l'opérateur new.	propriété	RegExp.prototype
rightContext	$'	La sous-chaîne suivant la plus récente correspondance. $' est un autre nom pour la même propriété.	propriété	RegExp.rightContext //ou RegExp.$'
source		Une propriété en lecture seule contenant le texte du modèle.	propriété	RegExp.source
toSource		Retourne une chaîne représentant le code source de l'objet.	méthode	RegExp.toSource
toString		Convertit un objet en chaîne de caractères.	méthode	RegExp.toString
valueOf		Retourne la valeur primitive de l'objet.	méthode	RegExp.valueOf

7 / Les fonctions

Les fonctions sont des modules fondamentaux dans JavaScript. Une fonction est une procédure de JavaScript où un ensemble d'instructions accomplit une tâche spécifique.

Pour utiliser une fonction, il faut d'abord lui définir un nom, puis un ou plusieurs arguments et évidemment une ou plusieurs instructions.

Enfin, le nom de la fonction et ses arguments lui permettront d'être appelée dans un endroit quelconque d'un programme Javascript.

```
nom_fonction(argument1,...)=instructions

Aire(longueur, largeur)=longueur*largeur
```

7.1 / Function

En Javascript comme dans beaucoup d'autres langages, la fonction est créée par l'instruction *function* contenant entre paranthèses une liste de paramètres séparés par des virgules et entre accolades un ensemble d'instructions terminé par *return* qui permet de retourner le résultat de la fonction.

```
function nom(param1, param2,...)
{
  //Instructions;
}

function surface(base, hauteur)
{
  return (base*hauteur)/2;
}
```

7.2 / new Function

Une fonction peut être créée par le constructeur d'objet *new* associé à *Function()*.

Entre les parenthèses de cette dernière, sont placés les arguments encadrés par des guillemets et séparés par des virgules ainsi que les instructions.

L'appel de ce genre de fonction est strictement identique à la première.

```
nom_fonction=new Function("param1","param2",..., "Instructions");

var factoriel = new Function("n", instructions);

instructions: //un label
{
  if ((n==0)||(n==1))
  {
    return 1;
  }
  else
  {
    resultat=(n * factoriel(n-1));
    return resultat;
  }
}
```

7.3 / Appel d'une fonction

Une fonction est opérante à partir du moment où elle est appelée dans un programme.

C'est-à-dire, tant qu'une fonction n'est pas sollicitée, elle n'aura donc aucune influence sur l'ensemble du script.

L'appel d'une foncion est simple puisqu'il suffit d'utiliser son nom et de renseigner les arguments si besoin est.

```
nom_fonction (arg1, arg2,...)

Remplacement(paragraphe);

var Aire=surface(100, 50);
```

7.4 / Les arguments

Les arguments sont transmis à la fonction par l'intermédiaire d'une affectation de valeur entre les arguments lors de l'appel de la fonction et les paramètres de la fonction elle-même.

```
function nom_fonction (rep_param1, rep_param2,...)
{ ... };
nom_fonction (arg1, arg2,...);
/*L'appel provoque les affectations rep_arg1=arg1 et rep_arg2=arg2*/

function carre(x){return x*x;};
var aire=carre(100);
/*L'appel provoque l'affectation x = 100*/
```

7.5 / Les tableaux d'arguments

Les arguments d'une fonction sont mis à jour dans un tableau. En conséquence, l'utilisation des arguments passés vers une fonction peut être effectuée par *arguments[i]* où *i* est la position de l'argument dans le tableau et débute par zéro.

Le total du nombre d'arguments passés est indiqué par la propriété *length* (*arguments.length*).

```
function calcul(x)
{
  var resultat=arguments[0]*arguments[0];
  return resultat;
}

function calcul(x,y)
{
  var resultat=arguments[0]*arguments[1];
  return resultat;
}

function imprime(a,b,c,d,e)
{
  for (var i=0; i<arguments.length; i++)
  {
    document.write(argument[i]+"\n");
  }
  return;
}
```

7.6 / Eval

Cette fonction exécute un code Javascript à partir d'une chaîne de caractères.

```
eval(chaîne_de_caractères);
```

```
...
<script language="JavaScript">
  function evaluation()
  {
    document.formulaire.calcul.value=eval(document.formulaire.saisie.value);
  }
</script>
...
<form name="formulaire">
  Saisissez une expression mathématique :
    <input type="text" name="saisie" maxlength="40" size="40">
    <input type="button" value="evaluation." onclick="evaluation()">
    <input type="text" name="calcul" maxlength="40" size="40">
</form>
...
```

7.7 / isFinite

Cette fonction évalue un argument et détermine si la valeur est un nombre fini.

Si c'est le cas, elle renvoie *true* ou *false* si ce n'est pas un nombre ou l'infini positif ou infini négatif.

```
isFinite(Argument)

isFinite(240) //retourne true

isFinite("Un nombre") //retourne false
```

7.8 / isNaN

Cette fonction évalue un argument pour déterminer s'il ne s'agit pas d'un nombre (NaN : Not a Number).

Si c'est le cas, elle renvoie *true* ou bien *false* si c'est un nombre.

```
isNaN(Argument)

isNaN("un nombre") //retourne true

isNaN(20) //retourne false
```

Exemple [voir]

```
<html>
<head>
 <script language="JavaScript" type="text/javascript">
  function valider(){
     var nombre = document.form.nombre.value;
     if(!isNaN(nombre))
       alert("Le nombre est correct !");
     else
       alert("La valeur saisie n'est pas un nombre !")
     return false;
  }
 </script>
<body>
   <form name="form">
      <input type="text" name="nombre" value="0.123456789"/>
      <input type="submit" value="Valider" onclick="valider();">
   </form>
</body>
</html>
```

7.9 / parseFloat

Cette fonction analyse une chaîne de caractères et retourne un nombre décimal.

Si l'argument évalué n'est pas un nombre, la fonction renvoie *NaN* (Not a Number).

```
parseFloat(Chaîne_de_caractère)

var numero="125";

var nombre=parseFloat(numero);
//retourne le nombre 125
```

7.10 / parseInt

Cette fonction analyse une chaîne de caractères et retourne un nombre entier de la base spécifiée.

Cette dernière peut prendre les valeurs *16* (hexadécimal) *10* (décimal), *8* (octal), *2* (binaire).

```
ParseInt(Chaîne_de_caractère, racine)

var prix=30.75;
var arrondi = parseInt(prix, 10); //retourne 30
```

7.11 / Number

Cette fonction convertit l'objet spécifié en valeur numérique.

Number(Objet)

```
var jour = new Date("December 17, 1995 03:24:00");
//converit la date en millisecondes
alert (Number(jour));
```

7.12 / String

Cette fonction convertit l'objet spécifié en chaîne de caractères.

```
String(Objet)

jour = new Date(430054663215);
//Convertit le nombre en date Mois jour, Annéee etc.
alert (String(jour));
```

7.13 / Escape

Cette fonction retourne la valeur hexadécimale à partir d'une chaîne de caractère codée en ISO-Latin-1.

```
escape("Chaîne_de_caractère")

//retourne %21%26%
escape("!&")
```

7.14 / Unescape

Cette fonction retourne la chaîne de caractère codée en ISO-Latin-1 à partir d'une valeur hexadécimale (entre *0x0* et *0xFF*) ou d'une valeur entière symbolisé par *%xx* (entre *0* et *255*).

```
unescape("Code");

unescape("%23%26") //retourne #&
```

Exemple :

```
<form name="formulaire">
  <input type="text" name="saisie" maxlength="40"
          size="40" value="Saisissez un texte">
  <input type="button" value="Conversion ASCII." onclick="conversion()">

  Conversion en mode "escape" :
  <input type="text" name="conv_escape" maxlength="100" size="100">

  re-conversion en mode normal :
  <input type="text" name="conv_unescape" maxlength="40" size="40">
</form>

<script language="JavaScript">
  function conversion()
  {
    document.formulaire.conv_escape.value=
          escape(document.formulaire.saisie.value);
    document.formulaire.conv_unescape.value=
          unescape(document.formulaire.conv_escape.value);
  }
</script>
```

7.15 / Return

L'instruction *return* permet d'indiquer la valeur de retour d'une fonction. D'autre part, elle permet également de terminer une fonction.

Ainsi, une fonction pourrait contenir des blocs d'instructions conditionnels où chacun serait ponctué par une instruction *return*.

```
function nom(param1, param2,...)
{
  //Instructions;
  return Résultat;
}

function Remplacement(texte)
{
  var ok="modification effectuée";
  var echec="modification impossible";
  var modele=/(\w+)\s(\w+)/;
  if (modele.test(texte))
  {
    modif=texte.replace(modele, "$1$2");
    return ok;
  }
  else
  {
    return echec;
  }
}
```

8 / Les objets

Le langage JavaScript est conçu sur un paradigme simple basé sur des objets.

Un objet est construit à partir de propriétés, de méthodes et d'événements permettant de travailler sur tous les éléments d'une page HTML.

A - B - C - D - E - F - G - H - I - J - K - L - M -
N - O - P - Q - R - S - T - U - V - W - X - Y - Z

Objet	Propriétés	Méthodes	Evénements
a	hash host hostname href pathname port protocol search target text x y	handleEvent	onClick onDblClick onKeyDown onKeyPress onKeyUp onMouseDown onMouseOut onMouseUp onMouseOver
ActiveXObject			
Anchor	name text (Anchor) x y		
Area	hash host hostname href pathname port protocol search target text x y	handleEvent	onDblClick onMouseOut onMouseOver
Array	constructor index input length (Array) prototype	concat (Array) join pop push reverse shift slice (Array) splice sort toSource toString unshift valueOf	
Boolean	constructor prototypet	toSource toString valueOf	
Button	form name type value	blur click focus handleEvent	onBlur onClick onFocus onMouseDown onMouseUp
Checkbox	checked defaultChecked form name type value	blur click focus handleEvent	onBlur onClick onFocus

Date	constructor prototype	getDate getDay getFullYear getHours getMilliseconds getMinutes getMonth getSeconds getTime getTimezoneOffset getUTCDate getUTCDay getUTCFullYear getUTCHours getUTCMilliseconds getUTCMinutes getUTCMonth getUTCSeconds getVarDate getYear parse setDate setFullYear setHours setMilliseconds setMinutes setMonth setSeconds setTime setUTCDate setUTCFullYear setUTCHours setUTCMilliseconds setUTCMinutes setUTCMonth setUTCSeconds setYear toGMTString toLocaleString toSourcetoString toUTCString UTC valueOf	
Dictionary	Add Exists Items Keys Remove RemoveAll		
document	alinkColor anchors applets bgColor classes cookie domain embeds fgColor formName forms height ids images lastModified layers	captureEvents close contextual getSelection handleEvent open releaseEvents routeEvent write writeln	onClick onDblClick onKeyDown onKeyPress onKeyUp onMouseDown onMouseUp

	linkColor links plugins referrer tags title URL vlinkColor width		
Drive	AvailableSpace DriveLetter DriveType FileSystem FreeSpace IsReady Path RootFolder SerialNumber ShareName TotalSize VolumeName		
Drives	Count Item		
Enumerator		atEnd item moveFirst moveNext	
Error	description number		
Event	data height layerX layerY modifiers pageX pageY screenX screenY target type which width x y		
File	Attributes DateCreated DateLastAccessed DateLastModified Drive Name ParentFolder Path ShortName ShortPath Size Type	Copy Delete Move OpenAsTextStream	
Files	Count Item		
FileSystemObject	Drives	BuildPath CopyFile CopyFolder	

		CreateFolder CreateTextFile DeleteFile DeleteFolder DriveExists FileExists FolderExists GetAbsolutePathName GetBaseName GetDrive GetDriveName GetExtensionName GetFile GetFileName GetFolder GetParentFolderName GetSpecialFolder GetTempName MoveFile MoveFolder OpenTextFile	
FileUpload	form name type value	blur focus handleEvent select	onBlur onChange onFocus
Folder	Attributes DateCreated DateLastAccessed DateLastModified Drive Files IsRootFolder Name ParentFolder Path ShortName ShortPath Size SubFolders Type	Copy Delete Move OpenAsTextStream	
Folders	Count Item	Add	
Form	action elements encoding length method name target	handleEvent reset submit	onReset onSubmit
Function	arguments arguments.callee arguments.caller arguments.length arity caller constructor length (Function) prototype	apply call toSource toString valueOf	
Global	NaN (Global) Infinity	escape eval isFinite	

		isNaN parseFloat parseInt unescape	
hidden		form name type value	
History	current length next previous	back forward go	
Image	border complete height vspace lowsrc name src vspace width	handleEvent	onAbort onError onKeyDown onKeyPress onKeyUp onLoad
JavaArray	length	toString	
JavaClass			
JavaObject	length	toString	
JavaPackage			
JSObject		call equals eval getMember getSlot getWindow removeMember setMember setSlot toString	
Layer	above background below bgColor clip.bottom clip.height clip.left clip.right clip.top clip.width document left name pageX pageY parentLayer siblingAbove siblingBelow src top visibility window x y	captureEvents handleEvent load moveAbove moveBelow moveBy moveTo moveToAbsolute releaseEvents resizeBy resizeTo routeEvent	onBlur onFocus onLoad onMouseOut onMouseOver

	zIndex		
Link	hash host hostname href pathname port protocol search target text x y	handleEvent	onClick onDblClick onKeyDown onKeyPress onKeyUp onMouseDown onMouseOut onMouseOver onMouseUp
Location	hash host hostname href pathname port protocol search	reload replace	
Math	E LN2 LN10 LOG2E LOG10E PI SQRT1_2 SQRT2	abs acos asin atan atan2 ceil cos exp floor log max min pow random round sin sqrt tan	
MimeType	description enabledPlugin suffixes type		
navigator	appCodeName appName appVersion language mimeTypes platform plugins userAgent	javaEnabled plugins.refresh preference savePreferences taintEnabled	
Number	constructor MAX_VALUE MIN_VALUE NaN (Number) NEGATIVE_INFINITY POSITIVE_INFINITY prototype	toSource toString valueOf	

Object	constructor prototype	eval toSource toString unwatch valueOf watch	
Option	defaultSelected index length selected text value		
Packages	className java netscape sun		
Password	defaultValue form name type value	blur focus handleEvent select	onBlur onFocus
Plugin	description filename length name		
Radio	onBlur onClick onFocus	checked defaultChecked form name type value	blur click focus handleEvent
RegExp	$1...$9 $_ $* $& $+ $` $' constructor global ignoreCase index input lastIndex lastMatch lastParen leftContext multiline prototype rightContext source	compile exec test toSource toString valueOf	
Expression régulière	lastIndex source	compile exec test	
Reset	form name type value	blur click focus handleEvent	onBlur onClick onFocus

screen	availHeight availLeft availTop availWidth colorDepth height pixelDepth width		
Select	form length name options selectedIndex type	blur focus handleEvent	onBlur onChange onFocus
String	constructor length (String) prototype	anchor big blink bold charAt charCodeAt concat (String) fixed fontcolor fontsize fromCharCode indexOf italics lastIndexOf link match replace search slice (String) small split strike sub substr substring sup toLowerCase toSourcetoString toUpperCase valueOf	
Style	align backgroundColor backgroundImage borderBottomWidth borderColor borderLeftWidth borderRightWidth borderStyle borderTopWidth clear color display fontFamily fontSize fontStyle fontWeight lineHeight listStyleType	borderWidths margins paddings	

	marginBottom marginLeft marginRight marginTop paddingBottom paddingLeft paddingRight paddingTop textAlign textDecoration textIndent textTransform whiteSpace width		
Submit	onBlur onClick onFocus	form name type value	blur click focus handleEvent
Text	defaultValue form name type value	blur focus handleEvent select	onBlur onChange onFocus onSelect
Textarea	defaultValue form name type value	blur focus handleEvent select	onBlur onChange onFocus onKeyDown onKeyPress onKeyUp onSelect
TextStream	AtEndOfLine AtEndOfStream Column Line	Close Read ReadAll ReadLine Skip SkipLine Write WriteBlankLines WriteLine	
VBArray		dimensions getItem lbound toArray ubound	
window	closed crypto defaultStatus document frames history innerHeight innerWidth length location locationbar menubar name offscreenBuffering opener outerHeight outerWidth pageXOffset	alert atob back blur btoa captureEvents clearInterval clearTimeout close confirm crypto.random crypto.signText disableExternalCapture enableExternalCapture find focus forward handleEvent	onBlur onDragDrop onError onFocus onLoad onMove onResize onUnload

	pageYOffset parent personalbar screenX screenY scrollbars self status statusbar toolbar top window	home moveBy moveTo open print prompt releaseEvents resizeBy resizeTo routeEvent scroll scrollBy scrollTo setHotKeys setInterval setResizable setTimeout setZOptions stop Propriétés pour l'objet Dictionary Count Item Key	

8.1 / ActiveXObject

ActiveXObject **active et renvoie une référence vers un objet exposé aux autres applications ou outils de programmation par l'intermédiaire des serveurs Automation.**

Compatibilité

Disponibilité

JScript 1

Constructeur

new ActiveXObject("application.type_objet", serveur)

Exemples [voir]

Cet exemple renvoie les versions des applications Excel, Word et Powerpoint installées sur la machine cliente.

```
<html>
 <head>
  <script language="JScript">
   function GetAppVersion()
   {
    var app_excel = new ActiveXObject("Excel.Application");
    var app_word = new ActiveXObject("Word.Application");
    var app_powerpoint = new ActiveXObject("Powerpoint.Application");

    document.write('Version d\'Excel : ' + app_excel.Version);
    document.write('<br>Version de Word : ' + app_word.Version);
    document.write('<br>Version de Powerpoint : ' + app_powerpoint.Version);
   }
  </script>
 </head>
 <body onload="javascript:GetAppVersion()">
 </body>
</html>
```

La première ligne permet d'accéder à l'application Word et la seconde à un document Word.

```
var app_word = new ActiveXObject("Word.Application");
var doc_word = new ActiveXObject("Word.Document");
```

L'objet document créé ci-dessus est rendu visible par l'intermédiaire de l'objet Application.

```
doc_word.Application.Visible = true;
```

L'objet document est enregistré.

```
document_word.SaveAs("essai.doc");
```

Le document est fermé par la méthode Quit de l'objet Application.

```
doc_word.Application.Quit();
```

Référence :

Microsoft

8.2 / Anchor

***Anchor* est la cible d'un lien hypertexte.**

Compatibilité


Disponibilité

Javascript 1.2 côté client

Correspondance HTML

```
<a name="ancre"> ... </a>
```

Héritage

Herite de HTMLElement

Synopsis

```
document.anchors[i]
document.anchors.length
```

Propriétés

Anchor hérite des propriétés d'HTMLELement et définit ou écrase les propriétés suivantes.

Propriété	Description
name	Le nom d'un ancrage.
text	Le texte associé au point d'ancrage NE 4.
x	La coordonnée x du point d'ancrage NE 4.
y	La coordonnée y du point d'ancrage NE 4.

Exemples [voir]

```
<html>
  <body>
    <a name="ancre">Première ancre</a>
    <a name="ancre1">Seconde ancre</a>
    <a name="ancre2">Troisième ancre</a>
    <a name="ancre3">Quatrième ancre</a>
    <a name="ancre4">Cinquième ancre</a>
    <script language="javascript">
      document.write("La page contient " + document.anchors.length + " ancres.")
    </script>
  </body>
</html>
```

L'exemple suivant permet de créer une ancre dans une page :

```
var texte="Index X";
msgWindow.document.write(texte.anchor("ancre_X"));
```

Le script ci-dessus crée cette ancre :

```
<a name="ancre_X">Index X</a>
```

En savoir plus :

Netscape

8.3 / applet

***applet* correspond à un applet Java incorporé dans une page HTML.**

Compatibilité


Disponibilité

Javascript 1.1 côté client

Correspondance HTML

```
<applet> ... </applet>
```

Synopsis

```
document.applets[i]
document.applets.length
document.appletName
```

Propriétés

Les propriétés d'un *applet* sont identiques aux méthodes publiques de l'appet java qu'il représente.

Exemples

```
<html>
  <body>
    <applet
          code="prog_java.class"
          name="application_ecommerce">
    </applet>
    <script language="javascript">
      document.write("La page contient " + document.applets.length + " applet.")
    </script>
  </body>
</html>
```

Dans cet exemple, les trois lignes permettent de faire référence à l'applet java ci-dessus :

```
document.application_ecommerce;
document.applets["application-ecommerce"];
document.applets[0];
```

En savoir plus :

Netscape Microsoft

8.4 / Arguments

***Arguments* représente les arguments et les propriétés d'une fonction.**

Compatibilité


Disponibilité

Javascript 1.1

Synopsis

```
arguments
FunctionName.arguments[i]
```

Propriétés

Les propriétés d'un Applet sont identiques aux méthodes publiques de l'appet java qu'il représente.

Propriété	Description
callee	la fonction qui tourne à cet instant.
caller	le conteste d'appel NE 4.
length	le nombre d'arguments passés à la fonction.

Exemples [voir]

```
<html>
 <head>
  <script language="javascript">
   function Ouverture(page,ancre,cible,type)
   {
    nb = arguments.length;
    arg = "";
    for (var i = 1; i < nb; i++)
    {
     arg += arguments[i] + separator;
    }
    document.write("La fonction contient " + nb + " arguments.")
    document.write("Les arguments sont : " + arg + ".")
   }
  </script>
 </head>
 <body>
  <a href="javascript:Ouverture('index.html','chap3','self','http')">Un lien</a>
 </body>
</html>
```

En savoir plus :

Netscape

8.5 / Array

***Array* permet de créer des tableaux et de travailler avec eux.**

Compatibilité


Disponibilité

Javascript 1.1

Constructeur

```
new Array();
new Array(i);
new Array(element,element1,...,elementn);
```

Propriétés

Propriété	Description
length	la taille du tableau.

Méthodes

Méthode	Description
concat(valeur,...)	concatène des tableaux.

`var tableau = new Array('un','deux','trois','quatre','cinq');`
`var tableau2 = new Array('six','sept','huit','neuf','dix');`
`tableau1.concat('tableau2');`	

join(séparateur)	forme une chaîne de caractère à partir du tableau.
`var tableau = new Array('push','pop','reverse','shift');` `tableau.join(', ');`	
pop()	supprime et retourne le dernier élément du tableau.
`var tableau = new Array('o','ko','Mo','Go','To');` `tableau.pop();`	
push(valeur,...)	ajoute des éléments à un tableau (NE 4).
`var tableau = new Array('0');` `tableau.push('1','2','3','4');`	
reverse()	permute les éléments du tableau.
`var tableau = new Array('0','1','2','3');` `tableau.reverse();`	
shift()	décale les éléments du tableau.
`var tableau = new Array('février','mars','avril','mai','juin');` `tableau.shift();`	
slice(début, fin)	retourne une partie du tableau.
`var tableau = new Array(45,77,20,87,10,32);` `tableau.slice(1, 2);`	
sort(fonction_tri)	trie les éléments du tableau.

```
var tableau = new Array('Marc','Angélique','Edith','Annick','Isabelle');
tableau.sort();
```

splice (début, nb_élt_à_effacer, valeur, ...)	insére, supprime ou remplace des éléments du tableau.

```
var tableau = new Array('a','1','2','&','\?','f','g');
tableau.splice(1, 4, 'b', 'c', 'd', 'e');
```

toString()	convertit un tableau en une chaîne de caractères.

```
var tableau = new Array('Ceci','est','un','programme','javascript','.')
tableau.toString()
```

unshift()	ajoute des éléments au début d'un tableau.

```
var tableau = new Array('Mardi','Mercredi','Jeudi','Vendredi','Samedi','Dimanche')
tableau.unshift('lundi')
```

Exemple [voir]

```
<html>
  <body>
    <script language="javascript">
      var tableau = new Array('lundi','mardi','mercredi','jeudi',
                              'vendredi','samedi','dimanche')
      document.write("<table>")
      for(var i = 0; i < tableau.length; i++)
      {
        document.write("<tr><td bgcolor='#EECC00'>" + tableau[i] + "</td></tr>")
      }
      document.write("</table>")
    </script>
  </body>
</html>
```

En savoir plus :

Netscape Microsoft

8.6 / Boolean

Boolean **permet de créer ou de convertir des valeurs booléennes.**

Compatibilité


Disponibilité

Javascript 1.1

Constructeur

```
new Boolean(valeur) //fonction de construction. Boolean(valeur) //fonction de conversion.
```

Propriétés

Propriété	Description
objet.constructor	spécifie la fonction qui crée un objet.
nom_objet.prototype	représente le prototype pour cette classe.

Méthodes

Méthode	Description
tosource()	retourne un objet littéral représentant l'ojet booléen spécifié.
tostring()	convertit une valeur booléenne en une chaîne de caractères.
ValueOf()	renvoie la valeur primitive de l'objet spécifié.

Exemple

```
<html>
  <body>
    <script language="JScript">
      var porte = new Boolean(true)
      var variable = porte.toString()
      document.write('La valeur booléenne est actuellement à ' + variable);
    </script>
  </body>
</html>
```

Création d'objets booléens avec la valeur initiale à false :

```
chaine = new Boolean("");
porte = new Boolean('false');
faux = new Boolean(0);
```

Création d'objets booléens avec la valeur initiale à true :

```
numero = new Boolean(1);
vrai = new Boolean('true');
chaine = new Boolean('debut');
```

En savoir plus :

Netscape Microsoft

8.7 / Button

Button **reprtésente un bouton poussoir.**

Compatibilité


Disponibilité

Javascript 1.0

Correspondance HTML

<input ...>

Héritage

Input et HTMLElement

Synopsis

form.elements[i]
form.name

Propriétés

Propriété	Description
form	spécifie le formulaire contenant le bouton.
name	correspond à l'attribut NAME de INPUT.
type	correspond à l'attribut TYPE de INPUT.
value	affiche un texte sur le bouton

Méthodes

Méthode	Description
blur	enlève le focus sur le bouton.
click	simule un clique de souris sur le bouton.
focus	donne le focus au bouton.
handleEvent	invoque le Handler pour spécifier un événement.

Evénements

Evénement	Description
onBlur	perte de focus de l'élément Input.
onClick	clic de souris sur le bouton.
onFocus	réception de focus sur l'élément Input.
onMouseDown	bouton de la souris enfoncé sur l'élément Input.
onMouseUp	bouton de la souris relaché sur l'élément Input.

Exemple [voir]

```
<html>
 <head>
  <script language="javascript">
   function quel_bouton(btn)
   {
    if (btn = 'premier')
     document.write('Vous avez cliqué sur le bouton
                Soumettre du second formulaire.');
    else
     document.write('Vous avez cliqué sur le bouton
                Annuler du second formulaire.');
   }
  </script>
 </head>
 <body>
  <p>Cet exemple permet de récupérer le clic
        de souris sur l'un des boutons.</p>
  <form name="formulaire">
   <input type="button" name="soumettre" value="soumettre">
   <br>
   <input type="button" name="annuler" value="annuler">
  </form>
  <script language="javascript">
   document.formulaire.soumettre.onclick = function()
   {
    document.write('Vous avez cliqué sur le bouton
                Soumettre du premier formulaire.');
   }
   document.formulaire.annuler.onclick = function()
   {
     document.write('Vous avez cliqué sur le bouton
              Annuler du premier formulaire.');
   }
  </script>
  <p>Ce second exemple effectue la même fonction
          avec un passage d'arguments.</p>
  <form name="formulaire2">
   <input type="submit" name="soumettre"
          onclick="javascript:quel_bouton('premier')"><br>
   <input type="reset" name="annuler"
          onclick="javascript:quel_bouton('second')">
  </form>
 </body>
</html>
```

En savoir plus :

Netscape Microsoft

8.8 / Checkbox

***Checkbox* représente les cases à cocher.**

Compatibilité


Disponibilité

Javascript 1.0 côté client

Correspondance HTML

<input...>

Héritage

Input et HTMLElement

Synopsis

```
form.elements[i]
form.name //une seule cas à cocher. form.name[i] //un groupe de cas à cocher.
```

Propriétés

Propriété	Description
checked	spécifie l'état de la case à cocher.
defaultchecked	la case à cocher est côchée par défaut.
value	valeur de la case à cocher.
form	spécifie le formulaire contenant la case à cocher.
name	correspond au nom de la case ou du groupe de cases à cocher.
type	correspond à l'attribut TYPE.

Méthodes

Méthode	Description
blur	enlève le focus sur la case à cocher.
click	simule un clique de souris sur la case à cocher.
focus	donne le focus à le case à cocher.
handleEvent	invoque le Handler pour spécifier un événement.

Evénements

Evénement	Description
onBlur	perte de focus de l'élément Checkbox.
onClick	clique de souris sur la case à cocher.
onFocus	réception de focus sur l'élément Checkbox.

Exemple [voir]

```
<html>
  <head>
    <script language="JavaScript">
      function controle_choix(i)
      {
        var choix = document.formulaire.cases;
        if (i == 0)
        {
          if (choix[0].checked == true)
          {
            for (i = 1; i < choix.length; i++)
              choix[i].checked = false;
          }
        }
        else
        {
          if (choix[i].checked == true)
          {
            choix[0].checked = false;
          }
        }
      }
    </script>
  </head>
  <body>
    Sélectionner les cases à cocher:
    <form name="formulaire">
      <input type="checkbox" name="cases" value="seule"
             onclick="controle_choix(0)">Case à cocher unique
      <br><br>
      <input type="checkbox" name="cases" value="un"
             onclick="controle_choix(1)">Premièr choix
      <br>
      <input type="checkbox" name="cases" value="deux"
             onclick="controle_choix(2)">Second choix
      <br>
      <input type="checkbox" name="cases" value="trois"
             onclick="controle_choix(3)">Troisième choix
      <br>
      <input type="checkbox" name="cases" value="quatre"
             onclick="controle_choix(4)">Quatrième choix
      <br>
      <input type="checkbox" name="cases" value="cinq"
             onclick="controle_choix(5)">Cinquième choix
      <br>
    </form>
  </body>
</html>
```

Exemple [voir]

```
<html>
 <head>
  <script language="javascript" type="text/javascript">
   function change(code){
    var valeur = document.form.champ.value;
    if(valeur.search(code) != -1){
     valeur = valeur.replace(' '+code+' ', '');
     document.form.champ.value = valeur;
     return false;
    }
    else{
     document.form.champ.value += ' '+code+' ';
     return true;
    }
   }
  </script>
 </head>
 <body>
  <form name="form">
   <input type="text" name="champ" size="50"><br>
   <input type="checkbox" name="cocher" value="1"
        onclick="change('Première valeur');" > Première valeur
   <input type="checkbox" name="cocher" value="2"
        onclick="change('Seconde valeur');"> Seconde valeur
   <input type="checkbox" name="cocher" value="3"
        onclick="change('Troisième valeur');"> Troisième valeur
  </form>
 </body>
</html>
```

En savoir plus :

Netscape Microsoft

8.9 / Date

***Date* permet de manipuler des dates et des heures.**

Compatibilité


Disponibilité

Javascript 1.0

Constructeur

```
Objet_date = new Date()
Objet_date = new Date(dateVal)
Objet_date = new Date(année, mois, jour, heure, minute, seconde, milliseconde)
```

Synopsis

```
form.element[i]
form.name
```

Propriétés

Propriété	Description
constructor	spécifie la fonction qui crée un objet prototype.
property	permet l'ajout des propriétés à l'objet *Date*.

Méthodes

Méthode	Description
getDate	retourne le jour du mois en accord avec le temps local.
getDay	retourne le jour de la semaine en accord avec le temps local.
getFullYear	retoune l'année en accord avec le temps local.
getHours	retourne l'heure.
getMilliseconds	retourne les millisecondes.
getMinutes	retourne les minutes.
getMonth	retourne le mois.
getSeconds	retourne les secondes.
getTime	retourne une date en millisecondes.
getTimezoneOffset	détermine le décalage par rapport à l'heure GMT.
getUTCDate	retoune le jour du mois en accord avec le système horaire universel (UTC).
getUTCDay	retoune le jour de la semaine en accord avec le système horaire universel (UTC).
getUTCFullYear	retoune l'année en accord avec le système horaire universel (UTC).
getUTCHours	retoune l'heure en accord avec le système horaire universel (UTC).
getUTCMilliseconds	retoune les millisecondes en accord avec le système horaire universel (UTC).
getUTCMinutes	retoune les minutes en accord avec le système horaire universel (UTC).
getUTCMonth	retoune le mois en accord avec le système horaire universel (UTC).
getUTCSeconds	retoune les secondes en accord avec le système horaire universel (UTC).
getYear	retourne l'année sur deux caractères.
parse	retourne le nombre de millisecondes d'une chaîne de caractères date.
setDate(jour_du_mois)	détermine le jour du mois.
setFullYear(année)	détermine l'année.
setHours(heures)	détermine l'heure.
setMilliseconds(millisecondes)	détermine les millisecondes.
setMinutes(minutes)	détermine les minutes.
setMonth(mois)	détermine le mois.
setSeconds(seconde)	détermine les secondes.
setTime	détermine un objet date en millisecondes.
setUTCDate	détermine le jour du mois en accord avec le système horaire universel (UTC).
setUTCFullYear	détermine l'année en accord avec le système horaire universel (UTC).
setUTCHours	détermine l'heure en accord avec le système horaire universel (UTC).
setUTCMilliseconds	détermine les millisecondes en accord avec le système horaire universel (UTC).

setUTCMinutes	détermine les minutes en accord avec le système horaire universel (UTC).
setUTCMonth	détermine le mois en accord avec le système horaire universel (UTC).
setUTCSeconds	détermine les secondes en accord avec le système horaire universel (UTC).
setYear	détermine l'année sur deux caractères.
toGMTString	convertit un objet date GMT en une chaîne de caractères.
toLocaleString	convertit un objet date en temps local en une chaîne de caractères.
toSource	retourne un objet littéral représentant l'objet date spécifié.
toString	retourne une chaîne de caractères représentant l'objet date spécifié.
toUTCString	convertit un objet date UTC en une chaîne de caratères
UTC	retourne des millisecondes à partir d'un objet date.
valueOf	convertit un objet Date en nombre.

Exemple [voir]

```
<html>
 <head>
  <script language="JavaScript">
   function systeme_24h()
   {
    if (document.formulaire.choix_systeme[0].checked)
    {
     return true;
    }
    return false;
   }

   function voir_heure(hh)
   {
    if (systeme_24h() || (hh > 0 && hh < 13))
    {
     return (hh);
    }
    if (hh == 0)
    {
     return (12);
    }
    return (hh - 12);
   }

   function voir_zero(secondes)
   {
    if (secondes > 9)
    {
     return "" + secondes;
    }
    return "0" + secondes;
   }

   function voir_ampm()
   {
     if (systeme_24h())
     {
      return ("");
     }
     if (heure.getHours() < 12)
     {
      return (" am");
     }
     return (" pm");
   }

   function affiche_heure()
   {
    heure = new Date;
    document.formulaire.temps.value = voir_heure(heure.getHours())
                    + ":" + voir_zero(heure.getMinutes())
                    + ":" + voir_zero(heure.getSeconds())
                    + voir_ampm()
    setTimeout("affiche_heure()",1000)
   }
  </script>
 </head>
 <body onload="affiche_heure()">
  <form name="formulaire">
   <input type="text" name="temps" size=10>
   <br><br>
   <b><u>Type d'affichage :</u></b><br>
   <input type="radio" name="choix_systeme" checked>
      sur 24 heures<br>
   <input type="radio" name="choix_systeme">
      sur 12 heures
  </form>
 </body>
</html>
```

En savoir plus :

Netscape Microsoft

8.10 / Dictionary

***Dictionary* est un objet conservant des paires clés-éléments de données.**

Compatibilité

Disponibilité

JScript 1

Propriétés

Propriété	Description
Count	renvoie le nombre d'éléments dans une collection ou dans un objet *Dictionary*.
Item(clé)	définit ou renvoie un item pour une key spécifiée d'un objet *Dictionary*.
Key(nouvelle_clé)	Définit une clé (key) dans un objet *Dictionary*.

Méthodes

Méthode	Description
Add(clé, item)	ajoute une paire clé-élément à un objet *Dictionary*.
Exists(clé)	renvoie *true* si l'élément spécifié existe dans l'objet *Dictionary*, ou *false* dans le cas contraire.
Items	renvoie un tableau contenant tous les éléments d'un objet *Dictionary*.
Keys	renvoie un tableau contenant toutes les clés existantes dans un objet *Dictionary*.
Remove(clé)	supprime une paire clé-élément d'un objet *Dictionary*.
RemoveAll	Supprime toutes les paires clé-élément d'un objet *Dictionary*.

Exemples [voir]

```
<html>
 <head>
  <title>Démonstration de Dictionary</title>
  <script language="JScript">
   function AfficheDico()
   {
     var elements, dico, i, liste;
     dico = new ActiveXObject("Scripting.Dictionary");
     dico.Add ("a", "area");
     dico.Add ("b", "body");
     dico.Add ("c", "center");
     dico.Add ("d", "div");
     dico.Add ("e", "em");
     // Obtenir les éléments.
     elements = (new VBArray(dico.Items())).toArray();
     liste = "";
     for (i in elements) //Parcourir le dictionnaire.
     {
       liste += elements[i] + "<br>";
     }
     return(liste);
   }
  </script>
 </head>
 <body>
  <a href="javascript:AfficheDico()">Liste des mots du dictionnaire</a>
 </body>
</html>
```

En savoir plus :

Microsoft

8.11 / Document

***Document* correspond à un document HTML.**

Compatibilité


Disponibilité

Javascript 1.0

Correspondance HTML

<html>...</html>

Héritage

HTMLElement

Synopsis

window.document
document

Propriétés

Propriété	Description
alinkColor	couleurs des liens activés (ALINK).
anchors[]	tableau des ancres contenues dans le document.
applets[]	applets java contenu dans le document
bgColor	couleur de fond du document (BGCOLOR).
classes	définit des classes de style CSS (CLASS).
cookie	spécifie un cookie.
domain	spécifie le nom de domaine du serveur.
embeds[]	tableau des objets imbriqués dans le document.
fgColor	couleur du texte dans le document (TEXT).
formName	représente le nom du formaulaire dans le document.
forms[]	tableau des formulaires contenus dans le document.
height	hauteur du document en pixels.
ids	associe un style à un élément HTML unique (ID).
images[]	tableau des images contenues dans le document.
location	objet contenant des informations à propos de l'adresse URL courante.
lastModified	chaîne de caractères représentant la date de dernière modification.
layers[]	tableau des couches (layer) contenues dans le document.
linkColor	couleur des liens dans le document (LINK).
links[]	tableau des liens contenus dans le document.
plugins[]	tableau des plug-ins contenus dans le document.
referrer	URL du document d'où provient le document courant.
tags.Nom_balise	crée un objet style associé aux balises HTML.
title	représente le titre du document (TITLE).
URL	chaîne de caractères spécifiant l'URL complète du document.
vlinkColor	couleur des liens visités (VLINK).
width	largeur du document en pixels.

Méthodes

Méthode	Description
captureEvents(event.TYPE)	spécifie le type d'événements à capturer (NE4).
clear()	efface un document.
close()	ferme un canal de sortie.
contextual(contextN)	définit un style CSS contextuel.
elementFromPoint(x,y)	indique l'élément se trouvant aux coordonnées x et y.
getElementById	retourne le premier élément d'une collection.
getSelection	retourne une chaîne de caractères contenant le texte de la sélection courante.
handleEvent(événement)	invoque l'Handler d'un événement spécifié.
open("type_mime", "replace")	ouvre un canal collectant le résultat des méthodes *write* ou *writeln*.
releaseEvents(event.TYPE)	cesse la capture des événements.
routeEvent(événement)	transmet l'événement capturé aux gestionnaire d'événements
write("expression")	écrit une ou plusieurs expressions HTML dans un document.
writeln("expression")	écrit une ou plusieurs expressions HTML avec un retour à la ligne dans un document.

Evénements

Evénement	Description
onClick	clic sur le bouton de la souris.
onDblClick	double-clic sur le bouton de la souris
onKeyDown	touche du clavier enfoncée.
onKeyPress	la touche du clavier enfoncée est relâchée.
onKeyUp	touche du clavier relâchée.
onMouseDown	bouton de la souris enfoncé sur l'élément Input.
onMouseUp	bouton de la souris relaché sur l'élément Input.

Exemple [voir]

```
<html>
 <head>
  <title>Démonstration de Document</title>
  <script>
   function ouverture()
   {
    var fenetre = window.open('','','toolbar="yes",scrollbars="yes",
                  width=320,height=240');
    var texte = "Une nouvelle fenêtre s'ouvre !";
    fenetre.document.open("text/html","replace");
    fenetre.document.write("<p>" + texte + "<p>");
    fenetre.document.close();
   }
  </script>
 </head>
 <body onload="ouverture()">
  Cette fenêtre va ouvrir automatiquement une autre fenêtre.
 </body>
</html>
```

En savoir plus :

Netscape Microsoft

8.12 / Drive

Drive **permet l'accès aux propriétés d'un lecteur de disque particulier ou d'un élément de réseau partagé.**

Compatibilité

Disponibilité

JScript 1

Propriétés

Propriété	Description
AvailableSpace	retourne l'espace disponible d'un lecteur.
DriveLetter	retourne la lettre symbolisant le lecteur.
DriveType	retourne le type du disque dur ou du lecteur réseau.
FileSystem	retourne le type du système de fichier du lecteur.
FreeSpace	retourne l'espace libre disponible sur un lecteur pour un utilisateur d'un lecteur.
IsReady	retourne *true* si le lecteur est prêt sinon *false*.
Path	retourne le chemin d'accès du lecteur.
RootFolder	retourne un objet *Folder* représentant le dossier racine.
SerialNumber	renvoie le numéro de série du lecteur.
ShareName	retourne le nom de partage réseau d'un lecteur.
TotalSize	retourne la taille totale d'un lecteur.
VolumeName	retourne le nom de volume d'un lecteur.

Exemples [voir]

```
<html>
 <head>
   <script language="JScript">
    function ShowFreeSpace(drvPath)
    {
     var fso, d, s;
     fso = new ActiveXObject("Scripting.FileSystemObject");
     d = fso.GetDrive(fso.GetDriveName(drvPath));
     s = "Lecteur  + drvPath + " - " ;
     s += d.VolumeName + "<br>";
     s += "Espace disponible : " + d.FreeSpace/1024 + " Koctets";
     return(s);
    }
   </script>
  </head>
  <body onload="javascript:GetAppVersion()">
  </body>
</html>
```

Référence :

MICROSOFT

8.13 / Drives

***Drives* représente une collection en lecture seule de tous les lecteurs disponibles.**

Compatibilité

Disponibilité

JScript 1

Propriétés

Propriété	Description
Count	retourne le nombre d'éléments dans une collection ou dans un objet *Dictionary*.
Item(clé) [= item]	définit ou retourne un item pour une clé spécifiée d'un objet *Dictionary*.

Exemples [voir]

```
<html>
 <head>
  <title>Démonstration de Drives</title>
  <script language="JScript">
   function ListeUnite()
   {
    var fso, info = "", nom, liste, lecteurs;
    fso = new ActiveXObject("Scripting.FileSystemObject");
    liste = new Enumerator(fso.Drives);
    for (; !liste.atEnd(); liste.moveNext())
    {
     lecteurs = liste.item();
     info = info + lecteurs.DriveLetter;
     info += " - ";
     if (lecteurs.DriveType == 3)
      nom = lecteurs.ShareName;
     else if (lecteurs.IsReady)
      nom = lecteurs.VolumeName;
     else
      nom = "[Lecteur non prêt]";
     info +=   nom + "<br>";
    }
    return(info);
   }
  </script>
 </head>
 <body>
  <a href="javascript:ListeUnite()">Liste des lecteurs</a>
 </body>
</html>
```

Référence :

Microsoft

8.14 / Enumerator

***Enumerator* permet d'énumérer les éléments contenus dans une collection.**

Compatibilité

Disponibilité

JScript 3.0

Constructeur

Objet_enumerator = new Enumerator(collection)

Méthodes

Méthode	Description
atEnd()	renvoie une valeur booléenne indiquant si un objet Enumerator se trouve à la fin d'une collection.
item()	renvoie l'élément en cours dans une collection.
moveFirst()	place l'élément en cours dans une collection en première position.
moveNext()	déplace l'élément en cours vers un autre élément dans une collection.

Exemple [voir]

```
<html>
 <head>
  <script language="jscript">
   function Liste_DD()
   {
    var systeme_fichier;
    var liste_unite;
    var nom_partage;
    var obj_enum;
    var element_cours;
    systeme_fichier = new ActiveXObject("Scripting.FileSystemObject");
    //Créer Enumerator sur les unités.
    obj_enum = new Enumerator(systeme_fichier.Drives);
    liste_unite = "";
    //Énumérer la collection d'unités.
    for (;!obj_enum.atEnd();obj_enum.moveNext())
    {
     element_cours = obj_enum.item();
     liste_unite = liste_unite + element_cours.DriveLetter;
     liste_unite += "<br>";
     //Déterminer s'il s'agit d'une unité de réseau.
     if (element_cours.DriveType == 3)
      //Obtenir le nom de partage.
      nom_partage = element_cours.ShareName;
     //Déterminer si l'unité est prête.
     else if (element_cours.IsReady)
      //Obtenir le nom du volume.
      nom_partage = element_cours.VolumeName;
     else
      nom_partage = "[Unité non prête]";
     liste_unite += " - " + nom_partage + "<br>";
    }
    //Renvoyer le statut de l'unité.
    document.write("Liste des unités actives : " + s);
   }
  </script>
 </head>
 <body onload="liste_DD()">
 </body>
</html>
```

En savoir plus :

Microsoft

8.15 / Error

***Error* contient des informations sur les erreurs.**

Compatibilité

Disponibilité

Jscript 5

Constructeur

```
Objet_erreur = new Error()
Objet_erreur = new Error(nombre)
Objet_erreur = new Error(nombre, description)
```

Propriétés

Propriété	Description
description	renvoie ou définit la chaîne de description associée à une erreur spécifique.
number	renvoie ou définit la valeur numérique associée à une erreur spécifique. La propriété par défaut de l'objet Error est number.

Exemple

```
<html>
  <body>
    <script language=Javascript>
      try { x = y; }
      catch(erreur)
      {
        document.write("Opération effectuée : x = y;");
        document.write("<br>Erreur détectée : " + erreur);
        document.write("<br>numéro d'erreur : " + (erreur.number & 0xFFFF));
        document.write("<br>Description de l'erreur : " + erreur.description);
      }
    </script>
  </body>
</html>
```

En savoir plus :

Microsoft

8.16 / Event

***Event* représente un événement sur un objet.**

Compatibilité


Disponibilité

Javascript 1.2

Synopsis

```
//argument du gestionnaire d'événement sous Netscape.
function handler(event) {...}

//propriété de l'objet window avec Explorer.
window.event
```

Propriétés

Pour Netscape :	
data	retourne un tableau de chaînes de caractères contenant les URL des objets déposés avec l'événement Drag and Drop.
height	représente la hauteur de la fenêtre ou du cadre.
layerX	représente la coordonnée X au sein d'un calque (Layer).
layerY	représente la coordonnée Y au sein d'un calque (Layer).
modifiers	chaîne de caractères spécifiant la modification de clés associée à un événement de souris ou de clavier. Les valeurs sont : ALT_MASK, CONTROL_MASK, SHIFT_MASK, and META_MASK.
pageX	nombre spécifiant la position horizontale du curseur en pixels par rapport à la page.
pageY	nombre spécifiant la position verticale du curseur en pixels par rapport à la page.
target	chaîne de caractères représentant l'objet sur lequel l'événement a eu lieu.
which	nombre spécifiant quel bouton de souris a été pressé ou la valeur ASCII d'une touche de clavier pressée.
width	représente la largeur de la fenêtre ou du cadre.

Pour Explorer :	
altKey	détermine si la touche Alt était pressée lors de l'événement.
button	détermine le bouton de la souris pressée lors de l'événement.
cancelBubble	arrête la propagation de l'événement.
clientX	nombre spécifiant la position horizontale du curseur en pixels par rapport à la page.
clientY	nombre spécifiant la position verticale du curseur en pixels par rapport à la page.
ctrlKey	détermine si la touche Ctrl était pressée lors de l'événement.
fromElement	représente l'objet source d'un déplacement de la souris.
keyCode	détermine l'encodage Unicode de la touche pressée.
offsetX	coordonnée horizontale de l'événement en position relative par rapport au conteneur.
offsetY	coordonnée verticale de l'événement en position relative par rapport au conteneur.
reason	représente l'état d'un transfert de données.
returnValue	indique la valeur de retour pour le gestionnaire d'événement.
shiftKey	détermine si la touche Shift était pressée lors de l'événement.
srcElement	indique l'objet sur lequel l'événement s'est produit.
srcFilter	indique le filtre qui a changé.
toElement	indique l'objet de destination d'un déplacement de souris.

Commun :	
screenX	nombre spécifiant la position horizontale du curseur en pixels par rapport à l'écran.
screenY	nombre spécifiant la position verticale du curseur en pixels par rapport à l'écran.
type	chaîne de caratères représentant le type de l'événement.
x	identique à *layerX*.
y	identique à *layerY*.

Exemple [voir]

```
<html>
 <head>
  <script language=Javascript>
   function clavier_ie() //pour Explorer
   {
    document.test.zone_clavier.value = "Code Touche= "
            + event.keyCode + ' Carac.= '
            + String.fromCharCode(event.keyCode);
   }

   function souris_ie()//pour Explorer
   {
    var val="";
    document.test.zone_souris.value = event.type
                    + ' ' + event.x
                    + '  ' + event.y;
   }

   function clavier(e) //pour Netscape
   {
    document.test.zone_clavier.value = "Code ASCII= "
                + e.which + ' Carac.= '
                + String.fromCharCode(e.which);
   }

   function souris(e) //pour Netscape
   {
    document.test.zone_souris.value = e.type
                    + '  ' + e.x
                    + '  ' + e.y;
   }

   document.onKeyPress = clavier; // pour Netscape
   document.onClick = souris ; // pour Netscape
  </script>
 </head>
 <body
        onKeyDown="clavier_ie()"
        onMouseDown="souris_ie()"
        onMouseMove="souris_ie()">
  <form name="test">
   <input type="text" name="zone_souris" size=20><br>
   <input type="text" name="zone_clavier" size=30>
  </form>
 </body>
</html>
```

En savoir plus :

Netscape Microsoft

8.17 / File

***File* procure un accès à toutes les propriétés d'un fichier.**

Compatibilité

Disponibilité

JScript 1

Propriétés

Propriété	Description
Attributes [= nombre]	définit ou renvoie les attributs des fichiers ou des dossiers.
DateCreated	retourne la date et l'heure de création du fichier.
DateLastAccessed	retourne la date et l'heure du dernier accès sur le fichier.
DateLastModified	retourne la date et l'heure de la dernière modification du fichier.
Drive	retourne la lettre de l'unité contenant le fichier.
Name [= nom]	définit ou retourne le nom d'un fichier ou d'un dossier.
ParentFolder	retourne l'objet dossier du parent pour le fichier.
Path	retourne le chemin d'accès du fichier.
ShortName	retourne un nom de fichier dans le format Dos : 8.3.
ShortPath	retourne le chemin dans le format Dos.
Size	retourne la taille du fichier.
Type	retourne le type du fichier.

Méthodes

Méthode	Description
Copy(destination, mode)	permet de copier un fichier d'un emplacement à un autre.
Delete	efface un fichier.
Move(destination)	déplace un fichier à un destination indiquée.
OpenAsTextStream(mode, format)	ouvre un fichier spécifié et renvoie un objet *TextStream* qui peut être utilisé pour manipuler un fichier.

Exemples [voir]

```
<html>
 <head>
  <title>Démonstration de File</title>
  <script language="JScript">
   function InfoFichier(chemin)
   {
    var fso, fichier, info;
    fso = new ActiveXObject("Scripting.FileSystemObject");
    fichier = fso.GetFile(chemin);
    info = fichier.Name + " sur le lecteur " + fichier.Drive + "<br>";
    info += "Créé le : " + fichier.DateCreated + "<br>";
    info += "Dernier accès le : " + fichier.DateLastAccessed + "<br>";
    info += "Dernière modification le : " + fichier.DateLastModified;
    return(info);
   }
  </script>
 </head>
 <body>
  <a href="javascript:InfoFichier('c:\\autoexec.bat')">
   Informations sur le fichier Autoexec.bat
  </a>
 </body>
</html>
```

Référence :

Microsoft

8.18 / Files

***Files* représente une collection de tous les objets File à l'intérieur d'un dossier.**

Compatibilité

Disponibilité

JScript 1

Propriétés

Propriété	Description
Count	retourne le nombre d'éléments dans une collection ou dans un objet *Dictionary*.
Item(clé) [= item]	définit ou retourne un item pour une clé spécifiée d'un objet *Dictionary*.

Exemples [voir]

```
<html>
 <head>
  <title>Démonstration de Files</title>
  <script language="jscript">
   function ListeFichier(rep)
   {
    var fso, fichier, liste, info="";
    fso = new ActiveXObject("Scripting.FileSystemObject");
    fichier = fso.GetFolder(rep);
    liste = new Enumerator(fichier.files);
    for (; !liste.atEnd(); liste.moveNext())
    {
     info += liste.item();
     info += "<br>";
    }
    return(info);
   }
  </script>
 </head>
 <body>
  <a href="javascript:ListeFichier('c:&&')">Liste des fichiers sur C:&</a>
 </body>
</html>
```

Référence :

Microsoft

8.19 / FileSystemObject

FileSystemObject **donne accès au système de fichiers d'un serveur.**

Compatibilité

Disponibilité

JScript

Constructeur

```
y = new ActiveXObject("Scripting.FileSystemObject")
```

Propriétés

Propriété	Description
Drives	renvoie une collection Drives composée de tous les objets Drive disponibles sur la machine cliente.

Méthodes

Méthode	Description
BuildPath(chemin, nom)	ajoute un nom à un chemin existant.
CopyFile(source, destination, bool_ecrase)	copie un ou plusieurs fichiers d'un emplacement à un autre.
CopyFolder(source, destination, bool_ecrase)	copie de façon récursive un dossier d'un emplacement à un autre.
CreateFolder(nom_dossier)	crée un dossier.
CreateTextFile(fichier, bool_ecrase, unicode_ASCII)	crée le nom d'un fichier spécifié et renvoie un objet TextStream qui peut être utilisé pour lire ou écrire dans le fichier.
DeleteFile(fichier_supprimer, bool_forçage)	supprime un fichier spécifié.
DeleteFolder(dossier_supprimer, bool_forçage)	supprime un dossier spécifié et son contenu.
DriveExists(lecteur)	renvoie True si le lecteur spécifié existe et False dans le cas contraire.
FileExists(fichier)	renvoie True si un fichier spécifié existe et False dans le cas contraire.
FolderExists(dossier)	renvoie True si un dossier spécifié existe et False dans le cas contraire.
GetAbsolutePathName(chemin)	renvoie un chemin d'accès complet et sans ambiguïté à partir d'une spécification de chemin fournie.
GetBaseName(chemin)	renvoie une chaîne contenant le nom de base du dernier composant d'un chemin d'accès, sans extension de fichier.
GetDrive(lecteur)	Renvoie un objet Drive correspondant au lecteur spécifié dans un chemin d'accès.
GetDriveName(chemin)	renvoie une chaîne contenant le nom correspondant au lecteur spécifié dans un chemin d'accès.
GetExtensionName(chemin)	Renvoie une chaîne contenant le nom d'extension du dernier composant d'un chemin d'accès.

GetFile(fichier)	renvoie un objet File correspondant à un nom de fichier situé dans un chemin d'accès spécifié.
GetFileName(chemin)	renvoie le dernier composant du chemin spécifié qui ne fait pas partie de la spécification du lecteur.
GetFolder(chemin)	renvoie un objet Folder qui correspond à un dossier spécifié dans un chemin d'accès.
GetParentFolderName(chemin)	renvoie une chaîne contenant le nom du dossier parent du dernier composant d'un chemin spécifié.
GetSpecialFolder(dossier)	Renvoie l'objet dossier spécial indiqué.
GetTempName()	renvoie un nom de dossier ou de fichier temporaire généré de façon aléatoire, utile pour les opérations qui requièrent ce genre de dossiers ou de fichiers.
MoveFile(source, destination)	déplace un ou plusieurs fichiers d'un emplacement à un autre.
MoveFolder(source, destination)	déplace un ou plusieurs dossiers d'un endroit à un autre.
OpenTextFile(fichier, iomode, bool_création, format)	Ouvre un fichier spécifié et renvoie un objet TextStream qui peut être utilisé pour lire, écrire et effectuer des ajouts au fichier.

Exemple [voir]

```
<html>
 <head>
  <title>Démonstration de FileSystemObject</title>
  <script language="jscript">
   function effacer_fichier(chemin)
   {
    var fso;
    fso = new ActiveXObject('Scripting.FileSystemObject');
    fso.DeleteFile(chemin);
   }
   function Voir_info(chemin)
   {
    var fso;
    var fichier = "";
    var caracteristique = "";
    fso = new ActiveXObject('Scripting.FileSystemObject');
    fichier = fso.CreateTextFile("c:&&texte.txt", true);
    fichier = fso.GetFile(chemin);
    caracteristique = fichier.Path.toUpperCase() + "<br>";
    caracteristique += "Créé le : " + fichier.DateCreated + "<br>";
    caracteristique += "Dernier accès le : "
                + fichier.DateLastAccessed + "<br>";
    caracteristique += "Dernière modification le : "
                + fichier.DateLastModified
    document.write(caracteristique);
   }
  </script>
 </head>
 <body>
  <b class="soustitre">Voir les caractéristique
      du fichier : c:&texte.txt...</b><br>
  <i>Si ce fichier n'est pas créé sous la racine de C:&,
      créez-le afin que ce script puisse fonctionner.</i>
  <br><br>
  <a href="javascript:Voir_info('c:&&texte.txt')"><b>
   Caractéristiques
  </b></a><br>
  <a href="javascript:effacer_fichier('c:&&texte.txt')"><b>
   Supprimer ce fichier
  </b></a>
 </body>
</html>
```

En savoir plus :

Microsoft

8.20 / FileUpload

FileUpload **représente un champ de saisie pour un téléchargement de fichier au sein d'un formulaire.**

Compatibilité


Disponibilité

Javascript 1.0 côté client

Correspondance HTML

```
<input type="file"...>
```

Héritage

Input et HTMLElement

Synopsis

```
form.name
form.element[i]
```

Propriétés

Propriété	Description
form	spécifie le formulaire contenant le bouton.
name	correspond à l'attribut NAME de INPUT.
type	correspond à l'attribut TYPE de INPUT.
value	affiche un texte sur le bouton

Méthodes

Méthode	Description
blur	enlève le focus sur l'élément Input.
focus	donne le focus à l'élément Input.
handleEvent	invoque le Handler pour spécifier un événement.
select	simule la sélection de l'élément Input.

Evénements

Evénement	Description
onBlur	perte de focus de l'élément Input.
onChange	modification de la valeur du champs.
onFocus	réception de focus sur l'élément Input.

Exemple [voir]

```
<html>
 <head>
  <script language="Javascript">
   tableau = new Array(".htm", ".html", ".txt", ".gif", ".jpg", ".png");
   function verification(formulaire, fichier)
   {
     soumission = false;
     if (!fichier) return;
       while (fichier.indexOf("&&") != -1)
         fichier = fichier.slice(fichier.indexOf("&&") + 1);
     ext = fichier.slice(fichier.indexOf(".")).toLowerCase();
     for (var i = 0; i < tableau.length; i++)
     {
       if (tableau[i] == ext)
       {
         soumission = true; break;
       }
     }
     if (soumission) formulaire.submit();
     else
       alert("Vous ne pouvez uploader que ces types de fichiers : "
           + "(" + tableau.join("  ") + ")");
   }
  </script>
 </head>
 <body>
  Sélectionner des documents webs :<br>
  <script>
   document.write("(" + tableau.join("  ") + ")");
  </script>
  <form name="formulaire">
   <input type="file" name="telechargement"><br>
   <input type="button" name="soumission" value="soumettre"
              onclick="verification(this.form, this.form.telechargement.value)">
  </form>
 </body>
</html>
```

En savoir plus :

Netscape Microsoft

8.21 / Folder

***Folder* autorise l'accès à toutes les propriétés d'un dossier.**

Compatibilité

Disponibilité

JScript 1

Propriétés

Propriété	Description
Attributes [= attribut]	définit ou retourne les attributs des dossiers.
DateCreated	retourne la date et l'heure de création du dossier.
DateLastAccessed	retourne la date et l'heure du dernier accès au dossier.
DateLastModified	retourne la date et l'heure de la dernière modification du dossier.
Drive	retourne la lettre désignant le lecteur sur lequel réside le dossier.
Files	retourne un collection de fichier contenu dans le dossier.
IsRootFolder	retourne *true* si le dossier est le dossier racine.
Name [= nom]	définit ou retourne le nom du dossier.
ParentFolder	retourne le dossier parent du dossier spécifié.
Path	retourne le chemin d'accès du dossier.
ShortName	retourne le nom du dossier dans le format DOS 8.3.
ShortPath	retourne le chemin du dossier dans le format DOS 8.3.
Size	retourne la taille du dossier.
SubFolders	retourne une collection de dossiers contenus dans le dossier spécifié.
Type	retourne le type du dossier.

Méthodes

Méthode	Description
Copy(destination, mode)	permet de copier un dossier d'un emplacement à un autre.
Delete	efface un dossier.
Move(destination)	déplace un dossier à un destination indiquée.
OpenAsTextStream(mode, format)	ouvre un dossier spécifié et renvoie un objet *TextStream* qui peut être utilisé pour manipuler un dossier.

Exemples [voir]

```
<html>
 <head>
  <title>Démonstration de Folder</title>
   <script language="jscript">
    function CreeRepertoire()
    {
     var fso, rep;
     fso = new ActiveXObject("Scripting.FileSystemObject");
     rep = fso.CreateFolder("C:&&Repertoire");
     document.write("<html><head><script language=&"jscript&">&n"
      + "function supprime()&n"
      + "{&n"
      + "  var fso;&n"
      + "  fso = new ActiveXObject(&"Scripting.FileSystemObject&");&n"
      + "  fso.DeleteFolder (&"C:&&&&Repertoire&");&n"
      + "  alert(&"Le dossier Repertoire a été supprimé&");&n"
      + "}&n"
      + "<&/script><&/head><body>&n");
     document.write("Le répertoire " + rep.Name
              + " a été créé sur le disque dur C:");
     document.write("<br><a href=&"javascript:supprime()&">"
              + "Supprimer le répertoire</a>");
     document.write("<&/body><&/html>");
    }
  </script>
 </head>
 <body>
  <a href="javascript:CreeRepertoire()">Créer un Répertoire</a>
 </body>
</html>
```

Référence :

Microsoft

8.22 / Folders

***Folders* représente une collection de tous les objets File à l'intérieur d'un dossier.**

Compatibilité

Disponibilité

JScript 1

Propriétés

Propriété	Description
Count	retourne le nombre d'éléments dans une collection ou dans un objet *Dictionary*.
Item(clé) [= item]	définit ou retourne un item pour une clé spécifiée d'un objet *Dictionary*.

Propriétés

Propriété	Description
Add(clé, item)	ajoute une paire clé-item à un objet *Dictionary*.

Exemples [voir]

```
<html>
  <head>
  <title>Démonstration de Files</title>
    <script language="jscript">
      function ListeRep(rep)
      {
        var fso, nomRep, liste, info="";
        fso = new ActiveXObject("Scripting.FileSystemObject");
        nomRep = fso.GetFolder(rep);
        liste = new Enumerator(nomRep.SubFolders);
        for (; !liste.atEnd(); liste.moveNext())
        {
          info += fc.item();
          info += "<br>";
        }
        return(s);
      }
    </script>
  </head>
  <body>
    <a href="javascript:ListeRep('c:&&')">
      Liste des répertoires sur C:&
    </a>
  </body>
</html>
```

Référence :

Microsoft

8.23 / Form

***Form* représente un formulaire de saisie.**

Compatibilité


Disponibilité

Javascript 1.0 côté client

Correspondance HTML

<form>...</form>

Héritage

HTMLElement

Synopsis

document.nom_formulaire
document.forms[i]

Propriétés

Propriété	Description
action	représente l'URL de soumission (ACTION).
elements[]	tableau représentant tous les éléments dans le formulaire.
encoding	représente l'encodage du formulaire (ENCTYPE).
length	représente le nombre d'éléments dans un formulaire.
method	représente la méthode de soumission du formulaire (METHOD).
name	spécifie le nom du formulaire (NAME).
target	indique la fenêtre cible du résultat du formulaire (TARGET).

Méthodes

Méthode	Description
handleEvent	invoque l'Handler pour l'événement spécifié.
reset	simules un clic de souris sur un bouton d'annulation.
submit	simules un clic de souris sur un bouton de soumission.

Evénements

Evénement	Description
onreset	invoqué lorsque le formulaire est réinitialisé.
onsubmit	invoqué lorsque le formulaire est soumis.

Exemple [voir]

```
<html>
 <head>
  <script language="javascript">
   function controle()
   {
    var correct = true
    if (document.formulaire.nom.value == "")
    {
     correct = false;
     alert("Saisissez votre nom !")
    }
    if (document.forms[0].sexe.checked == "")
    {
     correct = false;
     alert("Choisissez votre sexe !")
    }
    if (document.formulaire.email.value == "")
    {
     correct = false;
     alert("Entrez votre adresse eMail !")
    }
    if (correct)
    {
     alert("Merci de votre patience !&nLe formulaire est maintenant soumis.")
    }
    document.formulaire.submit();
   }
  </script>
 </head>
 <body>
  <form name="formulaire"><br>
   Votre nom : <input name="nom" size="25"><br>
   <input name="sexe" type="radio" value="homme">Homme<br>
   <input name="sexe" type="radio" value="femme">Femme<br>
   eMail : <input name="email" size=15><br>
   <input type="button" value="Soumettre" onclick="controle()">
   <input type="reset" value="Annuler">
  </form>
 </body>
</html>
```

En savoir plus :

Netscape Microsoft

8.24 / Frame

***Frame* représente un cadre dans une fenêtre du navigateur.**

Compatibilité


Disponibilité

Javascript 1.0 côté client

Correspondance HTML

<frame...>...</frame>

Synopsis

window.frames[i]
window.frames.length

On fournit une convenance à l'objet d'Encadrement (frame) pour la référence aux objets qui constituent des cadres. Cependant, JavaScript représente en réalité un cadre employant un objet de fenêtre. Chaque objet Frame est un objet de fenêtre et a toutes les méthodes et les propriétés d'un objet Window. Cependant, une fenêtre qui est un cadre diffère légèrement d'une fenêtre au plus haut niveau.

Propriétés

Propriété	Description
closed	spécifie si une fenêtre a été fermée.
crypto	un objet qui permet les particularités de chiffrage du Navigateur d'accès.
defaultStatus	représente l'affichage par défaut d'un message dans la barre de statut du navigateur.
document	contient l'information sur le document actuel et fournit des méthodes pour l'affichage de la production de HTML à l'utilisateur.
frames[]	représente un tableau de toutes les cadres d'une fenêtre.
history	contient l'information sur les adresses URL que les clients ont visité à l'intérieur d'une fenêtre.
innerHeight	spécifie la dimension verticale de la zone de contenu d'une fenêtre en pixels.
innerWidth	spécifie la dimension horizontale de la zone de contenu d'une fenêtre en pixels.
length	indique le nombre de cadres dans une fenêtre.
location	contient l'information sur l'adresse URL courante.
locationbar	représente la barre d'affichage d'adresse sur le navigateur.
menubar	représente la barre de menu du navigateur.
name	spécifie un nom unique pour la fenêtre.
offscreenBuffering	spécifie si les mises à jour de la fenêtre sont placées en mémoire tampon.
opener	spécifie le nom de fenêtre du document appelant quand une fenêtre est ouverte en employant la méthode open.
outerHeight	spécifie la dimension verticale en pixels, de la bordure extérieure de la fenêtre.
outerWidth	spécifie la dimension horizontale en pixels, de la bordure extérieure de la fenêtre.
pageXOffset	fournit la position x actuelle, en pixels, de la page vue d'une fenêtre.
pageYOffset	fournit la position y actuelle, en pixels, de la page vue d'une fenêtre.
parent	représente le parent du cadre.
personalbar	repérésente la barre personnelle du navigateur.
screenX	spécifie la coordonnée x du bord gauche d'une fenêtre.
screenY	spécifie la coordonnée y du bord supérieure d'une fenêtre.
scrollbars	représente la barre de défilement du navigateur.
self	est un synonyme pour la fenêtre courante.
status	spécifie une priorité ou le message temporaire s'affiche dans la barre d'état de la fenêtre.
statusbar	représente la barre de statut du navigateur.
toolbar	représente la barre d'outils du navigateur.
top	représente la fenêtre associée au cadre.
window	est un synonyme pour la fenêtre courante.

Méthodes

Méthode	Description

alert(texte)	affiche un message dans une boîte de dialogue contenant un bouton OK.
atob(Chaîne_de_données)	décode une chaîne de données qui a été encodées en employant la base 64 d'encodage.
back()	défait la dernière étape de l'historique dans n'importe quel cadre à l'intérieur d'une fenêtre de plus haut niveau.
blur()	enlève le focus sur l'élément window.
btoa(chaîne_de_caractère)	crée une base d'encodage sur 64 bits.
captureEvents(event.TYPE)	permet de capturer tous les types d'événements spécifiés produit dans la fenêtre ou le document.
clearInterval(temps)	stoppe une minuterie qui était lancé par la méthode *setInterval*.
clearTimeout(temps)	stoppe une minuterie qui était lancé par la méthode *setTimeout*.
close()	ferme la fenêtre spécifiée.
confirm(texte)	affiche une boîte de dialogue de confirmation contenant les boutons OK et Cancel.
crypto.random(nombre)	retourne une chaîne de caractères pseudo-aléatoire dont la longueur est le nombre indiqué d'octets.
crypto.signText(texte, ask/auto, autorité)	retourne une chaîne de données encodées qui représente un objet signé.
disableExternalCapture()	met hors de service l'événement externe capturé lancé par la méthode *enableExternalCapture*.
enableExternalCapture()	permet à une fenêtre avec cadre à capturer des événements dans des pages chargées à partir de différents serveurs.
find(texte, bool_cas_sensible, bool_recherche_arrière)	permet de trouver la chaîne de caractère spécifiée dans le contenu de la fenêtre indiquée.
focus()	donne le focus à l'objet spécifié.
forward()	charge la prochaîne adresse URL dans le tableau de l'historique.
handleEvent(événement)	invoque le Handler pour spécifier un événement.
home()	dirige l'URL indiqué dans des favoris comme page d'accueil du navigateur.
moveBy(horizontal, vertical)	déplace la fenêtre à partir d'une valeur spécifiée.
moveTo(x, y)	déplace le coin supérieur gauche de la fenêtre aux coordonnées d'écran spécifiées.
open(URL, Nom, séparateur)	ouvre une nouvelle fenêtre du navigateur.
print()	imprime le contenu d'une fenêtre ou d'un cadre.
prompt(texte, valeur_par_défaut)	affiche une boîte de dialogue avec un message et une zone de saisie.
releaseEvents(envent.TYPE)	cesse la capture du type d'événements spécifié.
resizeBy(horizontal, vertical)	redimensionne une fenêtre entière par le déplacement du coin inférieur droit de la fenêtre par une valeur spécifiée.
resizeTo(largeur, hauteur)	redimensionne une fenêtre entière à la hauteur et la largeur extérieures spécifiées.

routeEvent(événement)	transmet un événement capturé au prochaine gestionnaire.
scroll	permet de faire défiler une fenêtre à partir des coordonnées spécifiées.
scrollBy(horizontal, vertical)	permet de faire défiler une zone d'une fenêtre par une valeur spécifiée.
scrollTo(x, y)	permet de faire défiler une zone visible de la fenêtre par des coordonnées spécifiées, tel que le point spécifié devienne le coin supérieur gauche.
setHotKeys(true/false)	active ou désactive les raccourcis claviers dans une fenêtre.
setInterval(fonction, temps, argumentN)	évalue une expression ou appelle une fonction à intervalle régulier spécifié en millisecondes.
setResizable(true/false)	Indique si on permet à un utilisateur de redimensionner une fenêtre.
setTimeout(fonction, temps, argumentN)	évalue une expression ou appelle une fonction une fois que le temps spécifié en millisecondes se soit écoulé.
setZOptions(option)	contrôle l'empilement de la fenêtre.
stop()	stoppe le téléchargement courant.

Evénements

Evénement	Description
onBlur	la fenêtre perd son statut d'élément actif.
onDragDrop	des éléments sont déposés dans la fenêtre.
onError	erreur au cours du chargement d'une image
onFocus	la fenêtre devient active.
onLoad	le hargement de la fenêtre est opéré.
onMove	déplacement de la fenêtre
onResize	redimensionnement de la fenêtre.
onLoad	chargement de l'image.

Exemple [voir]

```
<html>
  <head>
    <title>Cadre de Frames</title>
  </head>
  <frameset rows="60,*">
    <frame name="haut" src="frame_haut_ex.html">
    <frame name="bas" src="frame_bas_ex.html">
  </frameset>
</html>
```

```
<html>
  <head>
    <title>Frame du Haut</title>
    <script language="Javascript">
      var cadre = "bas";
      function ouverture()
      {
        var index = document.forms[0].elements[0].options.selectedIndex;
        if (index > 0)
        {
          var page = document.forms[0].elements[0].options[index].value;
          parent.frames[cadre].location.href = page;
        }
        return true;
      }
    </script>
  </head>
  <body>
    <form>
      <select name="choisir" onchange="javascript:ouverture()">
        <option>Choisir un site</option>
        <option value="http://developer.netscape.com/">Netscape</option>
        <optionvalue="http://msdn.microsoft.com/>Microsoft</option>
      </select>
    </form>
  </body>
</html>
```

```
<html>
  <head>
    <title>Frame du Bas</title>
  </head>
  <body>
    Sélectionner un site ci-dessus...
  </body>
</html>
```

En savoir plus :

Netscape Microsoft

8.25 / Function

Function **représente un champ de saisie pour un téléchargement de fichier au sein d'un formulaire.**

Compatibilité


Disponibilité

Javascript 1.0

Constructeur

```
Nom_fonction = new Function( [parametre1,..., parametreN, instructions);
```

Synopsis

```
function Nom_fonction(parametre1,..., parametreN)
{
  // instructions...
}

function Nom_fonction()
{
  // instructions...
}
```

Propriétés

Propriété	Description
arguments[]	tableau correspondant aux arguments passés à la fonction.
arguments.callee	spécifie le corps de la fonction qui invoque l'exécution de la fonction courante.
arguments.caller	spécifie le nom de la fonction qui invoque l'exécution de la fonction courante.
arguments.length	spécifie le nombre d'arguments passés à la fonction.
arity	spécifie le nombre d'arguments déclarés par la fonction (NE 4).
constructor	spécifie la fonction qui crée un prototype de classe d'objet.
length	spécifie le nombre d'arguments déclarés par la fonction.
prototype	autorise l'ajout de propriété à un objet Function.

Méthodes

Méthode	Description
apply(objet, arguments)	permet d'appliquer une méthode à d'un autre objet dans le contexte d'objets différents.
call	permet l'appel (exécution) d'une méthode d'un autre objet dans le contexte d'objets différents.
toSource	retourne une chapine de caractères représentant le code source de la fonction.
toString	retourne une chaîne de caractères représentant le code source de la fonction.
valueOf	retourne une chapine de caractères représentant le code source de la fontion.

Exemple [voir]

```
<html>
 <head>
  <script language="javascript">
   function conversion()
   {
    var euro = 6.55957;
    var somme = document.formulaire.valeur.value
    if (document.formulaire.choix[0].checked == true)
    {
     resultat = euro_franc(somme, euro);
    }
    else
    {
     resultat = franc_euro(somme, euro);
    }
    if (isNaN(resultat))
    {
     document.formulaire.result.value = 0;
     document.formulaire.valeur.value = 0;
     alert("Veuillez ne saisir que des nombres !");
    }
    else
    {
     document.formulaire.result.value = resultat;
    }
   }
   var franc_euro = new Function("chiffre", "monnaie",
                        "return(chiffre/monnaie)");
   var euro_franc = new Function("chiffre", "monnaie",
                        "return(monnaie*chiffre)");
  </script>
 </head>
 <body>
  <form name="formulaire">
   <input type="text" name="valeur" size="10" value="0">
   <input type="text" name="result" size="10" value="0"><br>
   <u>Type de conversion :</u><br>
   <input type="radio" name="choix" value="eurofranc">
        Euro vers francs<br>
   <input type="radio" name="choix" value="franceuro" CHECKED>
        Francs vers Euros<br>
   <input type="button" name="calculer" value="Calculer"
         onclick="javascript:conversion()">
  </form>
 </body>
</html>
```

En savoir plus :

Netscape Microsoft

8.26 / Global

***Global* représente un objet intrinsèque dont le but est de réunir les méthodes globales en un seul objet.**

Compatibilité

Disponibilité

Javascript

Propriétés

Propriété	Description
NaN	renvoie la valeur spéciale NaN indiquant qu'une expression ne représente pas un nombre.
Infinity	renvoie une valeur initiale de Number.POSITIVE_INFINITY.

Méthodes

Méthode	Description
escape(littéral)	encode les objets String pour en permettre la lecture sur tous les ordinateurs.
eval(code)	évalue le code JScript et l'exécute.
isFinite(nombre)	renvoie une valeur booléenne indiquant si un nombre fourni est fini.
isNaN(valeur)	renvoie une valeur booléenne indiquant si une valeur correspond à la valeur réservée NaN (not a number/pas un nombre).
parseFloat(nombre)	renvoie un nombre à virgule flottante converti à partir d'une chaîne.
parseInt(chaîne-caractère, valeur)	renvoie un entier converti à partir d'une chaîne.
unescape(littéral)	décode les objets String codés au moyen de la méthode escape.

Exemple [voir]

```
<html>
 <body>
  <script language="javascript">
   var pi = 3.14159
   var nombre = parseInt(pi);
   var chaine = "150.15678215436978412530111253000478235";
   var chaine2 = "#FCE10D"
   var nombre2 = parseFloat(chaine);
   var resultat = isNaN(pi) ? "n'est pas un nombre" : "est un nombre";
   var resultat2 = isNaN(chaine) ? "n'est pas un nombre" : "est un nombre";
   var resultat3 = isNaN(chaine2) ? "n'est pas un nombre" : "est un nombre";
   document.write("<br>La valeur Pi (" + pi + ") évaluée par isNaN "
          + resultat + "<br>La valeur suivante : " + chaine
          + " évaluée par isNaN " + resultat2 + "<br>La valeur "
          + chaine2 + " évaluée par isNaN " + resultat3 + ".");
   document.write("<br>La valeur Pi vaut " + pi
          + "<br>parseInt de Pi vaut " + nombre);
   document.write("<br>Soit la chaîne de caractères "
          + chaine + ",<br>parseFloat vaut " + nombre2);
  </script>
 </body>
</html>
```

En savoir plus :

Netscape

8.27 / hidden

***hidden* représente un champ de saisie cachée au sein d'un formulaire.**

Compatibilité


Disponibilité

Javascript 1.0 côté client

Correspondance HTML

```
<input type="hidden"...>
```

Héritage

Input et HTMLElement

Synopsis

```
form.name
form.element[i]
```

Propriétés

Propriété	Description
form	spécifie le formulaire contenant le bouton.
name	correspond à l'attribut NAME de INPUT.
type	correspond à l'attribut TYPE de INPUT.
value	affiche un texte sur le bouton.

Exemple [voir]

```
<html>
 <head>
  <script language="JavaScript">
   function retourne_valeur()
   {
    var nb = form.length;
    form.vide.value = 'laltruiste.com';
    var desti = form.destinataire.value;
    var suj = form.sujet.value;
    var redirec = form.redirection.value;
    var merci = form.remerciement.value;
    var autre = form.vide.value;
    document.write('Résultat des champs cachés (' + nb + '):<br>destinataire : '
              + desti + '<br>Sujet : ' + suj + '<br>Redirection : '
              + redirec + '<br>' + merci + '<br>' + autre);
   }
  </script>
 </head>
 <body>
  <form name="form" method="post" onsubmit="return retourne_valeur()">
   <input type="hidden"
       name="destinataire"
       value="adresse@email.org">
   <input type="hidden"
       name="sujet"
       value="Documentation">
   <input type="hidden"
       name="redirection"
       value="http://www.unsite.com/">
   <input type="hidden"
       name="remerciement"
       value="Merci pour votre patience et bonne continuation...">
   <input type="hidden" name="vide">
    <input type="submit" value="Valider">
  </form>
 </body>
</html>
```

En savoir plus :

Netscape Microsoft

8.28 / History

History **permet de naviguer dans l'historique des adresses URL du navigateur.**

Compatibilité


Disponibilité

Javascript 1.0 côté client

Synopsis

window.history
frame.history
history

Propriétés

Propriété	Description
current	spécifie l'URL du document courant affiché.
length	rerésente le nombre d'entrées dans la tableau de l' historique.
next	spécifie la prochaîne URL dans le tableau de l'historique.
previous	spécifie la précédente URL dans le tableau de l'historique.

Méthodes

Méthode	Description
back	chargement de la précédente URL dans le tableau de l'historique.
forward	chargement de la prochaîne URL dans le tableau de l'historique.
go	chargement d'une URL dans le tableau de l'historique.

Exemple [voir]

```
<html>
  <body>
    Ce script donne l'adresse de la page d'origine.<br>
    <script language="javascript">
      document.write("le document est : " + document.referrer);
      document.write("<br>le document courant est : "
              + window.history.current);
      document.write("<br>le nombre de documents dans le tableau de"
              + " l'historique est : " + window.history.length);
    </script><br>
    <a href="javascript:window.history.go(-1)">
      Chargez le document précédent
    </a><br>
    <a href="javascript:window.history.go(-3)">
      Chargez le troisième document précédent
    </a><br>
  </body>
</html>
```

En savoir plus :

Netscape

Netscape Microsoft

8.29 / HTMLELement

***HTMLELement* représente la superclasse de tous les éléments HTML.**

Compatibilité


Disponibilité

Javascript 1.2 côté client

Propriétés

Propriété	Description
Pour Internet Explorer	
all[]	tous les éléments contenus dans un document HTML.
children[]	l'enfant direct de l'événement.
className	la valeur de l'attrribut CLASS.
document	l'objet Document contenant l'élément.
id	la valeur de l'attibut ID.
innerHTML	le contenu HTML au sein de l'élément concerné.
innerText	le texte au ein de l'élément HTML concerné.
lang	valeur de l'attribut LANG.
offsetHeight	la hauteur de l'élément.
offsetLeft	la coordonnée horizontale de l'élément.
offsetParent	définition du système de coordonnées de l'élément.
offsetTop	la coordonnée verticale de l'élément.
ofsetWidth	la largeur de l'élément.
outerHTML	le contenu HTML de l'élément.
outerText	le texte de l'élément.
parentElement	le conteneur de l'élément.
sourceIndex	l'index de l'élément dans document.all[].
style	le style en ligne de l'élément (STYLE).
tagName	le type de balise de l'élément.
title	le titre de l'élément (TITLE).
Pour Netscape Navigator	
handleEvent(événement)	transmission d'un événement au gestionnaire approprié.

Méthodes

Méthode	Description

contains(cible)	spécifie si un élément est contenu dans un autre.
getAttribute(nom)	récupère la valeur d'un attribut.
insertAdjacentHTML(localisation, texte)	permet d'insérer du texte HTML avant ou après l'élément.
insertAdjacentText(localisation, texte)	permet d'insérer du texte avant ou après l'élément.
removeAttribute(nom)	engendre la suppression d'un attribut.
scrollIntoView(sommet)	rend un élément visible.
setAttribute(nom, valeur)	détermine la valeur d'un attribut.

Evénements

Evénement	Description
onclick	clic de souris sur un élément HTML.
ondblclick	double-clic de souris sur un élément HTML.
onhelp	pression sur la touche F1 (IE).
onkeydown	pression sur une touche du clavier.
onkeypress	relachement d'une touche après l'avoir enfoncée.
onkeyup	relachement d'une touche du clavier.
onmousedown	pression sur un bouton de la souris.
onmousemove	déplacement du curseur de la souris au sein de l'élément.
onmouseout	déplacement du curseur de la souris hors de la zone de l'élément.
onmouseover	survol du curseur de la souris sur la zone de l'élément.
onmouseup	relachement d'un bouton de la souris.

Exemple [voir]

```
<html>
 <head>
  <style>
   .intro
   {
    position:absolute;
    left:0;
    top:0;
    layer-background-color:green;
    background-color:green;
    border:0.1px solid green;
    z-index:10;
   }
  </style>
 </head>
 <body>
  <div id="i1" class="intro"></div>
  <div id="i2" class="intro"></div>
  <div id="i3"class="intro"></div>
  <div id="i4" class="intro"></div>
  <div id="i5" class="intro"></div>
  <div id="i6" class="intro"></div>
  <div id="i7" class="intro"></div>
  <div id="i8" class="intro"></div>

  <script language="JavaScript1.2">
   var ns4 = document.layers ? 1 : 0;
   var ie4 = document.all ? 1 : 0;
   var ns6 = document.getElementById && !document.all ? 1 : 0;
   var speed = 20;
   var temp = new Array();
   var temp2 = new Array();
   if (ns4)
   {
    for (i = 1; i <= 8; i++)
    {
    temp[i] = eval("document.i" + i + ".clip");
    temp2[i] = eval("document.i" + i);
    temp[i].width = window.innerWidth / 8 - 0.3;
    temp[i].height = window.innerHeight;
    temp2[i].left = (i - 1) * temp[i].width;
    }
   }
   else if (ie4 || ns6)
   {
    var clipbottom = ns6 ? window.innerHeight :
                     document.body.offsetHeight;
    cliptop = 0;
    for (i = 1; i <= 8; i++)
    {
     temp[i] = ns6 ? document.getElementById("i" + i).style :
                     eval("document.all.i" + i + ".style");
     temp[i].width = ns6 ? window.innerWidth / 8 - 2 :
                     document.body.clientWidth / 8;
     temp[i].height = ns6 ? window.innerHeight :
                     document.body.offsetHeight;
     temp[i].left = (i - 1) * parseInt(temp[i].width);
    }
   }

   function openit()
   {
    window.scrollTo(0,0);
    if (ns4)
    {
     for (i = 1; i <= 8; i = i + 2)
       temp[i].bottom -= speed;
     for (i = 2; i <= 8; i = i + 2)
       temp[i].top += speed;
     if (temp[2].top > window.innerHeight)
       clearInterval(stopit);
    }
    else if (ie4 || ns6)
    {
```

```
      clipbottom -= speed;
      for (i = 1; i <= 8; i = i + 2)
      {
        temp[i].clip = "rect(0 auto+" + clipbottom + " 0)";
      }
      cliptop += speed;
      for (i = 2; i <= 8; i = i + 2)
      {
        temp[i].clip = "rect(" + cliptop + " auto auto auto)";
      }
      if (clipbottom <= 0)
      {
        if (ns6)
        {
          for (i = 1; i <= 8; i++)
            temp[i].display = "none";
        }
        clearInterval(stopit);
      }
    }
  }
  function gogo()
  {
    stopit = setInterval("openit()", 100);
  }
  gogo();
 </script>
</body>
</html>
```

En savoir plus :

W3C

8.30 / Image

***Image* représente un champ de saisie pour un téléchargement de fichier au sein d'un formulaire.**

Compatibilité


Disponibilité

Javascript 1.0 côté client

Correspondance HTML

<img src...>

Constructeur

new Image(largeur, hauteur)

Héritage

HTMLElement

Synopsis

document.image-name
document.images[i]
document.images.length

Propriétés

Propriété	Description
border	représente la bordure de l'élément (BORDER).
complete	valeur booléenne indiquant si le chargement d'une image est achevée.
height	spécifie la hauteur de l'élément (HEIGHT).
vspace	spécifie une marge horizontale (HSPACE).
lowsrc	représente un système d'affichage en basse résolution (LOWSRC).
name	indique le nom de l'image (NAME).
src	spécifie l'adresse URL de l'image (SRC).
vspace	spécifie une marge verticale (VSPACE).
width	spécifie la largeur de l'élément (WIDTH).

Méthodes

Méthode	Description
handleEvent	invoque le Handler pour spécifier un événement.

Evénements

Evénement	Description
onAbort	annulation du chargement d'une image.

onError	erreur au cours du chargement d'une image
onkeydown	pression sur une touche du clavier.
onkeypress	relachement d'une touche après l'avoir enfoncée.
onkeyup	relachement d'une touche du clavier.
onLoad	chargement de l'image.

Exemple [voir]

```
<html>
 <head>
  <script language="javascript">
   function VersionNav(Netscape)
   {
    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;
   }
  </script>
 </head>
 <body>
  <a href="javascript:void(0)"
      onMouseOver="if (VersionNav(3.0,4.0))
      img1.src='images/poissons.gif' "
      onMouseOut="img1.src='images/poisson_seul.gif' ">
   <img name="img1" width="50" height="50" border="0"
      src="images/poisson_seul.gif"
      onload="tempImg=new Image(0,0);
            tempImg.src='images/poissons.gif'">
  </a>
 </body>
</html>
```

En savoir plus :

Netscape Microsoft

8.31 / Input

***Input* représente un élément de saisie dans un formulaire.**

Compatibilité


Disponibilité

Javascript 1.0 côté client

Correspondance HTML

<input...>

Héritage

HTMLElement

Synopsis

form.elements[i]
form.name

Propriétés

Propriété	Description
checked	spécifie l'état de l'élément de saisie.
defaultchecked	la case à cocher est côchée par défaut (CHECKED).
form	spécifie le formulaire contenant l'élément de saisie.
name	correspond au nom de l'éléments de saisie (NAME).
type	correspond à l'attribut TYPE.
value	valeur de l'élément Input (VALUE).

Méthodes

Méthode	Description
blur	enlève le focus sur l'élément de saisie.
click	simule un clique de souris sur l'élément de saisie.
focus	donne le focus à l'élément de saisie.
handleEvent	invoque le Handler pour spécifier un événement.
select	sélectionne le texte dans l'élément de saisie.

Evénements

Evénement	Description
onBlur	perte de focus de l'élément de saisie.
onChange	la valeur de l'élément de saisie change.
onClick	clic de souris sur l'élément de saisie.
onFocus	réception de focus sur l'élément de saisie.

Exemple [voir]

```
<html>
 <head>
  <script language="JavaScript">
   function validation_form()
   {
    valide = true;
    if (!verif_vide(document.form.societe.value))
    {
     valide = false; alert('Le champ Société est vide !');
    }
    if (!verif_vide(document.form.qualite.value))
    {
     valide = false; alert('Le champ Qualité est vide !');
    }
    if (!verif_vide(document.form.nom.value))
    {
     valide = false; alert('Le champ Nom est vide !');
    }
    if (!verif_vide(document.form.prenom.value))
    {
     valide = false; alert('Le champ Prénom est vide !');
    }
    if (!verif_email(document.form.email.value))
    {
     valide = false; alert('Votre adresse eMail est invalide !');
    }
    if (!verif_vide(document.form.telephone.value))
    {
     valide = false; alert('Le champ Téléphone est vide !');
    }
    retourne_valeur();
   }

   function verif_vide(text)
   {
    return (text.length > 0);
   }

   function verif_email(adresse)
   {
    if ((adresse == "")
        || (adresse.indexOf ('@') == -1)
        || (adresse.indexOf ('.') == -1))
     return false;
    return true;
   }
  </script>
 </head>
 <body>
 ...
 </body>
</html>
```

En savoir plus :

Netscape Microsoft

8.32 / JavaArray

***javaArray* représente tableau d'applets Java.**

Compatibilité

Disponibilité

Javascript

Synopsis

```
javaArray.length
javaArray[i]
```

Propriétés

Propriété	Description
length	indique le nombre d'éléments dans le tableau d'applets Java.

Méthodes

Méthode	Description
toString	retourne une chaîne identifiant l'objet javaArray.

Exemple [voir]

```
<html>
  <head>
    <title>Démonstration de JavaArray</title>
  </head>
  <body>
    <script>
      var uneChaineJava = new java.lang.String("Une chaîne de caractères !");
      var uneTableOctets = uneChaineJava.getBytes();
      document.write('<p>Taille de la chaîne : <b>'
                     + uneTableOctets.length + '</b></p>');
      for(i = 0; i < uneChaineJava.length(); i++)
        document.write(i + ' : ' + uneTableOctets[i] + '<br>');
    </script>
  </body>
</html>
```

En savoir plus :

Netscape

8.33 / JavaClass

***JavaClass* représente une classe Java.**

Compatibilité

Disponibilité

Javascript

Synopsis

```
javaClass.membreStatique
new javaClass(termes...)
```

Propriétés

Les propriétés d'un objet JavaClass sont les champs statiques de la classe Java.

Méthodes

Les méthodes d'un objet JavaClass sont les méthodes statiques de la classe Java.

Exemple [voir]

```
<html>
 <head>
  <title>Démonstration de JavaClass</title>
 </head>
 <body>
  <script>
    var uneChaineJava = new Packages.java.lang.String("Ceci
                                        est une chaîne de caractères Java !");

    document.write("<p>Classe Java : <b>");
    document.write(uneChaineJava.getClass());
    document.write("</b></p>");

    document.write("<p>Classe JavaScript : <b>");
    document.write(uneChaineJava.constructor.name);
    document.write("</b></p>");

    document.write("<p>Longueur de la chaîne : <b>");
    document.write(uneChaineJava.length());
    document.write("</b></p>");

    document.write("<p>Valeur de la chaîne : <b>");
    document.write(uneChaineJava);
    document.write("</b></p>");
  </script>
 </body>
</html>
```

En savoir plus :

Netscape

8.34 / JavaObject

***JavaObject* représente un objet java.**

Compatibilité

Disponibilité

Javascript

Synopsis

```
javaObject.membre
new Packages.classeJava(Liste_Paramètres);
```

Propriétés

Chaque objet javaObject possède des propriétés qui ont les mêmes noms que les méthodes et champs publics de l'instance de l'objet Java qu'il représente.

Méthodes

Hérite des méthodes publiques de la classe Java qu'il représente. L'objet JavaObject hérite aussi des méthodes de java.lang.Object et une autre superclasse.

Exemple [voir]

```
<html>
 <head>
  <title>Démonstration de JavaObject</title>
 </head>
 <body>
  <applet
       name="GTClockApplet"
       code="GTClockApplet.class"
       width="162" height="40">
   <param name="chime" value="on">
   <param name="format" value="24">
   <param name="credit" value="on">
   <param name="bgcolor" value="0,0,0">
  </applet>
  <form>
   <input
       type="button" value="Démarrer l'Applet"
       onclick="document.GTClockApplet.start()">
   <input
       type="button" value="Arrêter l'Applet"
       onclick="document.GTClockApplet.stop()">
  </form>
 </body>
</html>
```

En savoir plus :

Netscape

8.35 / JavaPackage

***JavaPackage* représente un paquetage Java.**

Compatibilité

Disponibilité

Javascript

Synopsis

```
package.nomPaquetage
package.nomClasse
```

Propriétés

Les propriétés d'un objet JavaPackage sont les objets de JavaClass et d'autres JavaPackage qu'il contient.

Exemple

```
<html>
  <head>
    <title>Démonstration de javaPackages</title>
  </head>
  <body>
    <script>
      var valeurDouble = new Packages.java.lang.Double(3.14);
      var classeDouble = getClass(valeurDouble);
      var nb = valeurDouble.doubleValue();
      document.write('classe de l\'objet : ' + classeDouble);
      document.write('<br>');
      document.write('Valeur de l\'objet : ' + nb);
      document.write('<br>');
      document.write(Math.round(Math.random()*100) + ' * ' + nb + ' = ');
      document.write(Math.round(Math.random()*100)*nb);
    </script>
  </body>
</html>
```

En savoir plus :

Netscape

8.36 / JSObject

***JSObject* représente un objet Javascript.**

La classe *JSObject* est disponible dans l'archive Java *plugin.jar* pour la version JDK 1.5 et pour les versions antérieures dans le fichier *javaws.jar*, normalement présent dans le répertoire *lib* de l'environnement Java (JRE : Java Runtime Environment).

Compatibilité

Disponibilité

Javascript

Synopsis

netscape.javascript.JSObject

Méthodes

Méthode	Description
call(nom_méthode, arguments[])	invoque une méthode d'un objet Javascript.
equals(Objet, objet)	détermine si deux objets JSObject se réfèrent à la même instance.
eval(chaîne_de_caractères)	évalue une chaîne de caractères.
getMember(nom)	permet de lire une propriété d'un objet Javascript.
getSlot(index)	lit un élément de tableau d'un objet Javascript.
getWindow(applet)	retourne un objet Javascript pour la fenêtre contenant l'applet donné.
removeMember(nom)	enlève une propriété d'un objet Javascript.
setMember(nom, valeur)	définit une propriété d'un objet Javascript.
setSlot(index, valeur)	définit un élément de tableau d'un objet Javascript.
toString()	convertit un objet JSObject en une chaîne de caractères.

Exemple [voir]

```
// Importations
import java.awt.Graphics;
import java.awt.Event;

 // LiveConnect... pour JavaScript
 import netscape.javascript.JSObject ;

 public class tmin_JS extends java.applet.Applet {
 // Variables

  // Initialisation de l'applet
  public void init() {                    // Methode init()
  }

  // Dessiner l'applet
  public void paint(Graphics g) {             // Methode paint()
      g.drawString("Click here...", 5, 10) ;
  }

  // Mouse down
  public boolean mouseDown(Event e, int x, int y) {
  try  {        // create JSObject
    JSObject.getWindow (this).eval ("javascript:alert('tmin_JS click " +
                    " x="+ x + " y="+ y + "')") ;
  }
  catch  (Exception  ex) {            // Error on create JSObject
    showStatus( "Error call javascript err="+ ex );
  }
  return true ;
 }
}
```

En savoir plus :

Netscape

8.37 / Layer

***Layer* représente un élément de saisie dans un formulaire.**

Compatibilité

Disponibilité

Javascript

Correspondance HTML

<layer...>...</layer>

Synopsis

document.layers[i]

Constructeur

new Layer(largeur, parent)

Propriétés

Propriété	Description
above	représente la couche au-dessus de la couche courante
background	représente l'image d'arrière plan de la couche
bgColor	représente la couleur d'arrière plan de la couche
below	représente la couche en-dessous de la couche courante
clip.bottom	représente le bord inférieur de la partie visible de la couche
clip.height	représente la hauteur de la partie visible de la couche
clip.left	représente le bord gauche de la partie visible de la couche
clip.right	représente le bord droit de la partie visible de la couche
clip.top	représente le bord supérieur de la partie visible de la couche
clip.width	représente la largeur de la partie visible de la couche
document	représente l'objet Document associé à la couche
left	représente la position horizontale du bord gauche de la couche, en pixels, par rapport à sa couche parente
name	indique le nom de la couche (ID)
pageX	représente la position horizontale de la couche, en pixels, par rapport à la page
pageY	représente la position verticale de la couche, en pixels, par rapport à la page
parentLayer	représente un objet Layer qui contient cette couche, ou l'objet Window inclus si cette couche n'est pas emboîtée dans une autre couche
siblingAbove	représente un objet Layer de même niveau hiérarchique au-dessus de la couche courante
siblingBelow	représente un objet Layer de même niveau hiérarchique en-dessous de la couche courante

src	représente une chaîne de caractères spécifiant l'adresse URL du contenu de la couche
top	représente la position verticale du bord supérieur de la couche, en pixels, par rapport à l'origine de sa couche parente
visibility	spécifie la visibilité ou l'invisibilité de la couche
window	représente la fenêtre qui contient la couche
x	correspond à *Layer.left*
y	correspond à *Layer.top*
zIndex	représente la position de la couche dans l'ordre d'empilement.

Méthodes

Méthode	Description
captureEvents(event.TYPE)	spécifie les types d'événements à capturer
handleEvent(événement)	transmet au gestionnaire approprié un événement
load(source, largeur)	change le contenu et la largeur d'une couche
moveAbove(cible)	déplace la couche au-dessus d'une autre couche
moveBelow(cible)	déplace la couche en-dessous d'une autre couche
moveBy(x, y)	déplace la couche vers une position relative à la position courante
moveTo(x, y)	déplace le coin supérieur gauche de la couche au coordonnées spécifiées
moveToAbsolute(x, y)	déplace la couche des coordonnées spécifiées à l'intérieur de la page
releaseEvents(event.TYPE)	cesse la capture du type d'événement spécifié
resizeBy(hauteur, largeur)	redimensionne la hauteur et la largeur de la couche selon les attributs relatifs
resizeTo(hauteur, largeur)	redimensionne la hauteur et la largeur de la couche
routeEvent	transmet un événement capturé au prochain gestionnaire.

Evénements

Evénement	Description
onBlur	perte de focus de l'élément.
onFocus	réception de focus sur l'élément.
onLoad	chargement de l'élément.
onMouseOver	déplacement de la souris sur la zone de l'élément.
onMouseOut	déplacement de la souris hors de la zone de l'élément.

Exemple [voir]

```
<html>
 <head>
  <title>Démonstration de Layer</title>
  <script language="JavaScript"><!--
   var nav;
   var timer;
   var temps = new Date;
   var mois = new Array('Janvier','Février','Mars','Avril','Mai',
                'Juin', 'Juillet','Août','Septembre',
                'Octobre','Novembre','Décembre');
   var j_semaine = new Array('Dimanche','Lundi','Mardi',
                  'Mercredi','Jeudi','Vendredi',
                  'Samedi');
   var val_date;
   var val_heure;
   var couche_date;
   var couche_heure;

   function init()
   {
    if (navigator.appName == "Netscape")
     nav = "NE" ;
    else nav = "IE";
     if (nav == "NE")
     {
      couche_date = document.layers['dat'];
      couche_heure = document.layers['heu'];
     }
   }

   function date()
   {
    var y = temps.getFullYear();
    var m = temps.getMonth();
    var j_s = temps.getDay();
    var j_m = temps.getDate();
    val_date = j_semaine[j_s] + ' '
          + j_m + ' ' + mois[m]
          + ' ' + y;
    if (nav == "NE")
    {
     couche_date.document.open();
     couche_date.document.write('<b style="font-size:14pt;'
               + 'font-family:Verdana; color:#000000;'
               + 'font-weight:bold">'+val_date+'</b>';
     couche_date.document.close();
    }
    else
     dat.innerHTML = '<b style="font-size:14pt;'
            + 'font-family:Verdana; color:#000000;'
            + 'font-weight:bold">'+val_date+'</b>';
   }

   function heure()
   {
    temps = new Date;
    h = temps.getHours();
    m = temps.getMinutes();
    s = temps.getSeconds();
    heures = (h <= 9) ? ('0' + h) : h;
    minutes = (m <= 9) ? ('0' + m) : m;
    secondes = (s <= 9) ? ('0' + s) : s;
    val_heure = heures + ' : ' + minutes + ' : ' + secondes;
    if (nav == "NE")
    {
     couche_heure.document.open();
     couche_heure.document.write('<b style="font-size:14pt;'
              + 'font-family:Verdana; color:#000000;'
              + 'font-weight:bold">' + val_heure + '</b>');
     couche_heure.document.close();
    }
    else
     heu.innerHTML = '<b style="font-size:14pt;'
         + 'font-family:Verdana; color:#000000;'
```

```
            +' font-weight:bold">' + val_heure + '</b>';
        if (s % 20 == 0) date();
          timer=setTimeout('heure()',1000);
       }
      </script>
     </head>
     <body onload="init();date();heure();" onunload="clearTimeout(timer)">
      <div id="dat"
            style="position:absolute;
                left:20;top:20;
                width:300;height:100">
      </div>
      <div id="heu"
            style="position:absolute;
                left:320;top:20;
                width:150;height:100">
      </div>
     </body>
    </html>
```

En savoir plus :

Netscape

8.38 / Link

***Link* représente un lien hypertexte.**

Compatibilité


Disponibilité

Javascript 1.0 côté client

Correspondance HTML

```
<a href=...>...</a>
<area href=...>
```

Héritage

HTMLElement

Synopsis

```
document.links[i]
document.links.length
```

Propriétés

Propriété	Description
hash	sépcifie un nom d'ancre dans une URL.
host	spécifie le nom de domaine de l'hôte ou l'adresse IP du réseau hôte.
hostname	indique le nom de domaine du serveur hôte.
href	spécifie l'URL complète du lien (HREF).
pathname	retourne le chemin d'accès du lien.
port	indique le port de communication que le serveur utilise.
protocol	spécifie le début de l'URL en incluant les deux points (http:).
search	spécifie une requête sous forme de chaîne de caractères.
target	correspond à l'attribut TARGET.
text	spécifie le texte contenu dans les balises A.
x	représente la position horizontale du lien en pixels.
y	représente la position verticale du lien en pixels.

Méthodes

Méthode	Description
handleEvent	invoque le gestionnaire pour spécifier un événement.

Evénements

Evénement	Description
Pour Area :	
onDblClick	double-clic de la souris.
onMouseOut	curseur de la souris sort de la zone de l'élément.
onMouseOver	curseur de la souris dans la zone de l'élément.
Pour A :	
onClick	clic de souris sur le lien.
onDblClick	double-clic de la souris.
onKeyDown	une touche du clavier est enfoncée.
onKeyPress	la touche venant d'être enfoncée est relâchée.
onKeyUp	une touche du clavier est relâchée.
onMouseDown	bouton de la souris enfoncé sur l'élément Link.
onMouseOut	curseur de la souris sort de la zone de l'élément.
onMouseUp	bouton de la souris relaché sur l'élément Link.
onMouseOver	curseur de la souris dans la zone de l'élément.

Exemple [voir]

```
<html>
  <head>
    <script>
      function voirlien()
      {
        msgWindow = window.open("","message","width=450,height=200")
        msgWindow.document.write("Nombre de liens dans la page : "
                    + document.links.length + "<br>")
        for (var i = 0; i < document.links.length; i++)
        {
          msgWindow.document.write( "Adresse (" + i + "): "
                    + document.links[i] + "<br>")
        }
      }
    </script>
  </head>
  <body>
    <a href="..">
      L'Altruiste : Le guide de votre site Web</a><br>
    <a href="courshtml/">
      Cours sur le langage HTML</a><br>
    <a href="courscss/">
      Cours sur les feuilles de style</a><br>
    <a href="coursjavascript/">
      Cours Javascript</a><br>
    <a href="http://www.microsoft.com/france/scripting/default.htm">
      Microsoft JScript 5.5</a><br>
    <a href="http://developer.netscape.com/docs/manuals
                    /js/client/jsref/contents.htm">
      Netscape Javascript 1.3</a><br><br>
    <a href="javascript:voirlien()">
      Cliquez ici pour afficher toutes les URL</a>
  </body>
</html>
```

En savoir plus :

Netscape

Netscape Microsoft

8.39 / Location

Location **représente l'emplacement pointé par le navigateur.**

Compatibilité


Disponibilité

Javascript 1.0 côté client

Adresse URL

Protocole//Hôte:Port/Chemin#Ancre?Recherche

(http://www.laltruiste.com/coursjavascript/index.html#LettreJ)

Synopsis

location
window.location

Propriétés

Propriété	Description
hash	sépcifie un nom d'ancre dans une URL (adresse#Ancre).
host	spécifie le nom de domaine de l'hôte ou l'adresse IP du réseau hôte.
hostname	indique le nom de domaine du serveur hôte.
href	spécifie l'URL complète du lien (HREF).
pathname	retourne le chemin d'accès du lien.
port	indique le port de communication que le serveur utilise.
protocol	spécifie le protocole de l'URL en incluant les deux-points (http:).
search	spécifie une requête de recherche (URL?Requête).

Méthodes

Méthode	Description
reload	force un rechargement du document courant.
replace	charge une URL spécifiée au-dessus du document courant.

Exemple [voir]

```
<html>
 <head>
  <script language="javascript">
   var cours = ['un.html','deux.html','trois.html'];
   var taille = cours.length;
   var navigateur = navigator.appName;
   var hote = document.location.host;
   var localisation = document.location.pathname;
   var racine = (hote != '' || navigateur == 'Netscape')
            ? localisation.split('/') : localisation.split(/&&/);
   var condition = racine.length;
   var fichier = racine[condition - 1];
   var acces = localisation.replace(fichier, '');

   document.write('<br>Navigateur : ' + navigateur
            + '<br>localisation : ' + localisation
            + '<br>hôte : ' + hote
            + '<br>fichier : ' + fichier
            + '<br>accès : ' + acces);

   function suivant()
   {
    for (var i = 0; i < taille; i++)
    {
     var situe = (acces + cours[i]);
     if (situe == localisation)
     {
      if (i != (taille - 1))
      {
       window.location.replace(cours[i + 1]);
       return true;
      }
      else
      {
       window.location.replace(cours[0]);
       return true;
      }
     }
    }
    alert('Le document suivant est invalide'
         + '&nVeuillez en informer le Webmaster !')
    window.location.replace(cours[0]);
   }

   function precedent()
   {
    for (var i = 0; i < taille; i++)
    {
     var situe = (acces + cours[i]);
     if (situe == localisation)
     {
      if (i != 0)
      {
       window.location.replace(cours[i - 1]);
       return true;
      }
      else
      {
       window.location.replace(cours[taille - 1]);
       return true;
      }
     }
    }
    alert('Le document précédent est invalide'
         + '&nVeuillez en informer le Webmaster !')
    window.location.replace(cours[0]);
   }
  </script>
 </head>
 <body>
  <a href="javascript:precedent()">Document précédent</a>
  << Première Page >>
  <a href="javascript:suivant()">Document suivant</a>
```

```
  </body>
</html>
```

Second exemple [voir]

```
<html>
  <head>
    <title>Fenêtre principale</title>
  </head>
  <frameset rows="15%,*">
    <frame name="cadre_superieur" src="cadre_superieur.html">
    <frameset cols="25%,*">
      <frame name="cadre_gauche" src="cadre_gauche.html">
      <frame name="cadre_droit" src="cadre_droit.html">
    </frameset>
    <noframes>
      <body>
      </body>
    </noframes>
  </frameset>
</html>

<html>
  <head>
    <title>Cadre gauche</title>
    <script language="Javascript" type="text/javascript">
      function chargement(page,cadre) {
        parent.frames[cadre].location.href = page;
      }
      function double_chargement(page1, page2) {
        parent.frames["cadre_droit"].location.href = page1;
        parent.frames["cadre_superieur"].location.href = page2;
      }
    </script>
  </head>
  <body>
    <h1>Cadre gauche</h1>
    <a href="http://www.laltruiste.com/index.html"
           target="cadre_droit">
      Lien HTML
    </a>
    <br>
    <a href="Javascript:chargement('http://www.laltruiste.com/contact.html',
                                   'cadre_droit')">
      Lien Javascript simple
    </a>
    <br>
    <a href="Javascript:double_chargement('../sommaire.html',
                '../../images/interface/bandeau-468x60.gif')">
      Lien Javascript double
    </a>
  </body>
</html>

<html>
  <head>
    <title>Cadre supérieur</title>
  </head>
  <body>
    <h1>Cadre supérieur</h1>
  </body>
</html>

<html>
  <head>
    <title>Cadre droit</title>
  </head>
  <body>
    <h1>Cadre droit</h1>
  </body>
</html>
```

En savoir plus :

Netscape Microsoft

8.40 / Math

***Math* constitue un support de fonctions et de constantes mathématiques.**

Compatibilité


Disponibilité

Javascript 1.0

Synopsis

```
Math.constant
Math.function()
```

Propriétés

Propriété	Description
E	représente la constante d'Euler E (env 2.178).
LN10	représente le logarithme naturel base 10, $\log_e 10$ (env 2.302).
LN2	représente le logarithme naturel base 2, $\log_e 2$ (env 0.693).
LOG10E	représente le logarithme naturel base e, $\log_{10} e$ (env 0.434).
LOG2E	représente le logarithme naturel base 10, $\log_2 e$ (env 1.442).
PI	représente le ratio de la circonférence d'un cercle et de son diamètre divisé par deux, Pi (3.14159).
SQRT1_2	représente l'inverse de la racine carrée (0.707).
SQRT2	représente la racine carrée de 2 (1.414).

Méthodes

Méthode	Description
abs(x)	retourne la valeur absolue d'un nombre.
acos(x)	retourne l'arc cosinus en radians d'un nombre.
asin(x)	retourne l'arc sinus en radians d'un nombre.
atan(x, y)	retourne l'arc tangente en radians d'un nombre.
atan2(x, y)	retourne l'angle entre l'axe des x et un point.
ceil(x)	retourne l'arrondi inférieur d'un nombre.
cos(x)	retourne le cosinus d'un nombre.
exp(x)	retourne e^x ou e est la constante d'Euler.
floor(x)	retourne l'arrondi vers le bas d'un nombre.
log(x)	retourne le logarithme (base E) d'un nombre.
max(a, b)	retourne la plus grande des valeurs.
min(a, b)	retourne la plus petite des valeurs.

pow(x, y)	calcule x^y.
random()	retourne un nombre pseudo-aléatoire.
round(x)	retourne l'arrondi de la valeur d'un nombre à l'entier le plus prôche.
sin(x)	retourne le sinus d'un nombre.
sqrt(x)	retourne la racine carrée d'un nombre.
tan(x)	retourne la tangente d'un nombre.

Exemple [voir]

```
<html>
 <head>
  <script language="javascript">
   var NDIGITS = 16;
   var STACKSIZE = 12;
   var value = 0;
   var level = 0;
   var entered = true;
   var decimal = 0;
   var fixed = 0;
   var exponent = false;

   function stackItem()
   {
    this.value = 0;
    this.op = "";
   }

   function array(length)
   {
    this[0] = 0;
    for (i=0; i<length; ++i)
    {
     this[i] = 0;
     this[i] = new stackItem();
    }
    this.length = length;
   }

   stack = new array(STACKSIZE);

   function push(value,op,prec)
   {
    if (level == STACKSIZE)
     return false;
    for (i = level; i > 0; --i)
    {
     stack[i].value = stack[i-1].value;
     stack[i].op = stack[i-1].op;
     stack[i].prec = stack[i-1].prec;
    }
    stack[0].value = value;
    stack[0].op = op;
    stack[0].prec = prec;
    ++level;
    return true;
   }

   function pop()
   {
    if (level==0)
     return false;
    for (i = 0; i < level; ++i)
    {
     stack[i].value = stack[i+1].value;
     stack[i].op = stack[i+1].op;
     stack[i].prec = stack[i+1].prec;
    }
    --level;
    return true;
   }

   function format(value)
   {
    var valStr = "" + value;
    if (valStr.indexOf("N") >= 0 ||
        (value == 2*value && value == 1 + value))
     return "Error ";
    var i = valStr.indexOf("e")
    if (i >= 0)
    {
     var expStr = valStr.substring(i + 1,valStr.length);
     if (i > NDIGITS - 5) i = NDIGITS - 5;
      valStr = valStr.substring(0,i);
```

```
    if (valStr.indexOf(".") < 0)
      valStr += ".";
    valStr += " " + expStr;
  }
  else
  {
    var valNeg = false;
    if (value < 0)
    {
      value = -value;
      valNeg = true;
    }
    var expval = Math.log(value)*Math.LOG10E;
    if (value == 0)
    {
      expval = 0;
    }
    else if (expval > NDIGITS-5)
    {
      expval = Math.floor(expval);
      value /= Math.pow(10, expval);
    }
    else if (-expval > NDIGITS-5)
    {
      expval = Math.ceil(expval);
      value /= Math.pow(10, expval);
    }
    else
    {
      expval = 0;
    }
    var valInt = Math.floor(value);
    var valFrac = value - valInt;
    var prec = NDIGITS - (""+valInt).length - 1;
    if (prec < 0)
      return "Error"
    if (!entered && fixed > 0)
      prec = fixed;
    var mult = " 1000000000000000000".substring(1,prec + 2);
    var frac = Math.floor(valFrac * mult + 0.5);
    valInt = Math.floor(Math.floor(value * mult + .5) / mult);
    if (valNeg)
      valStr = "-" + valInt;
    else
      valStr = "" + valInt;
      var fracStr = "00000000000000" + frac;
      fracStr = fracStr.substring(fracStr.length-prec, fracStr.length);
      i = fracStr.length - 1;
     if (entered || fixed == 0)
     {
       while (i >= 0 && fracStr.charAt(i)=="0")
       --i;
       fracStr = fracStr.substring(0, i + 1);
     }
     if (i >= 0)
       valStr += "." + fracStr;
     if (expval != 0)
     {
       var expStr = "" + expval;
       valStr += " " + expStr;
     }
  }
  return valStr;
}

function refresh()
{
  var display = format(value);
  if (exponent)
  {
    if (expval < 0)
      display += " " + expval;
    else
      display += " +" + expval;
  }
```

```
  if (display.indexOf(".")<0 && display != "Error ")
  {
   if (entered || decimal>0)
     display += '.';
    else
      display += ' ';
  }
  display = "             " + display;
  display = display.substring(display.length - NDIGITS - 1,display.length);
  document.calculator.result.value = display;
}

function clearDisp()
{
  exponent = false;
  value = 0;
  enter();
  refresh();
}

function clearAll()
{
  level = 0;
  clearDisp();
}

function evalx()
{
  if (level == 0)
    return false;
  op = stack[0].op;
  sval = stack[0].value;
  if (op == "+")
    value = sval + value;
  else if (op == '-')
    value = sval - value;
  else if (op == '*')
    value = sval * value;
  else if (op == '/')
    value = sval / value;
  else if (op == 'pow')
    value = Math.pow(sval,value);
  pop();
  if (op=='(')
    return false;
  return true;
}

function openp()
{
  enter();
  if (!push(0,'(',0))
    value = "NAN";
  refresh();
}

function closep()
{
  enter();
  while (evalx()) ;
    refresh();
}

function operator(op)
{
  enter();
  if (op == '+' || op == '-')
    prec = 1;
  else if (op == '*' || op == '/')
    prec = 2;
  else if (op == "pow")
    prec = 3;
  if (level > 0 && prec < = stack[0].prec)
     evalx();
  if (!push(value,op,prec))
```

```
    value = "NAN";
  refresh();
}

function enter()
{
  if (exponent)
    value = value * Math.exp(expval * Math.LN10);
  entered = true;
  exponent = false;
  decimal = 0;
  fixed = 0;
}

function equals()
{
  enter();
  while (level > 0)
    evalx();
  refresh();
}

function digit(n)
{
  if (entered)
  {
    value = 0;
    digits = 0;
    entered = false;
  }
  if (n == 0 && digits == 0)
  {
    refresh();
    return;
  }
  if (exponent)
  {
    if (expval < 0)
      n = -n;
    if (digits < 3)
    {
      expval = expval * 10 + n;
      ++digits;
      refresh();
    }
    return;
  }
  if (value < 0)
    n = -n;
  if (digits < NDIGITS - 1)
  {
    ++digits;
    if (decimal > 0)
    {
      decimal = decimal * 10;
      value = value + (n / decimal);
      ++fixed;
    }
    else
      value = value * 10 + n;
  }
  refresh();
}

function bksp()
{
  if (entered)
  {
    refresh();
    return;
  }
  if (digits == 0)
  {
    refresh();
    return;
```

```
  }
  if (exponent)
  {
    if (expval < 0)
      expval = -Math.floor(-expval / 10);
    else
      expval = Math.floor(expval / 10);
    --digits;
    refresh();
    return;
  }
  if (decimal > 1)
  {
    if (value < 0)
      value = -Math.floor(-value * decimal / 10);
    else
      value = Math.floor(value * decimal / 10);
    decimal = decimal / 10;
    value = value/decimal;
    --fixed;
    if (decimal == 1)
      decimal = 0;
  }
  else
  {
    if (value < 0)
      value = -Math.floor(-value / 10);
    else
      value = Math.floor(value / 10);
    decimal = 0;
  }
  --digits;
  refresh();
}

function sign()
{
  if (exponent)
    expval = -expval;
  else
    value = -value;
  refresh();
}

function period()
{
  if (entered)
  {
    value = 0;
    digits = 1;
  }
  entered = false;
  if (decimal == 0)
    decimal = 1;
  refresh();
}

function exp()
{
  if (entered || exponent)
    return;
  exponent = true;
  expval = 0;
  digits = 0;
  decimal = 0;
  refresh();
}

function func(f)
{
  enter();
  if (f=="1 / x")
  {
    value = 1 / value;
  }
```

```
      else if (f == 'n!')
      {
        if (value < 0 || value > 200 || value != Math.round(value))
          value = "NAN";
        else
        {
          var n = 1;
          var i;
          for (i = 1; i <= value; ++i)
            n *= i;
          value = n;
        }
      }
      else
      {
        if (f == "sin")
          value = Math.sin(value / 180 * Math.PI);
        else if (f == "cos")
          value = Math.cos(value / 180 * Math.PI);
        else if (f == "tan")
          value = Math.tan(value / 180 * Math.PI);
        else if (f == "log")
          value = Math.log(value) / Math.LN10;
        else if (f == "log2")
          value = Math.log(value) / Math.LN2;
        else if (f == "ln")
          value = Math.log(value);
        else if (f == "sqrt")
          value = Math.sqrt(value);
        else if (f == "pi")
          value = Math.PI;
        else if (f == "asin")
          value = Math.asin(value)*180 / Math.PI;
        else if (f=="acos")
          value = Math.acos(value)*180 / Math.PI;
        else if (f == "atan")
          value = Math.atan(value)*180/Math.PI;
        else if (f == "alog")
          value = Math.exp(value * Math.LN10);
        else if (f == "alog2")
          value = Math.exp(value * Math.LN2);
        else if (f == "exp")
          value = Math.exp(value);
        else if (f == "sqr")
          value = value*value;
        else if (f  == "e")
          value = Math.E;
      }
      refresh();
    }
  </script>
 </head>
 <body onload="refresh()">
  ...
 </body>
</html>
```

En savoir plus :

Netscape Microsoft

8.41 / MimeType

***MimeType* représente un type de données MIME (Multipart Internet Mail Extension).**

Compatibilité

Disponibilité

Javascript

Synopsis

```
navigator.mimeTypes[i]
navigator.mimeTypes["type"]
navigator.mimeTypes.length
```

Propriétés

Propriété	Description
description	indique une description d'un type MIME.
enabledPlugin	représente une extension logicielle qui manipule le type MIME.
suffixes	inscrit les possibles extensions des noms de fichier associées au type MIME.
type	correspond au nom du type MIME.

Exemple [voir]

```
<html>
 <body>
  <script language="javascript">
   if (navigator.appName == 'Netscape')
   {
    document.writeln("<table border=1><tr valign=TOP>"
              + "<th align=left>Nb</th>"
              + "<th align=left>Type</th>"
              + "<th align=left>Description</th>"
              + "<th align=left>Suffixes</th>"
              + "<th align=left>Plug-Ins</th></tr>")
    for (i = 0; i < navigator.mimeTypes.length; i++)
    {
     document.writeln("<tr valign=TOP><td>" + i
      + "</td><td>",navigator.mimeTypes[i].type
      + "</td><td>",navigator.mimeTypes[i].description
      + "</td><td>",navigator.mimeTypes[i].suffixes)
     if (navigator.mimeTypes[i].enabledPlugin == null)
     {
      document.writeln("</td><td>Absent</td></tr>")
     }
     else
     {
      document.writeln("</td><td>"
       + navigator.mimeTypes[i].enabledPlugin.name
       + "</td></tr>")
     }
    }
    document.writeln("</table>")
   }
   else
    document.write("Ce programme ne fonctionne que sous Netscape");
  </script>
 </body>
</html>
```

En savoir plus :

Netscape

8.42 / Navigator

***Navigator* représente les caractéristiques du navigateur en cours d'utilisation.**

Compatibilité


Disponibilité

Javascript 1.0 côté client

Synopsis

navigator

Propriétés

Propriété	Description
appCodeName	retourne le nom de code du navigateur.
appName	spécifie le nom du navigateur.
appVersion	spécifie la version du navigateur.
language	indique le langage par défaut du navigateur.
mimeTypes	détermine un tableau de tous les types MIME supportés par le navigateur.
platform	indique la plateforme sur laquelle tourne le navigateur.
plugins	détermine un tableau des plug-ins installés sur le navigateur.
userAgent	spécifie des informations relatifs au navigateur.

Méthodes

Méthode	Description
javaEnabled()	teste si Java est activé.
plugins.refresh()	rend disponibles les extensions logicielles récemment installées (NE).
preference(preference, valeur)	permet de fixer ou de retirer les préférences de l'utilisateur (NE).
savePreferences()	sauvegarde les préférences de l'utilisateur dans un fichier local *prefs.js* (NE).
taintEnabled()	spécifie si la corruption des données est activée (NE).

Exemple [voir]

```
<html>
 <body>
  <script language="javascript">
   var navigateur = navigator.appName;
   var version = navigator.appVersion;
   var code = navigator.appCodeName;
   var langage = navigator.language;
   var plateforme = navigator.platform;
   var langsys = navigator.systemLanguage;
   var agent = navigator.userAgent;
   var langue = navigator.userLanguage;
   document.write('Navigateur : ' + navigateur
            + '<br>Version : ' + version
            + '<br>Code : ' + code
            + '<br>Langage NE : ' + langage
            + '<br>Plateforme : ' + plateforme
            + '<br>Langage Système : ' + langsys
            + '<br>Agent : ' + agent
            + '<br>Langage IE : ' + langue);
  </script>
 </body>
</html>
```

En savoir plus :

Netscape Microsoft

8.43 / Number

Number **permet de prendre en charge les nombres.**

Compatibilité


Disponibilité

Javascript 1.1

Synopsis

new number(valeur)
number(valeur)

Propriétés

Propriété	Description
constructor	spécifie la fonction que crée un prototype d'objet.
MAX_VALUE	représente un nombre maximale utilisable.
MIN_VALUE	représente un nombre minimale utilisable.
NaN	représente une valeur qui n'est pas un nombre (Not a Number).
NEGATIVE_INFINITY	représente un infini négatif.
POSITIVE_INFINITY	représente un infini positif.
prototype	permet l'addition de propriétés à un objet Number.

Méthodes

Méthode	Description
toLocaleString()	retourne une date convertie en chaîne en utilisant les paramètres régionaux en cours.
toSource()	retourne un objet littéral représentant l'objet Number spécifié.
toString(racine)	retourne une chaîne de caractères représentant l'objet Number spécifié.
valueOf()	retourne la valeur primitive de l'objet Number spécifié.

Exemple [voir]

```
<html>
 <head>
  <script language="javascript">
   function verifie()
   {
    var nombre = new Number(formulaire.valeur.value);
    var val_max = Number.MAX_VALUE;
    var val_min = Number.MIN_VALUE;
    var invalide = Number.NaN;
    var inf_pos = Number.POSITIVE_INFINITY;
    var inf_neg = Number.NEGATIVE_INFINITY;
    if (nombre.toString() == invalide.toString())
     alert("La valeur fournie est invalide !&nElle est égale à : " + nombre);
    else
    {
     document.write("<br>La valeur est un nombre : " + nombre);
     document.write("<br>La valeur Number.MAX_VALUE : " + val_max);
     document.write("<br>La valeur Number.MIN_VALUE : " + val_min);
     document.write("<br>La valeur Number.NaN : " + invalide);
     document.write("<br>La valeur Number.POSITIVE_INFINITY : " + inf_neg);
     document.write("<br>La valeur Number.NEGATIVE_INFINITY : " + inf_pos);
    }
   }
  </script>
 </head>
 <body>
  <form name="formulaire">
   <input type="text" name="valeur" size="20" maxlength="20">
   <input type="button" value="Vérifier" onclick="verifie()">
  </form>
 </body>
</html>
```

En savoir plus :

Netscape Microsoft

8.44 / Object

***Object* représente la superclasse qui contient des caractéristiques propres à tous les objets Javascript.**

Compatibilité


Disponibilité

Javascript 1.0

Synopsis

```
new Object()
new Object(valeur)
```

Propriétés

Propriété	Description
constructor	spécifie la fonction qui crée un prototype d'objet.
prototype	permet l'addition de propriétés à tous les objets.

Méthodes

Méthode	Description
assign(valeur)	surcharge l'opération d'affectation (utilisation désapprouvée).
eval(code)	évalue une chaîne de caractères d'un code Javascript dans le contexte de l'objet spécifié (utilisation désapprouvée).
toLocaleString()	retourne une date convertie en chaîne de caractères en utilisant les paramètres régionaux en cours.
toSource	retourne un objet littéral représentant l'objet spécifié.
toString	retourne une chaîne de caractères représentant l'objet spécifié.
unwatch	enlève un point d'observation d'une propriété de l'objet.
valueOf	retourne la valeur primitive de l'objet spécifié.
watch	ajoute un point d'observation à la propriété de l'objet.

Exemple [voir]

```
<html>
 <head>
  <script>
   objet = new Object();
   horloge = new Object();
   function estompe(obj, arriere, taux, nettete)
   {
    if (obj != "[object]")
    {
     setTimeout("estompe(" + obj + ","
              + arriere + "," + taux + ","
              + nettete + ")", 0);
     return;
    }
    clearTimeout(horloge[obj.sourceIndex]);
    diff = arriere - obj.filters.alpha.opacity;
    direction = 1;
    if (obj.filters.alpha.opacity > arriere)
    {
     direction = -1;
    }
    nettete = Math.min(direction * diff, nettete);
    obj.filters.alpha.opacity += direction * nettete;
    if (obj.filters.alpha.opacity != arriere)
    {
     objet[obj.sourceIndex] = obj;
     horloge[obj.sourceIndex] = setTimeout("estompe(objet["
                + obj.sourceIndex + "],"
                + arriere + "," + taux + ","
                + nettete + ")", taux);
    }
   }
  </script>
 </head>
 <body>
  <img src="images/cyclisme.gif"
       border="0" width="104" height="85"
       style="filter:alpha(opacity = 10)"
       onmouseover="estompe(this,100,30,5)"
       onmouseout="estompe(this,10,96,5)">
 </body>
</html>
```

En savoir plus :

Netscape Microsoft

8.45 / Option

Option **représente une option dans un élément Select.**

Compatibilité


Disponibilité

Javascript 1.0 côté client

Correspondance HTML

<option...>... </option>

Héritage

HTMLElement

Constructeur

new Option(texte, valeur, Sélection_par_défaut, sélection)

Synopsis

select.options[i]

Propriétés

Propriété	Description
defaultSelected	spécifie l'état de la sélection initiale de l'option (selected).
index	indique la position de l'option dans l'élément Select.
length	indique le nombre d'options dans l'élément Select.
selected	indique si l'option est sélectionnée.
text	représente l'étiquette de l'option (LABEL).
value	spécifie la valeur de l'option retournée lorsque le formulaire est soumis (VALUE).

Exemple [voir]

```
<html>
 <head>
  <title>Démonstration de Option</title>
 </head>
 <body>
  <form name="form">
   <table border="0" width="100%" height="178">
    <tr>
     <td width="35%" valign="top" align="left">
      <select name="selecteur" size="6"
              onchange="afficheImage(this.options[this.selectedIndex].value)">
       <option value="images/javascript15.gif" selected>
        Javascript 1.5</option>
       <option value="images/jscript.gif">
        Javascript et VBScript</option>
       <option value="images/javascriptPro.jpg">
        Javascript Professionnel</option>
       <option value="images/javascriptO.gif">
        Javascript Précis & Concis</option>
       <option value="images/300astuces.gif">
        300 astuces pour HTML et Javascript</option>
      </select>
     </td>
     <td width="65%" valign="top" align="left">
      <ilayer id="couche1" width="100%" height="178">
      <layer id="couche2" width="100%" height="178">
      <div id="couche3"></div></layer></ilayer>
     </td>
    </tr>
   </table>
  </form>
  <script language="javascript">
   var description = new Array()
    description[0]="Javascript 1.5";
    description[1]="Javascript et VBScript";
    description[2]="Javascript Professionnel";
    description[3]="Javascript Précis & Concis";
    description[4]="300 astuces pour HTML et Javascript";

   var ie4 = document.all;
   var ns6 = document.getElementById;
   var tempobj = document.form.selecteur;
   if (ie4 || ns6)
    var contenu = document.getElementById ?
            document.getElementById("couche3") :
           document.all.couche3;
   function afficheImage(image)
   {
    if (ie4 || ns6)
    {
     contenu.innerHTML = '<center>Chargement en cours...</center>';
     contenu.innerHTML = '<center><img src="'+ image + '"><br><br>'
           + description[tempobj.options.selectedIndex] +'</center>';
    }
    else if (document.layers)
    {
     document.couche1.document.couche2.document.write("
           + '<center><img src="' + image + '"><br><br>'
           + description[tempobj.options.selectedIndex]
           + '</center>');
     document.couche1.document.couche2.document.close()
    }
    else
     alert('Vous avez besoin de Explorer 5.x, Netscape 4.x ou plus !')
   }

   function afficheDefaut()
   {
    afficheImage(tempobj.options[tempobj.options.selectedIndex].value);
   }

   if (ie4 || ns6 || document.layers)
   {
    if (tempobj.options.selectedIndex != -1)
```

```
    {
      if (ns6)
        afficheDefaut();
      else
        window.onload = afficheDefaut;
    }
  }
 </script>
</body>
</html>
```

En savoir plus :

Netscape Microsoft

8.46 / Packages

***Packages* représente un objet au plus haut niveau qui a accès aux classes de Java de l'intérieur le code de JavaScript.**

Compatibilité

Disponibilité

Javascript

Synopsis

Packages.nomClasse

Propriétés

Propriété	Description
className	représente le nom d'une classe Java dans un paquetage autre que Netscape, Java, ou Sun qui est disponible pour JavaScript.
java	représente n'importe quelle calsse dans un paquetage Java : Java.*.
netscape	représente n'importe quelle calsse dans un paquetage Java : netscape.*.
sun	représente n'importe quelle calsse dans un paquetage Java : sun.*.

Exemple [voir]

```
<html>
 <head>
  <title>Démonstration de Packages</title>
 </head>
 <body>
  <script>
   var dateJava = new Packages.java.util.Date();

   var heure = dateJava.getHours();
   var minutes = dateJava.getMinutes();
   var secondes = dateJava.getSeconds();

   var jour = dateJava.getDate();
   var mois = dateJava.getMonth() + 1;
   var annee = 1900 + dateJava.getYear();

   document.write('<p><u>Date Java :</u> <b>'
                  + dateJava + '</b></p>');
   document.write('<p><u>Jour :</u> <b>'
                  + jour + ' / ' + mois + ' / ' + annee + '</b></p>');
   document.write('<p><u>Temps :</u> <b>'
                  + heure + ' : ' + minutes + ' : ' + secondes + '</b></p>');
  </script>
 </body>
</html>
```

En savoir plus :

Netscape

8.47 / Password

Password **représente un élément de saisie textuel destiné au mot de passe.**

Compatibilité


Disponibilité

Javascript 1.0 côté client

Correspondance HTML

```
<input type="password"...>
```

Héritage

Input et HTMLElement

Synopsis

```
form.elements[i]
form.name
```

Propriétés

Propriété	Description
defaultValue	représente la valeur par défaut de l'élément.
form	spécifie le formulaire contenant l'élément de saisie.
name	correspond au nom de l'élément de saisie.
type	correspond à l'attribut TYPE.
value	valeur de l'élément Password.

Méthodes

Méthode	Description
blur	enlève le focus sur l'élément de saisie.
focus	donne le focus à l'élément de saisie.
handleEvent	invoque le Handler pour spécifier un événement.
select	sélectionne le texte dans l'élément de saisie.

Evénements

Evénement	Description
onBlur	perte de focus de l'élément de saisie.
onFocus	réception de focus sur l'élément de saisie.

Exemple [voir]

```
<html>
 <head>
  <title>Démonstration de Password</title>
  <script language="JavaScript">
   function ValidationForm(form){
    if (form.nom.value == "") {
     alert("Entrez un nom s'il vous plaît !");
     return false;
    }
    if ((form.motpasse1.value == "")
         || (form.motpasse2.value == "")) {
     alert("Entrez un mot de passe s'il vous plaît !");
     return false;
    }
    if (form.motpasse1.value != form.motpasse2.value ) {
     alert("ATTENTION, les mots de passe sont différents !");
     form.motpasse1.value = "";
     form.motpasse2.value = "";
     form.motpasse1.focus();
     return false;
    }
    return true;
   }
  </script>
 </head>
 <body>
  <form type="post" action="page_cible.url"
        onsubmit="return ValidationForm( this );">
   Utilisateur  :
    <input type="text" name="nom" size="30"><br>
   Mot de passe :
    <input type="password" name="motpasse1" size="30"><br>
   Mot de passe (Vérification):
    <input type="password" name="motpasse2" size="30"><br>
   <input type="submit" value="Valider">
   <input type="reset" value="Annuler"></p>
  </form>
 </body>
</html>
```

Exemple [voir]

```
<html>
 <head>
  <title>Démonstration de Password</title>
  <script language="javascript">
   function connecter()
   {
    utilisateur = document.connexion.nom.value;
    mot_passe = document.connexion.password.value;
    verif = utilisateur + mot_passe + ".html";
    location= verif;
    return true;
   }
  </script>
 </head>
 <body>
  <form type="post" name="connexion">
   <input name="nom" value="utilisateur" type="text">
   <input name="password" type="password" value="motdepasse">
   <br><br>
   <input value="Valider" type="button" onclick="connecter()">
  </form>
 </body>
</html>
```

En savoir plus :

Netscape

Microsoft

8.48 / Plugin

***Plugin* décrit les extensions logicielles installées.**

Compatibilité

Disponibilité

Javascript

Synopsis

```
navigator.plugins[i]
navigator.plugins['nom']
```

Propriétés

Propriété	Description
description	donne une description d'une extension logicielle.
filename	donne le nom de fichier du programme d'extension.
length	indique le nombre d'élément dans le tableau des types MIME.
name	spécifie le nom de l'extension logicielle.

Exemple [voir]

```
<html>
  <body>
    <script language="javascript">
      if (navigator.appName == 'Netscape')
      {
        document.writeln("<table border=1><tr valign=TOP>"
                  + "<th align=left>Nb</th>"
                  + "<th align=left>Nom</th>"
                  + "<th align=left>Nom de fichier</th>"
                  + "<th align=left>Description</th>")
        for (i = 0; i < navigator.plugins.length; i++)
        {
          document.writeln("<tr valign=TOP><td>" + i
            + "</td><td>",navigator.plugins[i].name
            + "</td><td>",navigator.plugins[i].filename
            + "</td><td>",navigator.plugins[i].description);
        }
        document.writeln("</td></tr></table>")
      }
      else
        document.write("Ce programme ne fonctionne que sous Netscape");
    </script>
  </body>
</html>
```

En savoir plus :

Netscape

8.49 / PrivilegeManager

***PrivilegeManager* représente la classe Java utilisée par les scripts signés.**

Compatibilité

Disponibilité

Javascript

Synopsis

netscape.security.PrivilegeManager

Méthodes

Méthode	Description
disablePrvilege(privilège)	désactive un privilège.
enablePrivilege(privilège)	active un privilège.

En savoir plus

Netscape

8.50 / Radio

***Radio* représente une case radio dans un formulaire.**

Compatibilité


Disponibilité

Javascript 1.0 côté client

Correspondance HTML

```
<input type="radio"...>
```

Héritage

Input et HTMLElement

Synopsis

```
form.elements[i]
form.name
```

Propriétés

Propriété	Description
checked	spécifie l'état de l'élément de saisie.
defaultchecked	la case à cocher est côchée par défaut (CHECKED).
form	spécifie le formulaire contenant l'élément de saisie.
name	correspond au nom de l'éléments de saisie (NAME).
type	correspond à l'attribut TYPE.
value	valeur de l'élément Radio (VALUE).

Méthodes

Méthode	Description
blur	enlève le focus sur l'élément de saisie.
click	simule un clique de souris sur l'élément de saisie.
focus	donne le focus à l'élément de saisie.
handleEvent	invoque le Handler pour spécifier un événement.

Evénements

Evénement	Description
onBlur	perte de focus de l'élément de saisie.
onChange	la valeur de l'élément de saisie change.
onFocus	réception de focus sur l'élément de saisie.

Exemple [voir]

```
<html>
  <head>
    <title>Démonstration de Radio</title>
    <script language="JavaScript">
      etat = 0;
      g = 0;
      args = new Array();
      noms = new Array();
      z = -1;
      if (document.images)
      {
        var selectionnee = new Image();
        selectionnee.src = "images/selection.gif";
        var deselectionnee = new Image();
        deselectionnee.src = "images/nonselection.gif";
      }
      function CreeRadio(nom, valeur, label, selection)
      {
        ok = false;
        inc = 0;
        if (document.images)
        {
          for (var i = 0; i < noms.length; i++)
          {
            if (noms[i] == nom)
            {
              ok = true;
              inc = i;
            }
          }
          if(ok == false)
          {
            ifhidden = '<input type="hidden" name="' + nom
                    + '" value="' + valeur + '">';
            taille_noms = noms.length;
            args[taille_noms] = new Array();
            args[taille_noms][0] = new Array();
            args[taille_noms][0][0] = nom;
            args[taille_noms][0][1] = valeur;
            noms[noms.length] = nom;
          }
          else
          {
            ifhidden = "";
            taille_args = args[inc].length
            args[inc][taille_args] = new Array();
            args[inc][taille_args][0] = nom;
            args[inc][taille_args][1] = valeur;
          }
          if(selection==1)
          {
            image="images/selection.gif";
            attribut="CHECKED";
          }
          else
          {
            image="images/nonselection.gif";
            attribut="UNCHECKED";
          }
          arg_nom = "'" + nom + "'";
          arg_valeur = "'" + valeur + "'";
          document.write('<a href="javascript:changement(' + arg_nom + ', '
                  + arg_valeur + ', ' + etat + ')"><img src="' + image
                  + '" name="' + valeur +'" border="0"></a>  '
                  + label + '<br>' + ifhidden);
          if(selection==1)
          {
            document.form.elements[nom].value = valeur;
          }
          etat++
        }
        else
        {
          browserver = navigator.userAgent;
```

```
        if (browserver.indexOf("Mozilla/2") != -1)
        {
          if(selection == 1)
          {
            attribut = "CHECKED";
          }
          else
          {
            attribut = "";
          }
          document.write('<input type="radio" name="' + nom + '" value="'
                  + valeur + '" ' + image + '>' + label + '<br>');
        }
      }
    }
    function changement(arg_nom , arg_valeur, arg_etat)
    {
      if(document.form.elements[arg_nom].value != arg_valeur)
      {
        for (i = 0; i < args.length; i++)
        {
          for (j = 0; j < args[i].length; j++)
          {
            if(args[i][j][0] == arg_nom)
            {
              document.images[args[i][j][1]].src = eval('deselectionnee.src');
            }
          }
        }
        document.images[arg_valeur].src = eval('selectionnee.src');
        document.form.elements[arg_nom].value = arg_valeur;
      }
    }
  </script>
 </head>
 <body>
  <form name="form">
   Sélectionner l'un des formats :<br><br>
   <script>
    CreeRadio("format", "Text", "Texte", 1)
    CreeRadio("format", "HTML", "HTML", 0)
    CreeRadio("format", "XML", "XML", 0)
    CreeRadio("format", "CSS", "CSS", 0)
    CreeRadio("format", "XSL", "XSL", 0)
   </script>
  </form>
 </body>
</html>
```

En savoir plus :

Netscape Microsoft

8.51 / RegExp

***RegExp* représente une expression régulière utilisée pour la mise en correspondance de motifs.**

Compatibilité


Disponibilité

Javascript 1.2

Constructeur

```
new RegExp(modèle, attribut);
```

La valeur attribut de l'expression régulière peut être *g* ou/et **i**. Respectivement, pour une recherche global du modèle et pour l'autre en ignorant la casse du modèle.

Propriétés

Propriété	Description
$1, ..., $9	retourne les neuf propriétés les plus récemment mémorisées parmi celles qui ont été trouvées au cours de la correspondance de modèles.
$_	voir input.
$*	voir multiline.
$&	voir lastMatch.
$+	voir lastParen.
$`	voir leftContext.
$'	voir rightContext.
constructor	spécifie la fonction qui crée un prototype d'objet.
global	indique si une expression régulière est mise en correspondance d'une manière globale.
ignoreCase	spécifie si une expression régulière est sensible à la casse.
input	représente une chaîne de caractère contre laquelle une expression régulière correspond.
lastIndex	retourne la position du caractère où débute la prochaine correspondance trouvée dans une chaîne recherchée.
lastMatch	représente le texte du dernier appariement.
lastParen	retournee le dernier sous-ensemble de caractères entre parenthèses correspondant aux critères d'une recherche d'expression régulière.
leftContext	représente le texte qui précède la dernière mise en correspondance.
multiline	indique si les mises en correspondances sont effectuées en mode multiligne.
prototype	permet d'ajouter des propriétés à tous les objets.
rightContext	représente le texte qui suit la dernière mise en correspondance.
source	spécifie le texte du modèle.

Méthodes

Méthode	Description
compile(motif, attributs)	compile un objet expression régulière.
exec(Chaîne_de_caractères, attribut)	exécute une recherche de l'expression régulière dans la chaîne de caractères spécifiée.
test(Chaîne_de_caractères)	retourne une valeur booléenne *true* selon si la chaîne de caractère spécifiée contient l'expression régulière.
toSource()	retourne un objet littéral représentant l'objet spécifié.
toString()	retourne une chaîne de caractères représentant l'objet spécifié.
valueOf()	retourne le valeur primitive de l'objet spécifié.

Exemple [voir]

```
<html>
  <body>
    <script language="javascript">
      modele = /(&w+)&s(&w+)/;
      chaine = "AMBROISE Caroline";
      resultat = chaine.replace(modele, "Mlle $2 $1");
      document.write("Expression initiale : " + chaine + "<br>");
      document.write("Expression transformée : " + resultat);
    </script>
  </body>
</html>
```

En savoir plus :

Netscape Microsoft

8.52 / Reset

***Reset* représente un bouton d'annulation dans un formulaire.**

Compatibilité

 

Disponibilité

Javascript 1.0 côté client

Correspondance HTML

```
<input type="reset"...>
```

Héritage

Input et HTMLElement

Synopsis

```
form.elements[i]
form.name
```

Propriétés

Propriété	Description
form	spécifie le formulaire contenant le bouton.
name	correspond au nom du bouton (NAME).
type	correspond à l'attribut TYPE.
value	valeur de l'élément Input (VALUE).

Méthodes

Méthode	Description
blur	enlève le focus sur le bouton.
click	simule un clique de souris sur l'élément Reset.
focus	donne le focus à le bouton.
handleEvent	invoque le Handler pour spécifier un événement.

Evénements

Evénement	Description
onBlur	perte de focus sur le bouton.
onClick	clique de souris sur l'élément Reset.
onFocus	réception de focus sur le bouton.

Exemple [voir]

```
<html>
 <head>
  <title>Démonstration de Reset</title>
  <script language="JavaScript">
   function Evenement()
   {
    code = event.keyCode;
    if (code == 13)
    {
     affichageCode.innerHTML += "Entrée = code inexistant<br>";
    }
    else if (code == 32)
    {
     affichageCode.innerHTML += "Space =  <br>";
    }
    else if (code == 8)
    {
     affichageCode.innerHTML += "Backspace = code inexistant<br>";
    }
    else if (code == 27)
    {
     Effacement();
    }
    else
    {
     affichageCode.innerHTML += "&#" + code + "; =  "
                 + "&#" + code + ";<br>";
    }
   }
   function Effacement()
   {
    affichageCode.innerHTML = "";
   }
  </script>
 </head>
 <body onkeypress="Evenement()">
  <p>Pressez une touche du clavier pour voir son code hexadécimal.</p>
  <input type="reset" value="Effacer" onclick="Effacement()"><br><br>
  <div id="affichageCode"></div>
 </body>
</html>
```

En savoir plus :

Netscape Microsoft

8.53 / Screen

***Screen* contient des propriétés décrivant la résolution d'écran.**

Compatibilité


Disponibilité

Javascript 1.2 côté client

Synopsis

```
screen
```

Propriétés

Propriété	Description
availHeight	spécifie la hauteur de l'écran en pixels moins les particularités d'interface du système comme la barre des tâches.
availLeft	spécifie la coordonnée x de la première pixel disponible sur l'axe horizontal (NE4).
availTop	spécifie la coordonnée y de la première pixel disponible sur l'axe vertical (NE4).
availWidth	spécifie la largeur de l'écran en pixels moins les particularités d'interface du système comme la barre des tâches.
colorDepth	donne la profondeur en bit de la palette de couleurs du navigateur.
height	indique la hauteur de l'écran.
pixelDepth	affiche la résolution en couleur de l'écran (NE4).
width	indique la largeur de l'écran.

Exemple [voir]

```
<html>
 <head>
  <script>
   var winheight = 100;
   var winsize = 100;
   var x = 5;

   function ouverture(adresse)
   {
    temp = adresse
    if (!(window.resizeTo && document.all)&&
          !(window.resizeTo && document.getElementById))
    {
     window.open(adresse);
     return;
    }
    fenetre = window.open("","","scrollbars");
    fenetre.moveTo(0,0);
    fenetre.resizeTo(100,100);
    depart();
   }

   function depart()
   {
    if (hauteur >= screen.availHeight - 3)
     x = 0;
    fenetre.resizeBy(5, x);
    hauteur += 5;
    taille += 5;
    if (taille >= screen.width - 5)
    {
     fenetre.location = temp;
     hauteur = 100;
     taille = 100;
     x = 5;
     return;
    }
    setTimeout("depart()", 50)
   }
  </script>
 </head>
 <body>
  <a href="javascript:ouverture('http://developer.netscape.com/index.html')">
   Netscape
  </a><br>
  <a href="javascript:ouverture('http://www.microsoft.com/france/scripting/
        default.htm?/france/scripting/jscript/default.htm')">
   Microsoft
  </a>
 </body>
</html>
```

En savoir plus :

Netscape Microsoft

8.54 / Select

***Select* représente une liste de choix dans un formulaire.**

Compatibilité


Disponibilité

Javascript 1.0 côté client

Correspondance HTML

<select...>... </select>

Héritage

Input et HTMLElement

Synopsis

form.elements[i]
form.name

Propriétés

Propriété	Description
form	spécifie le formulaire contenant l'élément de saisie.
length	représente le nombre d'options dans la liste de choix.
name	correspond au nom de la liste de sélection.
options	spécifie les éléments options de la liste de choix.
type	correspond à l'attribut TYPE.
selectedIndex	représente la sélection d'une option.

Méthodes

Méthode	Description
blur	enlève le focus sur l'élément Select.
focus	donne le focus à la liste de choix.
handleEvent	invoque le Handler pour spécifier un événement.

Evénements

Evénement	Description
onBlur	perte de focus de l'élément Select.
onChange	la valeur de l'élément Select change.
onFocus	réception de focus sur la liste de choix.

Exemple [voir]

```
<html>
 <head>
  <title>Démonstration de TextArea</title>
  <script language="JavaScript">
   function verif()
   {
    var valide = false;
    with (document.form)
    {
     if (sujet.selectedIndex == 0)
     {
      alert("Sélectionnez un sujet SVP.");
      sujet.focus();
     }
     else
     {
      confirm('Le sujet sélectionné est : '
            + sujet.options[sujet.selectedIndex].value)
      return valide;
     }
    }
   }
  </script>
 </head>
 <body>
  <form name="form" onsubmit="return verif();">
   <select name="sujet">
    <option select>Sélectionner un sujet</option>
    <option value="commentaire">1. Commentaire</option>
    <option value="Question">2. Question</option>
    <option value="Suggestion">3. Suggestion</option>
    <option value="Lien cassé">4. Lien cassé</option>
    <option value="Autre">5. Autre</option>
   </select>
   <input type="submit" value="Soumettre">
  </form>
 </body>
</html>
```

En savoir plus :

Netscape Microsoft

8.55 / String

***String* représente une chaîne de caractère.**

Compatibilité


Disponibilité

Javascript 1.0

Constructeur

new String(valeur)

Propriétés

Propriété	Description
constructor	spécifie la fonction qui crée un prototype d'objet.
length	représente la longueur de la chaîne de caractères.
prototype	permet l'ajout de propriétés à un objet chaîne de caractères.

Méthodes

Méthode	Description
anchor(nom)	crée une ancre **.
big()	permet de mettre en gras un objet String *<b>*.
blink()	provoque un clignotement de l'objet <blink>.
bold()	permet de mettre en gras un objet String *<b>*.
charAt(index)	retourne le caractère à l'index spécifié.
charCodeAt(index)	retourne un nombre indiquant la valeur Unicode du caractère à l'index spécifié.
concat(chaine)	permet de caoncaténer deux objets String et retourne le résultat.
fixed()	applique une largeur de caractères fixe à l'objet String *<tt>*.
fontcolor(couleur)	applique une couleur à la chaîne de caractères *<font color="couleur">*.
fontsize(taille)	applique une largeur de police à l'objet String *<font size="largeur">*.
fromCharCode(Unicode)	retourne une chaîne de caractère créée en utilisant la séquence Unicode spécifiée.
indexOf(valeur, début)	recherche une chaîn de caractère dans un objet String.
italics()	permet une mise en italique de l'objet String *<I>*.
lastIndexOf(valeur, début)	recherche une chaîne de caractère de l'arrière vers l'avant.
link(url)	crée un lien hypertexte **.
match(RegExp)	trouve une ou plusieurs mise en correspondance à partir de l'expression régulière spécifiée.
replace(valeur)	retourne une nouvelle chaîne de caractère en remplaçant les mises en correspondance avec une expression régulière.
search(RegExp)	exécute une recherche d'une expression régulière sur un objet String.
slice(début, fin)	extrait une section d'une chaîne de caractères.
small()	permet d'afficher un objet String en petite taille *<small>*.
split(délimiteur)	sépare un objet String en plusieurs sous-chaînes de caractères en utilisant un délimiteur.
strike()	affiche un texte barré *<strike>*.
sub()	permet une mise en indice *<sub>*.
substr(début, longueur)	extrait une sous chaîne de caractères.
substring(début, fin)	retourne une sous chaîne de caractères.
sup()	permet une mise en exposant *<sup>*.
toLowerCase()	convertit une chaîne de caractère en minuscules.
toSource()	retourne une chaîne de caractère représentant le code source de l'objet (NE).
toString()	retourne une chaîne de caractères représentants l'objet spécifié.
toUpperCase()	convertit une chaîne de caractère en majuscules.
valueOf()	retourne le valeur primitive de l'objet spécifié.

Exemple [voir]

```
<html>
 <body>
  <script language="javascript">
   var chaine = new String("Une chaîne de caractère");
   document.write("Taille de 45        : " + chaine.fontsize(45));
   document.write("<br>Tout en majuscules : " + chaine.toUpperCase());
   document.write("<br>Tout en minuscules : " + chaine.toLowerCase());
   document.write("<br>Clignotement (NE)  : " + chaine.blink());
   document.write("<br> En gras        : " + chaine.bold());
   document.write("<br>En italique       : " + chaine.italics());
   document.write("<br>En barré        : " + chaine.strike());
  </script>
 </body>
</html>
```

En savoir plus :

Netscape Microsoft

8.56 / Style

***Style* représente les paramètres courants de tous les styles en ligne possible pour un objet donné.**

Compatibilité


Disponibilité

Javascript 1.2

Correspondance HTML

```
<balise style="...">
<balise class="...">
<balise id="...">
```

Synopsis

```
document.classes.NomClasse.NomBalise
document.contextual(...)
document.ids.NomElement
document.tags.NomBalise
htmlElement.style //IE
```

Propriétés

Propriété	Description
align	aligne l'élément HTML dans son élément parent *align*.
background	spécifie l'ensemble des styles d'arrière plan d'un élément.
backgroundAttachment	indique si l'image de fond doit être fixe.
backgroundColor	applique une couleur de fond à un élément.
backgroundImage	applique une image en rrière plan à un élément.
backgroundPosition	spécifie le position de l'image de fond.
backgroundPositionX	spécifie le position horizontale de l'image de fond.
backgroundPositionY	spécifie le position verticale de l'image de fond.
backgroundRepeat	spécifie le type de répétitions d'une image.
border	spécifie l'ensemble des styles de bordure.
borderBottom	applique un style à la bordure inférieure.
borderBottomColor	applique une couleur à la bordure inférieure.
borderBottomStyle	applique un style de cadre à la bordure inférieure.
borderBottomWidth	donne la largeur de la bordure inférieure.
borderColor	applique une couleur à la bordure.
borderLeft	applique un style à la bordure gauche.
borderLeftColor	applique une couleur à la bordure gauche.
borderLeftStyle	applique un style de cadre à la bordure gauche.
borderLeftWidth	donne la largeur de la bordure gauche.
borderRight	applique un style à la bordure droite.
borderRightColor	applique une couleur à la bordure droite.
borderRightStyle	applique un style de cadre à la bordure droite.
borderRightWidth	donne la largeur de la bordure droite.
borderStyle	spécifie le style de bordure.
borderTop	applique un style à la bordure supérieure.
borderTopColor	applique une couleur à la bordure supérieure.
borderTopStyle	applique un style de cadre à la bordure supérieure.
borderTopWidth	donne la largeur de la bordure supérieure.
borderWidth	applique une largeur aux bordures.
clear	spécifie si les côtés d'un élément HTML sont flottants.
clip	spécifie une zone de visibilité de l'élément.
color	applique une couleur au texte de l'élément.
cssText	.
cursor	indique le type de curseur de souris.

display	spécifie le type d'affichage de l'élément.
filter	spoécifie un filtre à appliquer à un élément.
font	spécifie l'ensemble des styles de police.
fontFamily	applique un famille de police au texte.
fontSize	spécifie une taille de police.
fontStyle	spécifie un style de police.
fontVariant	détermine une police à afficher en petites majuscules.
fontWeight	spécifie le type de caractère d'une police.
height	spécifie la hauteur de l'élément.
left	spécifie la position gauche de l'élément par rapport au bord du document.
letterSpacing	indique l'espace entre les lettres.
lineHeight	spécifie la hauteur de ligne.
listStyle	spécifie le style de la liste.
listStyleImage	spécifie une image comme puce.
listStylePosition	spécifie la position de la liste.
listStyleType	spécifie le style de puces affichées dans une liste d'items.
margin	spécifie la taille de toutes les marges.
marginBottom	spécifie une marge inférieure.
marginLeft	spécifie une marge gauche.
marginRight	spécifie une marge droite.
marginTop	spécifie une marge supérieure.
overflow	détermine le comportement de l'élément lorsqu'il dépasse le cadre d'affichage.
paddingBottom	spécifie une espace de remplissage inférieur.
paddingLeft	spécifie une espace de remplissage gauche.
paddingRight	spécifie une espace de remplissage droit.
paddingTop	spécifie une espace de remplissage supérieur.
pageBreakAfter	indique où placer une fin de page et à quelle page (gauche ou droite) le contenu ultérieur devrait reprendre.
pageBreakBefore	indique où placer une fin de page et à quelle page (gauche ou droite) le contenu ultérieur devrait reprendre.
pixelHeight	spécifie la hauteur de l'élément en pixels.
pixelLeft	spécifie la position gauche de l'élément en pixels.
pixelTop	spécifie la position supérieur de l'élément en pixels.
pixelWidth	spécifie la largeur de l'élément en pixels.
posHeight	spécifie la hauteur de l'élément dans l'unité spécifié.
position	retrouve le type de positionnement utilisé pour l'élément.

posLeft	spécifie le positionnement horizontal pour l'élément.
posTop	spécifie le positionnement vertical pour l'élément.
posWidth	spécifie la largeur de l'élément dans l'unité spécifié.
styleFloat	place ou préserve l'attribut CSS de flottement, qui indique si l'élément flotte, faisant glisser le texte autour de lui.
textAlign	permet d'aligner un texte.
textDecoration	indique le type de décoration d'un texte.
textDecorationBlink	spécifie un clignotement du texte.
textDecorationLineThrough	spécifie un texte barré.
textDecorationNone	indique qu'aucune décoration ne s'applique à l'élément.
textDecorationOverline	spécifie un texte surligné.
textDecorationUnderline	spécifie un texte souligné.
textIndent	spécifie la longueur de l'indentation d'un texte.
textTransform	spécifie la casse d'un texte.
whiteSpace	spécifie si les espaces blancs à l'intérieur d'un élément HTML doivent permettre de passer à la ligne.
top	spécifie la position supérieure de l'élément par rapport au bord du document.
verticalAlign	spécifie l'alignement vertical d'un élément HTML.
visibility	indique la visibilité de l'élément.
width	spécifie la largeur.
zIndex	spécifie la position de l'élément dans la pile.

Méthodes

Méthode	Description
borderWidths(haut, droit, bas, gauche)	spécifie la largeur des bordures d'un élément HTML.
margins(haut, droit, bas, gauche)	spécifie les marges entre l'élément HTML et le reste des éléments adjacents.
paddings(haut, droit, bas, gauche)	définit les espaces de remplissages entre un élément HTML et son contenu.

Exemple [voir]

```
<html>
 <head>
  <style>
   .photo {position:relative}
  </style>
  <script language="JavaScript1.2">
   var degre = 5;
   var arret = 0;
   var etape = 1;

   function init(image)
   {
    arret = 0;
    tremblement = image;
    tremblement.style.left = 0;
    tremblement.style.top = 0;
   }

   function agitation()
   {
    if ((!document.all && !document.getElementById) || arret == 1)
     return;
    if (etape == 1)
    {
     tremblement.style.top = parseInt(tremblement.style.top) + degre;
    }
    else if (etape == 2)
    {
     tremblement.style.left = parseInt(tremblement.style.left) + degre;
    }
    else if (etape == 3)
    {
     tremblement.style.top = parseInt(tremblement.style.top) - degre;
    }
    else
    {
     tremblement.style.left = parseInt(tremblement.style.left) - degre;
    }
    if (etape < 4)
     etape++;
    else
     etape = 1;
    setTimeout("agitation()", 30)
   }

   function stabilisation(image)
   {
    arret = 1;
    image.style.left = 0;
    image.style.top = 0;
   }
  </script>
 </head>
 <body>
  <img src="images/informatique.jpg"
       border="0" width="200" height="133"
       class="photo"
       onMouseover="init(this);agitation()"
       onMouseout="stabilisation(this)">
 </body>
</html>
```

En savoir plus :

Netscape Microsoft

8.57 / Submit

***Submit* représente un bouton de soumission dans un formulaire.**

Compatibilité


Disponibilité

Javascript 1.0 côté client

Correspondance HTML

```
<input type="submit"...>
```

Héritage

Input et HTMLElement

Synopsis

```
form.elements[i]
form.name
```

Propriétés

Propriété	Description
form	spécifie le formulaire contenant le bouton.
name	correspond au nom du bouton (NAME).
type	correspond à l'attribut TYPE.
value	valeur de l'élément Input (VALUE).

Méthodes

Méthode	Description
blur	enlève le focus sur le bouton.
click	simule un clique de souris sur l'élément Submit.
focus	donne le focus à le bouton.
handleEvent	invoque le Handler pour spécifier un événement.

Evénements

Evénement	Description
onBlur	perte de focus sur le bouton.
onClick	clique de souris sur l'élément Submit.
onFocus	réception de focus sur le bouton.

Exemple [voir]

```
<html>
 <head>
  <script language="JavaScript" type="text/javascript">
   function validation_form()
   {
    valide = true;
    if (!verif_vide(document.form.societe.value))
    {
     valide = false; alert('Le champ Société est vide !');
    }
    if (!verif_vide(document.form.qualite.value))
    {
     valide = false; alert('Le champ Qualité est vide !');
    }
    if (!verif_vide(document.form.nom.value))
    {
     valide = false; alert('Le champ Nom est vide !');
    }
    if (!verif_vide(document.form.prenom.value))
    {
     valide = false; alert('Le champ Prénom est vide !');
    }
    if (!verif_email(document.form.email.value))
    {
     valide = false; alert('Votre adresse eMail est invalide !');
    }
    if (!verif_vide(document.form.telephone.value))
    {
     valide = false; alert('Le champ Téléphone est vide !');
    }
    if (valide == true)
    {
     alert('Merci !');
     return valide;
    }
    else
     alert('Veuillez remplir les champs correctement SVP !\nMerci !');
   }

   function verif_vide(text)
   {
    return (text.length > 0);
   }

   function verif_email(adresse)
   {
    if ((adresse == "")
       || (adresse.indexOf ('@') == -1)
       || (adresse.indexOf ('.') == -1))
     return false;
    return true;
   }

   function active()
   {
    document.form.soumission.focus()
   }
  </script>
 </head>
 <body onload="active()">
  <form name="form" onsubmit="validation_form()">
   <input type="text" name="societe" size="40" maxlength="40"><br>
   <input type="text" name="qualite" size="40" maxlength="40"><br>
   <input type="text" name="nom" size="40" maxlength="40"><br>
   <input type="text" name="prenom" size="40" maxlength="40"><br>
   <input type="text" name="email" size="40" maxlength="40"><br>
   <input type="text" name="telephone" size="40" maxlength="40"><br>
   <input type="submit" name="soumission" value="Envoyer">
   <input type="reset" name="annulation"value="Annuler">
  </form>
 </body>
</html>
</html>
```

En savoir plus :

Netscape Microsoft

8.58 / Text

***Text* représente un élément de saisie textuel dans un formulaire.**

Compatibilité


Disponibilité

Javascript 1.0 côté client

Correspondance HTML

<input type="text"...>

Héritage

Input et HTMLElement

Synopsis

form.elements[i] form.name

Propriétés

Propriété	Description
defaultValue	représente la valeur par défaut de l'élément.
form	spécifie le formulaire contenant l'élément de saisie.
name	correspond au nom de l'éléments de saisie (NAME).
type	correspond à l'attribut TYPE.
value	valeur de l'élément Input (VALUE).

Méthodes

Méthode	Description
blur	enlève le focus sur l'élément de saisie.
click	simule un clique de souris sur l'élément de saisie.
focus	donne le focus à l'élément de saisie.
handleEvent	invoque le Handler pour spécifier un événement.
select	sélectionne le texte dans l'élément de saisie.

Evénements

Evénement	Description
onBlur	perte de focus de l'élément de saisie.
onChange	la valeur de l'élément de saisie change.
onClick	clique de souris sur l'élément de saisie.
onFocus	réception de focus sur l'élément de saisie.

Exemple [voir]

```
<html>
 <head>
 <title>Démonstration de Text</title>
 <script language="JavaScript">
    function verif(form, bouton)
    {
      Retour = false;
      if (bouton.name == "1") Retour = verifSaisie1(form);
      if (bouton.name == "2") Retour = verifSaisie2(form);
      if (bouton.name == "3") Retour = verifSaisie3(form);
      if (bouton.name == "4") Retour = verifSaisie4(form);
      if (Retour)
        alert ("Saisie correcte!");
    }
    function verifSaisie1(form)
    {
      Ctrl = form.texte1;
      if (Ctrl.value == "" || Ctrl.value.indexOf ('@', 0) == -1)
      {
        validation (Ctrl, "Saisir une adresse email valide !")
        return (false);
      }
      else
        return (true);
    }
    function verifSaisie2(form)
    {
      Ctrl = form.texte2;
      if (Ctrl.value.length != 5)
      {
        validation (Ctrl, "Saisir 5 caractères maximum !")
        return (false);
      }
      else
        return (true);
    }
    function verifSaisie3(form)
    {
      Ctrl = form.texte3;
      if (Ctrl.value.length < 3)
      {
        validation (Ctrl, "Saisir au moins trois caractères !")
        return (false);
      }
      else
        return (true);
    }
    function verifSaisie4(form)
    {
      Ctrl = form.texte4;
      if (Ctrl.value == "")
      {
        validation (Ctrl, "Saisir un texte quelconque !")
        return (false);
      }
      else
        return (true);
    }
    function sousmission (form, bouton)
    {
      if (!verifSaisie1(form)) return;
      if (!verifSaisie2(form)) return;
      if (!verifSaisie3(form)) return;
      if (!verifSaisie4(form)) return;
      alert ("Toutes les saisies sont correctes !&nMerci...");
      return;
    }
    function validation(Ctrl, msg)
    {
      alert(msg)
      Ctrl.focus();
      return;
    }
  </script>
```

```
</head>
<body>
 <form name="form" method="get">
   <p>Saisir une adresse email (nom@serveur.suf): <br>
    <input type="text" name="texte1" size="20">
    <input type="button" name="1" value="Vérifier"
        onclick="verif(this.form, this)">
   </p>
   <p>Saisir 5 caractères au plus : <br>
    <input type="text" name="texte2" size="20">
    <input type="button" name="2" value="Vérifier"
        onclick="verif(this.form, this)">
    </p>
   <p>Saisir au moins trois caractères : <br>
    <input type="text" name="texte3" size="20">
    <input type="button" name="3" value="Vérifier"
        onclick="verif(this.form, this)">
   </p>
   <p>Saisir un texte sans espaces : <br>
    <input type="text" name="texte4" size="20">
    <input type="button" name="4" value="Vérifier"
        onclick="verif(this.form, this)">
   </p>
   <p>
    <input type="button" name="Soumettre" value="Submit"
        onclick="sousmission(this.form, this)">
   </p>
 </form>
</body>
</html>
```

En savoir plus :

Netscape Microsoft

8.59 / Textarea

***Textarea* représente une zone de saisie de texte multiligne dans un formulaire.**

Compatibilité


Disponibilité

Javascript 1.0 côté client

Correspondance HTML

<textarea...>... </textarea...>

Héritage

Input et HTMLElement

Synopsis

form.elements[i]
form.name

Propriétés

Propriété	Description
defaultValue	représente la valeur par défaut de l'élément.
form	spécifie le formulaire contenant l'élément de saisie.
name	correspond au nom de l'éléments de saisie (NAME).
type	correspond à l'attribut TYPE.
value	valeur de l'élément Textarea (VALUE).

Méthodes

Méthode	Description
blur	enlève le focus sur l'élément de saisie.
click	simule un clique de souris sur l'élément de saisie.
focus	donne le focus à l'élément de saisie.
handleEvent	invoque le Handler pour spécifier un événement.
select	sélectionne le texte dans l'élément de saisie.

Evénements

Evénement	Description
onBlur	perte de focus de l'élément de saisie.
onChange	la valeur de l'élément de saisie change.
onClick	clique de souris sur l'élément de saisie.
onFocus	réception de focus sur l'élément de saisie.

Exemple [voir]

```
<html>
 <head>
  <title>Démonstration de TextArea</title>
  <script language="JavaScript">
   function verif(form)
   {
    var max = 20;
    if (form.texte.value.length > max)
    {
     alert("Veuillez ne saisir qu'au plus 20 caractères !");
     return false;
    }
    else return true;
   }
  </script>
 </head>
 <body>
  <form onsubmit="return verif(this)">
   <textarea rows="5" cols="40" name="texte" wrap="hard">
    Saisir 20 caractères au maximum.
   </textarea>
   <br>
   <input type="submit" value="Soumettre">
  </form>
 </body>
</html>
```

En savoir plus :

Netscape Microsoft

8.60 / TextRange

***TextRange* représente du texte dans un document HTML.**

Compatibilité

Disponibilité

Jcript

Constructeur

createTextRange()

Propriétés

Propriété	Description
htmlText	retourne la source HTML comme un fragment HTML valide.
text	change ou récupère la chaîne de caractères contenue dans l'objet TextRange.
boundingHeight	récupère la hauteur du rectangle qui limite le TextRange.
boundingLeft	récupère la coordonnée horizontale qui limite le TextRange.
boundingTop	récupère la coordonnée verticale qui limite le TextRange.
boundingWidth	récupère la largeur du rectangle qui limite le TextRange.
offsetLeft	retourne la position gauche calculée, en pixels, par rapport à la fenêtre.
offsetTop	retourne la position supérieure calculée, en pixels, par rapport à la fenêtre.

Méthodes

Méthode	Description
collapse(bool_départ)	déplace le point d'insertion au début ou à la fin de l'objet texte courant.
compareEndPoints(type, obj_texte)	compare deux points de fin et retourne -1, 0, ou 1 pour moins que, égal à , ou plus grand que, respectivement.
duplicate()	retourne un objet TextRange dupliqué.
execCommand(commande, Interface, Valeur)	exécute une commande sur la sélection donnée ou l'objet texte spécifié.
expand(type)	étend l'objet texte pour que des unités partielles soient complètement contenues.
findText(texte, Type_Recherche, Drapeau)	permet de retrouver une chaîne de caractères dans un objet texte.
getBookmark()	récupère un signet qui peut être employé avec *moveToBookmark* pour retourner au même objet TextRange.
inRange(Référence_TextRange)	retourne *true* si le paramètre de la méthode est contenu dans l'objet TextRange ou est égal à ce dernier sur lequel la méthode est appelée.
isEqual(Objet_TextRange)	retourne *true* si le paramètre de la méthode est égal à l'objet TextRange sur lequel la méthode est appelée.

move(type, compteur)	réduit l'objet TextRange donné ou déplace l'intervalle vide par le nombre *compteur* spécifié.
moveEnd(Unité, Compteur)	change la limite de l'objet TextRange en déplaçant sa position de fin.
moveStart(Unité, Compteur)	change la limite de l'objet TextRange en déplaçant sa position de début.
moveToBookmark()	déplace un signet créé par la méthode *getBookmark*.
moveToElementText(Elément)	déplace l'intervalle des textes de sorte que les positions de début et de fin de l'intervalle entourent le texte dans l'élément donné.
moveToPoint(x, y)	déplace les positions de début et de fin de l'intervalle de texte aux coordonnées spécifiées.
parentElement()	récupère l'élément parent de l'objet TextRange spécifié. L'élément parent est l'élément qui entoure complètement le texte dans l'intervalle.
pasteHTML(texte_HTML)	colle du texte HTML dans l'objet TextRange en se substituant au texte précédent.
queryCommandEnabled(Commande)	retourne *true* si la commande a été exécutée avec succès en utilisant la méthode *ExecCommand*.
queryCommandIndeterm(Commande)	retourne *true* si la commande spécifiée est dans un état indéterminé.
queryCommandState(Commande)	retourne l'état courant de la commande.
queryCommandSupported(Commande)	retourne *true* si la commande courante est supportée sur l'ojet TextRange.
queryCommandValue(Commande)	retourne la valeur actuelle de la commande donnée sous forme de chaîne de caractères ou d'une valeur booléenne.
scrollIntoView()	provoque le défilement de l'objet TextRange dans l'affichage en alignant au-dessus ou en bas de la fenêtre.
select()	crée une sélection active sur l'intervalle de texte courant.
setEndPoint(Type, Intervalle)	place le point final d'un intervalle basé sur le point final des autres.

Exemple [voir]

```
<html>
 <head>
  <script language="JavaScript">
   var n = 0;
   function chercheSurPage(chaine)
   {
    if (chaine == "")
    return false;
    if (document.layers)
    {
     if (!window.find(chaine))
      while(window.find(chaine, false, true))
       n++;
     else
      n++;
     if (n == 0)
      alert("Ce mot clé ne se trouve pas sur la page.");
    }
    if (document.all)
    {
     var texte = window.document.body.createTextRange();
     for (var i = 0; i <= n && (var ok = texte.findText(chaine)) != false; i++)
     {
      texte.moveStart("character", 1);
      texte.moveEnd("textedit");
     }
     if (ok)
     {
      texte.moveStart("character", -1);
      texte.findText(chaine);
      texte.select();
      texte.scrollIntoView();
      n++;
     }
     else
     {
      if (n > 0)
      {
       n = 0;
       chercheSurPage(chaine);
      }
      else
       alert("Ce mot clé ne se trouve pas sur la page.");
     }
    }
    return false;
   }
  </script>
 </head>
 <body>
  <form name="recherche"
        onsubmit="return chercheSurPage(this.motcle.value);">
   <input name="motcle" type="text" size=15 onchange="n = 0;">
   <input type="submit" value="Rechercher">
  </form>
  ...
 </body>
</html>
```

En savoir plus :

Microsoft

8.61 / TextStream

***TextStream* facilite l'accès séquentiel au fichier.**

Compatibilité

Disponibilité

JScript 1

Constructeur

CreateTextFile()

Propriétés

Propriété	Description
AtEndOfLine	retourne *true* si le pointeur du fichier est situé immédiatement avant la marque de fin de ligne dans un fichier *TextStream* et sinon *false*.
AtEndOfStream	retourne *true* si le pointeur du fichier se situe à la fin d'un fichier *TextStream* et sinon *false*.
Column	retourne le numéro de colonne correspondant à la position du caractère courant dans un fichier *TextStream*.
Line	retourne le numéro de la ligne courante dans un fichier *TextStream*.

Méthodes

Méthode	Description
Close()	ferme un fichier *TextStream*.
Read(nombre)	Lit le nombre spécifié de caractères et renvoie la chaîne de caractères.
ReadAll()	lit un fichier *TextStream* entier et renvoie la chaîne de caractères.
ReadLine()	Lit une ligne complète et renvoie la chaîne de caractères obtenues.
Skip(nombre)	ignore un nombre spécifié de caractères lors de la lecture d'un fichier *TextStream*.
SkipLine	Ignore une ligne lors de la lecture d'un fichier *TextStream*.
Write(chaîne)	écrit une chaîne de caractères spécifiée dans un fichier *TextStream*.
WriteBlankLines(nombre)	écrit un nombre spécifié de caractères de nouvelle ligne dans un fichier *TextStream*.
WriteLine(chaîne)	écrit une chaîne de caractères spécifiée et un caractère de nouvelle ligne dans un fichier *TextStream*.

Exemples [voir]

```
<html>
 <head>
  <title>Démonstration de TextStream</title>
  <script language="JScript">
   var fso = new ActiveXObject("Scripting.FileSystemObject");
   var fichier = fso.CreateTextFile("c:\essai.txt", true);
   function CreeFichier()
   {
    fichier.Write("Voici un exemple de TextStream.&n");
    fichier.WriteLine("Un fichier Essai.txt a été créé
                par la commande CreateTextFile.");
    alert("Du texte a bien été écrit &ndans le fichier Essai.txt")
    fichier.Close();
   }
   function VoirTxt()
   {
    fichier = fso.OpenTextFile("c:\essai.txt", 1);
    var ligne = fichier.ReadLine();
    document.write("<u>Voici la première ligne du fichier"
            + "<i>Essai.txt</i> :</u> " + ligne);
    fichier.Close();
   }
  </script>
 </head>
 <body>
  <a href="javascript:CreeFichier()">
   Ecrire dans le fichier <i>Essai.txt</i> sur <i>C:</i>
  </a><br>
  <a href="javascript:VoirTxt()">
   Visualiser une partie du texte du fichier <i>Essai.txt</i>.
  </a><br>
 </body>
</html>
```

Référence :

Microsoft

8.62 / VBArray

***VBArray* permet un accès aux tableaux sécurisés de Visual Basic.**

Compatibilité

Disponibilité

JScript 3.0

Constructeur

```
new VBArray(vbarray);
```

Méthodes

Méthode	Description
dimensions()	renvoie le nombre de dimensions d'un objet VBArray.
getItem(dim1, dim2,..., dimN)	renvoie l'élément à l'emplacement indiqué.
lbound(dimension)	retourne l'index le moins élevé d'une dimension donnée d'un objet VBArray.
toArray()	renvoie un tableau JScript standard converti à partir d'un objet VBArray.
ubound(dimension)	retourne l'index le plus élevé d'une dimension donnée d'un objet VBArray.

Exemples [voir]

```
<html>
 <head>
  <script language="VBScript">
   Function CreeVBArray()
    Dim i, j, k
    Dim a(2, 2)
    k = 1
    For i = 0 To 2
    For j = 0 To 2
     a(j, i) = k
     document.writeln(k)
     k = k + 1
    Next
    document.writeln("<br>")
    Next
    CreeVBArray = a
   End Function
  </script>

  <script language="JScript">
   function VBArrayTest(vbarray)
   {
    var a = new VBArray(vbarray);
    var b = a.toArray();
    var i;
    for (i = 0; i < 9; i++)
    {
     document.write(b[i] + '<br>');
    }
   }
  </script>
 </head>
 <body>
  <script language="JScript">
   VBArrayTest(CreeVBArray());
  </script>
 </body>
</html>
```

En savoir plus :

Microsoft

8.63 / Window

***Window* représente une fenêtre du navigateur.**

Compatibilité


Disponibilité

Javascript 1.0 côté client

Synopsis

```
self
window
window.frames[i]
```

Propriétés

Propriété	Description
closed	spécifie si une fenêtre a été fermée.
crypto	un objet qui permet les particularités de chiffrage du Navigateur d'accès.
defaultStatus	représente l'affichage par défaut d'un message dans la barre de statut du navigateur.
document	contient l'information sur le document actuel et fournit des méthodes pour l'affichage de la production de HTML à l'utilisateur.
frames[]	représente un tableau de toutes les cadres d'une fenêtre.
history	contient l'information sur les adresses URL que les clients ont visité à l'intérieur d'une fenêtre.
innerHeight	spécifie la dimension verticale de la zone de contenu d'une fenêtre en pixels.
innerWidth	spécifie la dimension horizontale de la zone de contenu d'une fenêtre en pixels.
length	indique le nombre de cadres dans une fenêtre.
location	contient l'information sur l'adresse URL courante.
locationbar	représente la barre d'affichage d'adresse sur le navigateur.
menubar	représente la barre de menu du navigateur.
name	spécifie un nom unique pour la fenêtre.
offscreenBuffering	spécifie si les mises à jour de la fenêtre sont placées en mémoire tampon.
opener	spécifie le nom de fenêtre du document appelant quand une fenêtre est ouverte en employant la méthode open.
outerHeight	spécifie la dimension verticale en pixels, de la bordure extérieure de la fenêtre.
outerWidth	spécifie la dimension horizontale en pixels, de la bordure extérieure de la fenêtre.
pageXOffset	fournit la position x actuelle, en pixels, de la page vue d'une fenêtre.
pageYOffset	fournit la position y actuelle, en pixels, de la page vue d'une fenêtre.
parent	représente le parent du cadre.
personalbar	repérésente la barre personnelle du navigateur.
screenX	spécifie la coordonnée x du bord gauche d'une fenêtre.
screenY	spécifie la coordonnée y du bord supérieure d'une fenêtre.
scrollbars	représente la barre de défilement du navigateur.
self	est un synonyme pour la fenêtre courante.
status	spécifie une priorité ou le message temporaire s'affiche dans la barre d'état de la fenêtre.
statusbar	représente la barre de statut du navigateur.
toolbar	représente la barre d'outils du navigateur.
top	représente la fenêtre associée au cadre.
window	est un synonyme pour la fenêtre courante.

Méthodes

Propriété	Description

alert(texte)	affiche un message dans une boîte de dialogue contenant un bouton OK.
atob(Chaîne_de_données)	décode une chaîne de données qui a été encodées en employant la base 64 d'encodage.
back()	défait la dernière étape de l'historique dans n'importe quel cadre à l'intérieur d'une fenêtre de plus haut niveau.
blur()	enlève le focus sur l'élément window.
btoa(chaîne_de_caractère)	crée une base d'encodage sur 64 bits.
captureEvents(event.TYPE)	permet de capturer tous les types d'événements spécifiés produit dans la fenêtre ou le document.
clearInterval(temps)	stoppe une minuterie qui était lancé par la méthode *setInterval*.
clearTimeout(temps)	stoppe une minuterie qui était lancé par la méthode *setTimeout*.
close()	ferme la fenêtre spécifiée.
confirm(texte)	affiche une boîte de dialogue de confirmation contenant les boutons OK et Cancel.
crypto.random(nombre)	retourne une chaîne de caractères pseudo-aléatoire dont la longueur est le nombre indiqué d'octets.
crypto.signText(texte, ask/auto, autorité)	retourne une chaîne de données encodées qui représente un objet signé.
disableExternalCapture()	met hors de service l'événement externe capturé lancé par la méthode *enableExternalCapture*.
enableExternalCapture()	permet à une fenêtre avec cadre à capturer des événements dans des pages chargées à partir de différents serveurs.
find(texte, bool_cas_sensible, bool_recherche_arrière)	permet de trouver la chaîne de caractère spécifiée dans le contenu de la fenêtre indiquée.
focus()	donne le focus à l'objet spécifié.
forward()	charge la prochaîne adresse URL dans le tableau de l'historique.
handleEvent(événement)	invoque le Handler pour spécifier un événement.
home()	dirige l'URL indiqué dans des favoris comme page d'accueil du navigateur.
moveBy(horizontal, vertical)	déplace la fenêtre à partir d'une valeur spécifiée.
moveTo(x, y)	déplace le coin supérieur gauche de la fenêtre aux coordonnées d'écran spécifiées.
open(URL, Nom, séparateur)	ouvre une nouvelle fenêtre du navigateur.
print()	imprime le contenu d'une fenêtre ou d'un cadre.
prompt(texte, valeur_par_défaut)	affiche une boîte de dialogue avec un message et une zone de saisie.
releaseEvents(envent.TYPE)	cesse la capture du type d'événements spécifié.
resizeBy(horizontal, vertical)	redimensionne une fenêtre entière par le déplacement du coin inférieur droit de la fenêtre par une valeur spécifiée.
resizeTo(largeur, hauteur)	redimensionne une fenêtre entière à la hauteur et la largeur extérieures spécifiées.
routeEvent(événement)	transmet un événement capturé au prochaine gestionnaire.
scroll	permet de faire défiler une fenêtre à partir des coordonnées spécifiées.
scrollBy(horizontal, vertical)	permet de faire défiler une zone d'une fenêtre par une valeur spécifiée.

scrollTo(x, y)	permet de faire défiler une zone visible de la fenêtre par des coordonnées spécifiées, tel que le point spécifié devienne le coin supérieur gauche.
setHotKeys(true/false)	active ou désactive les raccourcis claviers dans une fenêtre.
setInterval(fonction, temps, argumentN)	évalue une expression ou appelle une fonction à intervalle régulier spécifié en millisecondes.
setResizable(true/false)	Indique si on permet à un utilisateur de redimensionner une fenêtre.
setTimeout(fonction, temps, argumentN)	évalue une expression ou appelle une fonction une fois que le temps spécifié en millisecondes se soit écoulé.
setZOptions(option)	contrôle l'empilement de la fenêtre.
stop()	stoppe le téléchargement courant.

Evénements

Propriété	Description
onBlur	la fenêtre perd son statut d'élément actif.
onDragDrop	des éléments sont déposés dans la fenêtre.
onError	erreur au cours du chargement d'une image.
onFocus	la fenêtre devient active.
onLoad	le hargement de la fenêtre est opéré.
onMove	déplacement de la fenêtre.
onResize	redimensionnement de la fenêtre.
onLoad	chargement de l'image.

Exemple [voir]

```
<html>
 <head>
  <script language="javascript">
   var vitesse = 1;
   var position_ini = 0;
   var alt = 1;
   var pos_ini1 = 0, pos_ini2 = -1;

   function defilement()
   {
    if (document.all)
     temp = document.body.scrollTop;
    else
     temp = window.pageYOffset;
    if (alt == 0)
     alt = 1;
    else
     alt = 0;
    if (alt == 0)
     pos_ini1 = temp;
    else
     pos_ini2 = temp;
    if (pos_ini1 != pos_ini2)
    {
     if (document.all)
      position_ini = document.body.scrollTop + vitesse;
     else
      position_ini = window.pageYOffset + vitesse;
     window.scroll(0, position_ini);
    }
    else
    {
     position_ini = 0;
     window.scroll(0, position_ini);
    }
   }

   function initialisation()
   {
    setInterval("defilement()", 10);
   }
   window.onload = initialisation;
  </script>
 </head>
 <body>
  ...
 </body>
</html>
```

Second exemple [voir]

```
<html>
 <head>
  <script language="javascript">
   var win = null;
   function fenetre(page, nom, L, H, defilement)
   {
    posX = (screen.width) ? (screen.width - L)/ 2 : 0;
    posY = (screen.height) ? (screen.height - H)/ 2 : 0;
    contenu = 'height=' + H + ', width = ' + L
            + ', top ='+ posY +', left =' + posX
            + ',  scrollbars = ' + scroll + ', resizable';
    win = window.open(page, nom, contenu);
   }
  </script>
 </head>
 <body>
  <a href=".."
    onclick="fenetre(this.href,'name','400','400','yes');
    return false">
   http://laltruiste.com
  </a>
 </body>
</html>
```

En savoir plus :

Netscape Microsoft

9 / Les objets DHTML pour Internet Explorer

A - B - C - D - E - F - G - H - I - J - K - L - M -
N - O - P - Q - R - S - T - U - V - W - X - Y - Z

Objet	Description
a	désigne un lien hypertexte.
acronym	indique une abbréviation acronyme.
address	désigne une adresse.
applet	insère un applet Java dans un document HTML.
area	définit une zone cliquable dans une image.map.
attribute	représente un attribut ou une propriété d'un élément HTMl ou d'un objet.
b	permet d'afficher un texte en gras.
base	spécifie une adresse URL explicite utilisé pour résoudre des liens et des références à des sources externes comme des images ou des feuilles de style.
baseFont	applique une police par défaut pour l'affichage du texte.
bdo	autorise la désactivation de l'algorithme bidirectionnel pour la sélection d'une partie de texte.
bgSound	active un son à partir de l'ouverture d'un document HTML.
big	affiche un texte dans une taille de police plus grande que celle par défaut.
blockQuote	formate un texte sous forme de citation.
body	correspond au corps du document.
br	insère un saut de ligne.
button	crée un bouton.
caption	indique une brève description pour un tableau.
center	permet de centrer des éléments HTML dans un document HTML.
cite	affiche un texte sous forme de citation.
clientInformation	contient des informations relatives au navigateur.
clipboardData	fournit un accès à des données stockées dans le presse-papier pour utiliser dans des opérations d'éditions (copier/coller).
code	formate un texte sous forme de code.
col	permet d'appliquer des propriétés à une colonne dans un tableau.
colGroup	permet d'appliquer des propriétés à un groupe de colonne dans un tableau.
comment	spécifie un commentaire qui n'est pas affiché.
currentStyle	représente des formats et styles en cascade d'un objet.
custom	représente un élément utilisateur défini.
dataTransfer	fournit un accès à des données stockées dans le presse-papier pour utiliser dans des opérations de glisser-déplacer (drag-and-drop).

dd	formate un item dans une liste en définition.
defaults	applique des propriétés par défaut à un comportement d'élément.
del	spécifie un texte qui a été supprimé d'un document.
dfn	formate un texte en définition.
dir	montre une liste de répertoire.
div	représente une division dans un document HTML.
dl	montre une liste de définition.
document	représente un document HTML dans un navigateur donné.
dt	indique une définition créé dans une liste de définitions.
em	permet d'accentuer un texte.
embed	permet d'insérer des éléments de plusieurs types dans un document.
event	représente les états d'un événement.
external	autorise l'accès à un modèle d'objet produit par les applications hôtes des composants du navigateur Microsoft® Internet Explorer.
fieldSet	desssine une boîte autour d'un ensemble d'éléments HTML.
font	applique une police à un texte.
form	indique que les éléments qu'il encadre appartiennent à un formulaire.
frame	crée un cadre (FRAME) à l'intérieur d'un jeu de cadres (frameset)
frameSet	crée un ensemble de cadres.
head	fournit plusieurs informations relatives au document à l'intérieur d'un en-tête (HEAD).
history	contient des informations à propos des adresses URL visitées par le client.
hn	applique un format de titre au texte.
hr	dessine une ligne horizontale.
html	identifie le document comme contenant d'éléments HTML.
HTML Comment	crée un commentaire qui ne sera pas affiché dans le navigateur.
i	affiche un texte en italique.
iframe	crée un cadre flottant dans un document HTML.
img	insère une image ou une vidéo dans un document.
implementation	contient des informations à propos des modules supportés par l'objet.
IMPORT	importe une définition de balise d'un comportement d'élément.
input	crée une variété de champs INPUT pour un formulaire.
input type=button	crée un bouton.
input type=checkbox	crée une case à cocher.
input type=file	crée un objet de téléchargement de fichier avec un champ texte et un bouton.

input type=hidden	crée un champ texte invisible transmis automatiquement lors de la soumission du formulaire.
input type=image	crée un bouton sous forme d'image.
input type=password	crée un champ de saisie de mot de passe dont les caractères affichés sont remplacés par des étoiles (*).
input type=radio	crée une case radio.
input type=reset	crée un bouton de réinitialisation du formulaire.
input type=submit	crée un bouton de soumission du formulaire.
input type=text	crée un champ de saisie de texte.
ins	spécifie qu'un texte a été inséré dans un document.
isIndex	affiche une feêtre de dialoque invitant l'utilisateur à saisir un texte.
kbd	affiche un texte avec une police de largeur fixe.
label	spécifie un label pour un élément sur la page.
legend	insère une légende pour un jeu d'éléments d'un formulaire (fieldSet).
li	indique un item dans une liste.
link	active le document courant à établir les liens avec des documents externes.
listing	affiche un texte avec une police de largeur fixe.
location	contient des information à propos de l'édresse URL courante.
map	contient les coordonnées d'une image divisée en zone cliquable (area).
marquee	crée une zone de texte défilante.
menu	crée un menu à partir d'items d'une liste.
meta	contient diverses informations cachées à propos du document, du serveur, du client, etc...
namespace	Importe dynamiquement un comportement d'élément dans un document.
navigator	contient des informations relatives au navigateur.
nextID	crée un identifiant unique que le logiciel d'édition de textes peut lire.
noBR	empêche des sauts de ligne.
noFrames	contient des éléments HTML à l'intérieur d'un ensemble de cadres (**frameset**).
noScript	spécifie une zone d'affichage d'éléments HTML pour un navigateur ne supportant pas les scripts.
object	insère un objet dans une page HTML.
ol	crée une liste numérotée.
optGroup	permet à des auteurs de grouper des choix logiquement dans un élément select.
option	montre un choix dans un élément select.
p	indique un paragraphe.
page	représente une règle @page à l'intérieur d'une feuille de style (styleSheet).

param	établit une valeur initiale d'une propriété pour un élément applet, embed, ou object.
plainText	affiche le texte dans une police de largeur fixe sans traiter des étiquettes.
popup	représente un type spécial de fenêtre superposé typiquement utilisé pour des zones de dialogue, des cadres de message, et d'autres fenêtres provisoires qui apparaissent séparées de la fenêtre principale d'une application.
pre	formate un texte tel qu'il est écrit dans l'éditeur HTML.
q	montre séparément une citation dans un texte.
rt	désignes le texte rouge pour l'élément RUBY.
ruby	désigne une annotation ou un guide de prononciation.
rule	représente un style dans une feuille de style CSS (Cascading Style Sheets) qui se compose d'un sélecteur et d'une ou plusieurs déclarations.
runtimeStyle	Représente le style et le modèle en cascade de l'objet qui ignore le style et le modèle indiqués dans les feuilles de styles globales, les modèles intégrés, et les attributs HTML.
s	affiche un texte barré.
samp	indique un exemple de code.
screen	contient des informations relatives à l'écran du client et ses capacités d'affichage.
script	indique un script de programmation.
select	montre une boîte de liste ou un boîte de liste déroulante.
selection	représente Représente la sélection active, qui est un bloc de texte mis en valeur, et/ou d'autres éléments dans le document sur lequel un utilisateur ou une script type peut effectuer une certaine action.
small	affiche un texte en petite police.
span	spécifie un contenant de texte.
strike	affiche un texte barré.
strong	affiche un texte en gras.
style	représente les paramètres courants de tout les styles en lignes possible pour un élément donné.
style	indique une feuille de style pour la page.
styleSheet	représente une unique feuille de style dans le document.
sub	affiche le texte en indice.
sup	affiche le texte en exposant.
table	crée la base d'un tableau formé de cellules et de colonnes.
tBody	désignes des lignes formant le corps du tableau.
td	crée une cellule d'un tableau.
textArea	crée un champ de saisie multiligne.
TextNode	représente une chaîne de caractères de texte comme un noeud dans la hiérarchie du document.
TextRange	représente le texte dans un élément HTML.

TextRectangle	spécifie un rectange qui contient une ligne de texte dans un élément ou un objet TextRange.
tFoot	désignes des lignes formant le pied du tableau.
th	spécifie une cellule d'en-tête.
tHead	désigne des lignes formant la tête du tableau.
title	contient le titre d'un document
tr	spécifie une ligne d'un tableau
tt	affiche le texte avec un largeur de police fixe.
u	affiche le texte en souligné.
ul	crée une liste à puce.
userProfile	fournit les méthodes qui permettent à une script type de demander l'accès en lecture et d'exécuter des actions de lecture sur une information de profil d'utilisateur.
var	définit une variable de programmation.
wbr	insère un saut de ligne si nécessaireà l'intérieur d'un bloc de texte encadré par nobr.
window	contient des informations relatives à la fenêtre courante dans un navigateur.
xml	définit un XML data island (ilôt de données XML) sur une page HTML.
xmp	affiche le texte utilisé comme exemple dans une largeur de police fixe.

10 / Les objets Javascript pour Netscape

A - B - C - D - E - F - G - H - I - J - K - L - M -
N - O - P - Q - R - S - T - U - V - W - X - Y - Z

Objet	Description
Anchor	symbolise une ancre ().
Applet	insère un applet Java (<applet>).
Area	crée une zone cliquable à partir d'une image en coordonnées (<area...>).
Array	crée un objet tableau.
Boolean	crée un objet booléén.
Button	crée un bouton (<button...>).
Checkbox	crée une case à cocher (<input type="checkbox").
Date	crée un objet date.
document	contient des informations relatives à la page HTML courante.
event	représente les états d'un événement.
FileUpload	crée un champ de saisie et un bouton pour le téléchargement de fichier (<input type="file">).
Form	établit les bases d'un formulaire (<form>...</form>).
Frame	crée un cadre (<frame>)dans un ensemble de cadres (<frameset>)
Function	spécifie une fonction contenant du code Javascript.
Hidden	crée un champ texte caché dans un formulaire (<input type="hidden">).
History	contient des informations à propos des adresses URL visitées par le client.
Image	insère une image dans un document HTML (<img>).
java	est un objet de niveau supérieur accédant à n'importe quelle classe de Java dans le module Java.*.
JavaArray	représente un tableau de module Java.
JavaClass	est une référence Javascript à une classe Java.
JavaObject	représente le type d'un objet Java enveloppé ayant accès de l'intérieur du code JavaScript.
JavaPackage	est une référence Javascript à un paquetage Java.
Layer	crée une couche dans un document HTML (<layer>).
Link	crée un lien (...).
Location	contient des informations relatives à l'adresse URL courante.
Math	est un objet programmé contenant des propriétés et des méthodes mathématiques.
MimeType	contient des informations relatives aux types MIME (Multipart Internet Mail Extension) du navigateur du client.
navigator	regroupe des informations à propos du navigateur client.

netscape	est un objet de niveau supérieur utilisé pour l'accès à n'importe quelle classe Java dans le paquetage netscape.*.
Number	permet de prendre en charge les nombres.
Object	représente la superclasse qui contient des caractéristiques propres à tous les objets Javascript.
Option	constitue une option dans un élément Select.
Packages	représente un objet au plus haut niveau qui a accès aux classes de Java de l'intérieur le code de JavaScript.
Password	crée un champ de saisie de mot de passe (<input type="password">).
Plugin	décrit les extensions logicielles installées sur le navigateur client.
Radio	crée une case radio (<input type="radio">).
RegExp	représente une expression régulière utilisée pour la mise en correspondance de modèles.
Reset	crée un bouton de réinitialisation (<input type="reset">).
screen	contient des informations relatives à la résolution d'écran du client.
Select	crée une liste de choix dans un formulaire (<select>).
String	représente un objet chaîne de caractères.
Style	représente les paramètres courants de tous les styles en ligne possible pour un objet donné.
Submit	réalise un bouton de soumission (input type="submit">).
sun	est un objet de niveau supérieur utilisé pour l'accès à n'importe quelle classe Java dans le paquetage sun.*.
Text	crée un champ de saisie (<input type="text">).
Textarea	réalise une zone de texte multiligne (<textarea>).
window	contient des informations relatives à la fenêtre affichée.

11 / Les événements

Les événements Javascript se produisent lors d'actions diverses (clic ou survol de souris, focus, chargement, etc.) sur les objets d'un document HTML.

Dans Javascript, tout un arsenal de commandes permet de capter ces événements et d'en faire usage dans ces scripts.

Le langage Javascript prend en charge plusieurs types d'événements. Ces derniers permettent aux objets d'apporter aux programmeurs diverses fonctionnalités rendant une page Web parfaîtement interactive ou adaptée aux besoins ou aux spécificités d'une application.

A - B - C - D - E - F - G - H - I - J - K - L - M -
N - O - P - Q - R - S - T - U - V - W - X - Y - Z

Evénement	Description	Syntaxe	Domaine
onAbort	Cet événement se produit lorsque l'utilisateur annule le chargement d'une image lorsqu'il active un lien ou appuie sur le bouton *Stop* du navigateur au cours du chargement de l'image.	onabort="Instruction"	Image
onafterprint	Cet événement se produit immédiatement après que les documents qui lui sont associés soient imprimés.	onafterprint ="Instruction"	
onafterupdate	Cet événement se produit immédiatement après qu'un objet contenant des données ait effectué la mise à jour de données.	onafterupdate ="Instruction"	
onbeforecopy	Cet événement se produit avant que la sélection ne soit copiée dans le presse-papiers.	onbeforecopy ="Instruction"	
onbeforecut	Cet événement se produit avant que la sélection ne soit coupée du document.	onbeforecut ="Instruction"	
onbeforeeditfocus	Cet événement se produit avant qu'un contrôle insère un changement au niveau de l'interface utilisateur.	onbeforeeditfocus ="Instruction"	
onbeforepaste	Cet événement se produit avant que la sélection ne soit collée du presse-papiers vers le document.	onbeforepaste ="Instruction"	
onbeforeprint	Cet événement se produit avant que les documents qui lui sont associés soient imprimés.	onbeforeprint ="Instruction"	
onbeforeunload	L'événement est activé avant qu'une page ne soit retirée du cadre.	onbeforeunload ="Instruction"	
onbeforeupdate	Cet événement se produit avant qu'un objet contenant des données n'effectue la mise à jour des données sur l'objet source.	onbeforeupdate ="Instruction"	

onblur	Cet événement se produit lorsque l'utilisateur sort d'un champ de formulaire HTML, d'un cadre ou d'une fenêtre (window), en appuyant sur la touche TAB du clavier ou en utilisant la souris. La fonction appelée peut servir à soumettre ou valider les informations saisies par l'utilisateur.	onBlur="Instruction"	Button Checkbox FileUpload Layer Password Radio Reset Select Submit Text Textarea window
onbounce	Cet événement est activé lorsque la valeur de l'attribut *BEHAVIOR* de l'élément *<MARQUEE>* est *ALTERNATE* et que le contenu de l'élément atteint l'une des limites de la zone de défilement.	onbounce ="Instruction"	
oncellchange	Cet événement est activé lorsque les données sont modifiées.	oncellchange ="Instruction"	
onchange	Cet événement se produit lorsque l'utilisateur modifie la valeur d'une boîte liste, d'une boîte à liste déroulante, d'une boîte texte ou d'une boîte texte multiligne. La fonction appelée peut servir à valider ou soumettre les informations entrées dans les champs d'un formulaire.	onchange="Instruction"	FileUpload Select Text Textarea
onclick	Cet événement se produit lorsque l'utilisateur enfonce puis relâche le bouton de la souris sur un bouton, une option dans un formulaire, un hyperlien ou sur le document.	onclick="Instruction"	Button Checkbox document Link Radio Reset Submit
oncontextmenu	Cet événement est activé lorsque l'utilisateur presse le bouton droit de la souris sur l'élément, activant ainsi un menu contextuel.	oncontextmenu ="Instruction"	
oncopy	Cet événement est activé lorsque l'utilisateur copie l'objet ou qu'une sélection est copiée dans le presse-papiers.	oncopy ="Instruction"	
oncut	Cet événement est activé lorsque l'objet ou la sélection est coupé du document et ajouté au presse-papiers.	oncut ="Instruction"	
ondataavailable	Cet événement est activé à chaque fois que des données transmises par un objet source sont reçues sur le système.	ondataavailable ="Instruction"	
ondatasetchange	L'événement ondatasetchange est activé lorsque les données exposées à un objet source sont modifiées.	ondatasetchange ="Instruction"	
ondatasetcomplete	Cet événement est activé pour indiquer que toutes les données de l'objet source sont	ondatasetcomplete ="Instruction"	

	disponibles.		
ondblclick	Cet événement se produit lors d'un double-clic suivi d'un relâchement sur un élément de formulaire ou un hyperlien.	ondblclick="Instruction"	Area document Link
ondrag	Cet événement se produit lorsqu'une opération *glisser-déplacer* est effectuée sur l'élément avec la souris.	ondrag ="Instruction"	
ondragdrop	Cet événement se produit lorsque l'utilisateur dépose un objet dans la fenêtre du navigateur à l'aide de la souris.	ondragdrop="Instruction"	window
ondragend	Cet événement se produit à la fin d'une opération *glisser-déplacer*.	ondragend ="Instruction"	
ondragenter	Cet événement est activé lorsque l'utilisateur glisse l'élément sur une cible valide lors d'un *glisser-déplacer* effectuée avec la souris.	ondragenter ="Instruction"	
ondragleave	Cet événement est activé lorsque l'utilisateur glisse l'élément hors d'une cible valide lors d'un *glisser-déplacer* effectuée avec la souris.	ondragleave ="Instruction"	
ondragover	Cet événement se produit lorsque l'utilisateur glisse l'élément et passe sur une cible valide lors d'un *glisser-déplacer* effectuée avec la souris.	ondragover ="Instruction"	
ondragstart	Cet événement se produit lorsque l'utilisateur entame une opération *glisser-déplacer* avec l'élément.	ondragstart ="Instruction"	
ondrop	Cet événement agit sur l'objet cible lorsque l'utilisateur relâche le bouton de la souris après une opération de *glisser-déplacer*.	ondrop ="Instruction"	
onerror	Cet événement se produit lorsqu'il y a une erreur au cours du chargement d'une image ou d'un document.	onerror="Instruction"	Image et window
onerrorupdate	Cet événement se produit lorsqu'une erreur survient lors de la mise à jour des données associées à l'objet source.	onerrorupdate ="Instruction"	
onfilterchange	Cet événement se produit lorsqu'un filtre visuel change d'état ou complète une transition.	onfilterchange ="Instruction"	
onfinish	Cet événement se produit lorsque l'animation de l'élément *<MARQUEE>* se termine.	onfinish ="Instruction"	
onfocus	Cet événement se produit lorsqu'un élément (champ de formulaire, cadre, layer, etc.) ou un objet (window) obtient le focus. Le focus peut être	onfocus="Instruction"	Button Checkbox FileUpload Layer Password Radio

	obtenu par l'intermédiaire de la souris, de la touche *TAB* du clavier ou par une méthode de focus.		Reset Select Submit Text Textarea window
onhelp	Cet événement se produit lorsque l'utilisateur presse la touche F1 sur le clavier et que la fenêtre active est celle du Navigateur Web.	onhelp ="Instruction"	
onkeydown	Cet événement se produit au moment où l'utilisateur enfonce une touche du clavier.	onkeydown="Instruction"	document Image Link Textarea
onkeypress	Cet événement se produit au moment où l'utilisateur relâche la touche qu'il venait d'enfoncer.	onkeypress="Instruction"	document Image Link Textarea
onkeyup	Cet événement se produit au moment où l'utilisateur relâche une touche du clavier. Cependant, beaucoup de précautions doivent être prises concernant l'utilisation de cet événement, étant donné la confusion qu'il risque d'exister avec l'événement *onKeyPress*.	onkeyup="Instruction"	document Image Link Textarea
onload	Cet événement se produit lorsque le navigateur a terminé le chargement d'une page HTML, de tous les cadres (*frameS*) de l'élément *<frameset>* ou d'une image.	onload="Instruction"	Image Layer window
onlosecapture	Cet événement se produit lorsque l'élément perd le focus de la souris.	onlosecapture ="Instruction"	
onmousedown	Cet événement se produit lorsque l'utilisateur enfonce un des boutons de la souris.	onmousedown="Instruction"	Button document Link
onmousemove	Cet événement se produit lorsque le pointeur de la souris est déplacé et passe sur l'objet.	onmousemove="Instruction"	
onmouseout	Cet événement se produit à chaque fois que l'utilisateur déplace le pointeur de la souris hors d'une zone contenant un hyperlien (images en coordonnées, lien hypertexte).	onmouseout="Instruction"	&Area Layer Link
onmouseover	Cet événement se produit à chaque fois que le pointeur de la souris entre dans une zone contenant un hyperlien (lien hypertexte ou images en coordonnées).	onmouseover="Instruction"	Area Layer Link
onmouseup	Cet événement se produit lorsque l'utilisateur relâche le bouton de la souris.	onmouseup="Instruction"	Button document Link
onmove	Cet événement se produit lorsque l'utilisateur ou un script déplace une fenêtre ou un	onmove="Instruction"	window

	cadre.		
onpaste	Cet événement se produit lorsque l'utilisateur " colle " des données du presse-papiers vers le document.	onpaste ="Instruction"	
onpropertychange	Cet événement se produit lorsque la propriété de l'élément change.	onpropertychange ="Instruction"	
onreadystatechange	Cet événement se produit lorsque l'état de l'élément a changé.	onreadystatechange ="Instruction"	
onreset	Cet événement se produit lorsque l'utilisateur active le bouton de type reset d'un formulaire.	onreset="Instruction"	Form
onresize	Cet événement se produit lorsque l'utilisateur modifie les dimensions de la fenêtre du navigateur.	onresize="Instruction"	window
onrowenter	Cet événement se produit lorsque la rangée courante a changé dans la source de données et que de nouvelles données sont disponibles.	onrowenter ="Instruction"	
onrowexit	Cet événement se produit juste avant que le contrôle de la source de données ne modifie la rangée courante de l'objet.	onrowexit ="Instruction"	
onrowsdelete	Cet événement se produit juste avant la suppression de rangées au jeu d'enregistrements.	onrowsdelete ="Instruction"	
onrowsinserted	Cet événement se produit après l'insertion de nouvelles rangées dans le jeu d'enregistrements.	onrowsinserted ="Instruction"	
onscroll	Cet événement se produit lorsque l'utilisateur déplace le curseur de la barre de défilement de l'objet.	onscroll ="Instruction"	
onselect	Cet événement se produit lorsque l'utilisateur sélectionne du texte dans une boîte texte ou dans une boîte texte multiligne.	onselect="Instruction"	Text Textarea
onselectstart	Cet événement se produit dès que l'objet est sélectionné.	onselectstart ="Instruction"	
onstart	Cet événement se produit à chacune des reprises de l'animation de l'élément *<MARQUEE>*.	onstart ="Instruction"	
onstop	Cet événement se produit lorsque l'utilisateur clique sur le bouton *Stop* ou *Arrêter* du navigateur.	onstop ="Instruction"	
onsubmit	Cet événement se produit lorsque l'utilisateur envoie les données d'un formulaire.	onsubmit="Instruction"	Form

onunload	Cet événement se produit au moment où le document HTML courant est retiré de la fenêtre au profit d'un autre ou lorsque le logiciel de navigation est fermé par l'utilisateur.	onunload="Instruction"	window

12 / Les chaînes de requêtes

Les chaînes de requêtes (QueryStrings) correspondent à des informations associées à l'adresse URL avec un point d'interrrogation comme séparateur.

```
http://www.laltruiste.com/page.html?chaine_de_requete
```

Une chaîne de requête doit être constituée d'un ou plusieurs éléments dont chacun est associé à une valeur. Si la chaîne de requête contient plusieurs éléments, alors **chaque couple élément/valeur doit être séparé par un caractère esperluette** (&).

```
?element=valeur&element2=valeur2&...&elementN=valeurN
```

L'inconvénient principal des chaînes de requêtes réside dans le fait que **les données transmises au serveur par ce biais sont visibles par les utilisateurs dans le champ *Adresse* de leur navigateur**.

De plus, **une chaîne de requêtes ne peut dépasser le maximum de 255 caractères**.

Ainsi, l'utilisation de cette technique peut générer des **problèmes de sécurité** et des **limitations quant à la taille des données à transmettre** à une application Web.

Il existe deux méthodes pour l'utilisation des chaînes de requêtes :

- **L'insertion des informations directement après l'adresse URL de la page à atteindre.**

```
<form action="page.html?element=valeur"
        method="post">
  <input type="text" name="nom" value="DUPUIS">
  <input type="text" name="prenom" value="Michel">
  <input type="submit" name="Soumettre" value="Soumission">
</form>
```

- **L'application de la valeur *GET* à l'attribut *method* d'un formulaire.**

```
<form action="page.html" method="get">
  <input type="text" name="nom" value="DUPUIS">
  <input type="text" name="prenom" value="Michel">
  <input type="submit" name="Soumettre" value="Soumission">
</form>
```

Dans le premier cas, **seules les informations contenues dans l'adresse indiquée par l'attribut *action*, seront envoyées** par l'intermédiaire de la collection *QueryString*.

```
element=valeur
```

Dans le second cas, **tous les éléments du formulaire seront transmis au moyen de la collection *QueryString***, à la page destinataire soit *page.html*.

```
nom=DUPUIS&prenom=Michel
```

En conséquence, **l'expression *method="get"* provoque la transmission complète et automatique d'un formulaire à partir d'une chaîne de requête**.

La propriété *search* de l'objet *location* permet d'extraire une chaîne de requêtes contenu dans l'adresse URL transmise, soit l'ensemble des éléments et leur valeur associée y compris celle du bouton de soumission et le fameux point d'interrogation introduisant la chaîne de requête.

```
document.write (window.location.search);
// Retourne
?nom=DUPUIS&prenom=Michel&Soumettre=Soumission
```

Le point d'interrogation peut être supprimé afin de ne conserver que la chaîne de requête à proprement parler.

```
var requete = window.location.search.subString(1);
```

Ensuite, chaque paire clé/valeur de la chaîne de requête peuvent être dissocier par l'intermédiaire de la méthode *split()*.

```
var elements = requete.split("&");
```

Chaques paires clé/valeur désormais stockées dans un tableau peuvent être exploités pour récupérer séparément les identificateurs, des valeurs.

```
for(i = 0; i < elements.length; i++) {
  temp = elements[i].split("=");
  tab[i][0] = temp[0];
  tab[i][1] = unescape(temp[1]);
}
```

A partir de ce tableau associatif, une simple boucle va permettre de parcourir l'ensemble des éléments.

```
document.write("<ol>");
for(i = 0; i < tab.length; i++) {
  document.write("<li>" + tab[i][0] + " : " + tab[i][1] + "</li>");
}
document.write("<ol>");
// affiche
<ol>
  <li>nom : DUPUIS</li>
  <li>prenom : Michel</li>
  <li>Soumettre : Soumission</li>
</ol>
```

Exemple [voir]

```
<!-- Formulaire -->
<html>
 <body>
  <form
      action="traitement.html"
      method="get"
      name="formGet">

    <u>Saisir un titre :</u><br>
    <input type="text" name="Titre" value="Fatrasie" size="20"><br>

    <textarea name="Paragraphe" cols="30" rows="4">
     La chose va très mal
     Où point n'a de justice
     La chose va très mal
     Dit un veau de métal
    </textarea>

    <input type="submit" name="Soumettre" value="Soumission">

  </form>

 </body>
</html>

<html>
 <body>
  <h2>La chaîne de requête</h2>
  <script lan language="JavaScript" type="text/javascript">
   var tab_args = new Array();
   var temp = new Array();

   var requete = location.search.substring(1);

   document.write("<h4>" + requete + "</h4>");

   var tab_paires = requete.split("&");

   for(var i = 0; i < tab_paires.length; i++) {
    temp = pairs[i].split("=");
    tab_elts[i] = new Array(temp[0], unescape(temp[1]));
   }

   for(i = 0; i < tab_elts.length; i++) {
    document.write(tab_elts[i][0] + " : "
              + tab_elts[i][1] + "<br>");
   }
  </script>
 </body>
</html>
```