JavaScript : Les bases du langage

Ce document permet de connaître l'intégration des scripts dans un document HTML et de découvrir les types de données, les structures de contrôle ainsi que les définitions et invocations de fonction. La plupart des concepts de ce document relèvent des connaissances de bases de tout langage. Le contenu du document permet donc d'actualiser ces concepts à l'environnement des scripts clients Javascript.

1. Introduction à JavaScript

Caractéristiques

Le langage s'apparente au langage Java pour ses particularités lexicales. Il est toutefois bien distinct de Java quant à son typage. Java est fortement typé, de son côté JavaScript est très peu typé.

Ses actions sont principalement :

- contrôle de l'apparence et du contenu de document
- contrôle du navigateur
- interaction avec les formulaires HTML
- interaction avec les utilisateurs
- lecture et écriture de cookies

Sécurité

JavaScript est limité dans ses actions possibles. Voici ce qu'il **ne peut pas** faire :

- Lecture et écriture de fichiers.
- Prendre en charge des tâches de gestion de réseau[1].

Emplacement

Les scripts réside du côté client dans l'architecture d'un système. Ils sont inclus dans un document HTML ou bien ils peuvent aussi constituer un document tel quel (avec l'extension .js).

[1] Il peut par contre rediriger les navigateurs vers d'autres URL.

2. Exploration du noyau de JavaScript

Structure lexicale

Depuis la version 1.2, le langage possède ces particularités :

- encodage de texte *unicode* sur 16 bites
- sensible à la casse
- ignore les espaces, tabulation et retour à la ligne
- reconnaît les opérations post-fixées (variable++)
- permet deux styles de commentaires : //…… et /* …… */

Types de données

Parmi les types de données reconnus par le langage, les plus courants sont :

- les nombres entiers décimaux, hexadécimaux, octaux et les nombres points flottants
- les chaînes de caractères et les séquences d'échappement
- les booléens
- les fonctions et fonctions littérales
- les objets
- les tableaux
- la valeur *null*
- le type Date

Les variables

JavaScript est un langage faiblement typé. Si faiblement qu'une variable peut recevoir comme valeur plusieurs types de valeur un à la suite de l'autre. Par exemple, la variable « a » pourrait être traitée comme suit :

```
var a; //déclaration
a = 12; //type entier
a = « douze »; //type String
```

Ce traitement est valable et ne génère aucune erreur.

La déclaration de variable peut se faire de différente façon :

```
var i;
var somme;
var somme = 99;
var i = 1, j = 2, k = 3;
```

Dans le cas du premier exemple (var i;), une variable qui ne reçoit pas de valeur lors de son initialisation, sa valeur est en fait *undefined.*

La portée d'une variable, comme la plupart des langages, dépend d'où elle a été déclarée. Si elle est déclarée dans une fonction, sa portée est locale à la fonction. Si elle est déclarée dans une fonction dans un bloc d'exécution particulier (comme une boucle), sa portée n'est pas retenue dans le bloc, mais bien dans la fonction. Si elle est déclarée à l'extérieur des fonctions, sa portée est globale au bloc de script.

Les opérateurs

Voir tableau (HTML La référence; O'Reilly)

Une particularité existe avec la manipulation des chaînes de caractères. La concaténation se fait avec l'opérateur « + ». Étant donné le faible typage du langage, l'opérateur est polyvalent :

1 + 2	donne 3	
« a » + « b »	donne « ab »	
1 + « b »	donne « 1b »	(voir exemple concaténation.htm)

Les instructions

Parmi toutes les instructions utilisables, les plus courantes sont :

- if : if (var1 = = var2) { instruction; }
- else if (condition) { instruction; }
- switch (expression) { instruction; }

- while (expression) { instruction; }
- do { instruction } while (expression);
- for (i=x; i< y; i++) { instruction; }
- for (var in objet) { instruction; }
- continue : permet de reprendre une boucle après une instruction *break*.
- throw : soulève une exception.

- with (objet) { instruction; } : Donne accès aux membres de l'objet.

Les fonctions

Définition de fonctions:

```
function nom_de_la_fonction ([paramètres]) {
    instructions;
}
```

Exemple :

```
function diviser (numerateur,denominateur) {
    return numerateur/denominateur;
}
```

Invocation de fonctions :

```
nom_de_la_fonction (paramètres);
```

Exemple :

```
if (diviser(participants,places) = =  1) assezDePlace = true;
```

(voir exemple concatenation.htm)

3. Intégrer les scripts au code HTML

Le code JavaScript s'intègre de trois façons dans le code HTML. Soit dans l'en-tête, soit directement dans le code HTML ou soit dans un fichier inclus (extension .js).

Dans l'en-tête d'un document HTML

Le bloc de script est écrit entre les balises <head> et </head>.

Voir exemple concatenation.htm

Imbriqué dans le code HTML

Le code Javascript peut se retrouver à travers le code HTML en l'insérant à l'intérieur de balises<script></script>.

Voir l'exemple concatenation.htm

Dans un fichier séparé

Tout le code de script est écrit dans un fichier portant l'extension .js. Le document HTML contient une inclusion du fichier *Javascript*.

Voir exemple porteeVariable.htm

4. Objet String

Chaînes

En JavaScript, les caractères et les chaînes de caractères sont exprimés par même type : des chaînes de caractères. C'est donc dire qu'il n'y a pas de type caractère.

La chaîne ne peut être écrites sur des lignes consécutives. La seule façon d'inclure un saut de ligne dans une chaîne est d'utiliser un caractère d'échappement.

Les guillemets doubles et simples ont la même signification. Les guillemets simples peuvent êtres imbriqués dans la portée des guillemets doubles.

ex : onclick=''alert('allo')''

Constructeur

String possède deux constructeurs :

new String(s)	Construit une instance de String avec la chaîne s.
String(s)	Converti en une instance de String avec s.

Propriété

length	Donne le nombre de caractères de la chaîne.

```
Ex :    var S = '' allo'';
        var L = s.length;       // L aura la valeur 4.
```

Méthodes

charAt( )

string.charAt(n)

Retourne le n ième caractère de string.

Si n n'est pas dans la plage de position couverte par la chaîne, la méthode retourne une chaîne vide.

indexOf( )

string.indexOf(sous-chaîne) ou string.indexOf(sous-chaîne, début)

Retourne la position de la première occurrence de la sous-chaîne après la position début. Retourne –1 si aucune n'est occurrence trouvée.

lastIndexOf( )

string.lastIndexOf (sous-chaîne) ou string.lastIndexOf (sous-chaîne, début)

Retourne la position de la dernière occurrence de la sous-chaîne avant la position début. Retourne –1 si aucune occurrence n'est trouvée.

slice( )

string.slice(début, fin)

Retourne une nouvelle chaîne contenant tous les caractères de string de l'index de chaîne début jusqu'à l'index de chaîne fin non compris.

split( )

string.split(délimiteur, limite)

Retourne un tableau dont chaque position contient une chaîne provenant de string découpé par délimiteur. La longueur du tableau retourné aura un nombre d'éléments égaux à limite.

Les arguments de valeurs négatives spécifient une position mesurée à partir de la fin de la chaîne.

substr( )

string.substr(début, longueur)

Retourne une copie de la sous-chaîne commençant à la position début et de longueur *longueur*.

substring( )

string.substring(de, à)

Retourne une nouvelle chaîne extraite de string dont le début commence à la position « de » et se termine à la position « à-1 ».

toLowerCase( )

string.toLowerCase( )

Retourne une copie string convertie en minuscules.

toUpperCase( )

string.toUpperCase( )

Retourne une copie de string convertie en majuscules.

toString( )

string.toString( )

Retourne la valeur primaire de la chaîne.

valueOf( )

idem

5. Objet Array

Javascript possède un type tableau. Ce type est en fait un objet possédant plusieurs méthodes utiles à la manipulation de son contenu. Le type défini un tableau à une dimension. Les éléments peuvent êtres de tous types pour un même tableau, y compris le type *Array*. Un tableau contenant des éléments de type tableau est en fait un tableau en deux dimensions.

Le tableau se déclare avec un constructeur ou sous forme de tableau littéral.
Ex : var t = [12, « texte », [7,8], 36]

Constructeur

new Array()
new Array(taille)
new Array(élément 0, élément 1, …. , élément n)

Le premier constructeur crée une instance de tableau de longueur 0. Le deuxième constructeur crée une instance de longueur *taille* ne contenant aucun élément. Le troisième constructeur crée une instance contenant *l'élément n* et est de taille[2] équivalente au nombre d'élément.

Valeur de retour

Les constructeurs retournent le tableau créé et initialisé selon le constructeur invoqué.

Propriété

length : Nombre entier en lecture et écriture exprimant la longueur du tableau.

Méthodes

concat()

Retourne un nouveau tableau contenant les arguments utilisés.

array.concat(valeur 1, valeur 2)

valeur 1 et valeur 2 sont concaténées à array.

[2] La première position du tableau est 0.

join()

array.join()
array.join(séparateur)

Retourne une chaîne le contenu du tableau et l'affiche avec le *séparateur* utilisé. « , » est le séparateur par défaut.

length

array.length
array.length = 4

Donne la longueur de tableau. Si une valeur lui est affectée, la longueur du tableau devient la longueur donnée. Si une valeur inférieure à la taille du tableau est donnée, le tableau est coupé et les éléments coupés sont perdus.

pop()

array.pop()

Supprime et retourne le dernier élément du tableau.

push()

array.push(valeur, …)

Ajoute les valeurs données en argument, dans l'ordre donné, à la fin du tableau. Retourne la nouvelle longueur du tableau.

reverse()

array.reverse()

Inverse l'ordre des éléments du tableau. Si plusieurs références au tableau existent, l'ordre est inversé pour toutes les références. *Array* est directement modifié.

shift()

array.shift()

Supprime et retourne le premier élément du tableau.

slice()

array.slice(indice de début, indice de fin)

Retourne un nouveau tableau contenant les éléments de array de l'indice de début comprise jusqu'à l'indice de fin non comprise.

sort()

array.sort()
array.sort(fonction de tri)

Retourne une référence à array. Array est directement modifié. L'appel sans argument exécute un tri selon le code de caractère. La fonction de tri doit prendre 2 arguments en entré et retourne un nombre indiquant leur position.

Ex : function alloTri(a, b) {return a-b;}
// retourne une valeur négative si a < b
// retourne 0 si a = b
// retourne une valeur positive si a > b

splice()

array.splice(indice de début, nombre, valeur, valeur, …)

Supprime un *nombre* d'éléments à partir de *indice à début*. Si *nombre* n'est pas donné, les éléments de indice début jusqu'à la fin est supprimé. Si l'argument *valeur* est utilisé, la valeur est ajoutée à partir de *indice début*.

toString()

array.toString()

Retourne une représentation de *array* sous forme de chaîne.

unshift()

array.unshift(valeur, …)

Insère des éléments au début de *array* et retourne la nouvelle taille.

6. Objet Date

Ce document donne les instructions du noyau Javascript concernant l'objet Date. Soit les constructeurs, les propriétés et les méthodes.

Caractéristiques de date

La date est créée par défaut selon l'heure local et non selon le format universel (UTC ou GMT)[3].

Les constructeurs

new Date ()
: Crée une instance contenant la date de création de l'instance.

new Date (millisecondes)
: Date en millisecondes.

new Date (chaîne)
: Représentation littérale de la date selon le format accepté.

new Date (année, mois, jour, heures, minutes, secondes, ms)
: Champs individuels de la date. Année et mois sont obligatoires, les 5 suivants sont optionnels.

Arguments des constructeurs

Millisecondes :	Nombre de millisecondes depuis le 1er janvier 1970.
Chaîne :	Chaîne de caractères donnant la date selon le format accepté par Date.parse().
Année :	Année sur 4 chiffres.
Mois :	Mois donné avec un entier de 0 à 11.
Jours :	Entier de 1 à 31.
Heures :	Entier de 0 à 23.
Minutes :	Entier de 0 à 59.
Secondes :	Entier de 0 à 59.
Ms	Entier de 0 à 999.

[3] La méthode Date.UTC() permet de convertir au format universelle.

Méthodes

Les instances possèdent une panoplie de méthodes permettant d'obtenir (get___) ou de régler (set___) les dates. Voici la liste de ces méthodes. L'ajout de « UTC » au nom de la méthode donne un retour selon le format universel.

Les méthodes GET

getDate()	getDay()	getFullYear()	getHours()
getMilliseconds()	getMinutes()	getMonth()	getSeconds()
getTime()	getTimezoneOffset()	getUTCDate()	getUTCDay()
getUTCFullYear()	getUTCHours()	getUTCMilliseconds()	getUTCMinutes()
getUTCMonth()	getUTCSeconds()	getYear()	

Les méthodes SET

setDate()	setFullYear()	setHours()	setMilliseconds()
setMinutes()	setMonth()	setSeconds()	setTime()
setUTCDate()	setUTCFullYear()	setUTCHours()	setUTCMilliseconds()
setUTCMinutes()	setUTCMonth()	setUTCSeconds()	setYear()

Les méthodes de conversion

toDateString()	toGMTString()	toLocalDateString()
toLocalString()	toLocalTimeString()	toString()
toUTCString()	toTimeString()	

Méthodes statiques

Les méthodes statiques appartiennent directement à l'objet Date et sont invoqués par le constructeur.

Date.parse()

La méthode accepte en argument une représentation littérale d'une date et retourne une représentation interne en millisecondes de la date.

Date.UTC()

Le méthode accepte en argument une représentation littérale d'une date et retourne une représentation interne en millisecondes de la date universelle.

7. L'objet Math

Ce document décrit l'objet Math et donne les fonctions mathématiques et constantes disponibles en Javascript.

Description

Ce type est considéré comme un objet parce qu'il fournit des méthodes et des propriétés (constantes). Il ne doit toutefois pas être considéré comme une classe puisque aucun constructeur ne permet d'instancier une variable. Les méthodes sont toutes statiques, donc appelés par l'objet directement.

Utilisation

Synopsis

Math.constante	appelle une des constantes
Math.fonction()	appelle une des fonctions statiques

Constantes

Math.E	Logarithme naturel de *e*
Math.LN10	Logarithme naturel de 10
Math.LN2	Logarithme nature de 2
Math.LOG10E	Logarithme base 10 de *e*
Math.LOG2E	Logarithme base 2 de *e*
Math.PI	Constante Π
Math.SQRT1_2	1 divisé par la racine carrée de 2
Math.SQRT2	Racine carrée de 2

Fonctions statiques

Math.abs()	Calcule une valeur absolue
Math.acos()	Calcule un arc cosinus
Math.asin()	Calcule un arc sinus
Math.atan()	Calcule un arc tangente
Math.atan2()	Angle d'un point du cercle trigonométrique p/r à l'axe des X
Math.ceil()	Arrondit un nombre
Math.cos()	Cosinus
Math.exp()	Exposant de *e*
Math.floor()	Arrondit à la valeur inférieure
Math.log()	Logarithme naturel
Math.max()	Retourne le plus grand de deux nombres
Math.min()	Retourne le plus petit de deux nombres
Math.pow()	x^y
Math.random()	Retourne un nombre au hasard
Math.round()	Arrondi à l'entier le plus proche
Math.sin()	Calcule un sinus
Math.sqrt()	Racine carrée
Math.tan()	Calcule une tangente

Exemples d'utilisation

Dans le cas de constante, l'invocation de Math.nom_de_constante retourne directement la valeur.

Ex : aire = rayon * rayon * Math.PI

Les fonctions sont invoquées en passant la valeur à modifier en arguments de la méthode.

Ex : y = Math.sin(x)

Valeurs numériques spéciales

JavaScript transforme les valeurs dépassant les intervalles de nombres représentables en *Infinity*. Même chose lorsque qu'un opérateur de Math créer ou utilise une valeur qui n'est pas un nombre, il produit *NaN.*

Le tableau suivant regroupe les constantes numériques spéciales

Constante	Signification
Infinity	Valeur spéciale pour représenter l'infini.
NaN	Valeur spéciale non-un-nombre.
Number.MAX_VALUE	Le plus grand nombre représentable.
Number.MIN_VALUE	Le plus petit nombre, proche de zéro, représentable.
Number.NaN	Valeur spéciale non-un-nombre
Number.POSITIVE_INFINITY	Valeur spéciale pour représenter l'infini par une valeur positive.
Number.NEGATIVE_INFINITY	Valeur spéciale pour représenter l'infini par une valeur négative.

8. L'objet Error

Ce document décrit l'usage de l'objet Error de Javascript. Soit l'utilisation des instances prédéfinies et de la création de nouvelles instances.

Description

Les instances de la classe Error sont utilisées par Javascript dans les blocs *try-throw-catch-finally*. Le langage lève des exceptions tels que *TypeError* ou *RangeError*. Dans l'utilisation de script complexe, l'usage d'instance de *Error* particulières à la logique d'affaire aide à l'entretien du système.

Constructeurs

new Error() Crée un instance sans détail.
new Error(message) Crée un instance incluant des détails sur l'erreur.

Argument

Message : Message d'erreur affichable lorsque l'erreur est levée.

Propriétés

error.message Message affiché lorsque l'erreur est levée et décrivant l'erreur.

error.name Chaîne de caractères donnant le type de l'erreur.

Méthodes

error.toString()

Retourne une chaîne définie par l'implémentation qui représente l'objet invoqué.

Exemple d'utilisation

```
Function division (numérateur, dénominateur){
        Try {

                If (dénominateur = = 0) {
                        throw new Error(« division par zero : valeur infinie;
                }else {
                        return numérateur/dénominateur;
                }
        } catch (e) {
                return e.toString();
        }

}
```

Erreurs prédéfinies

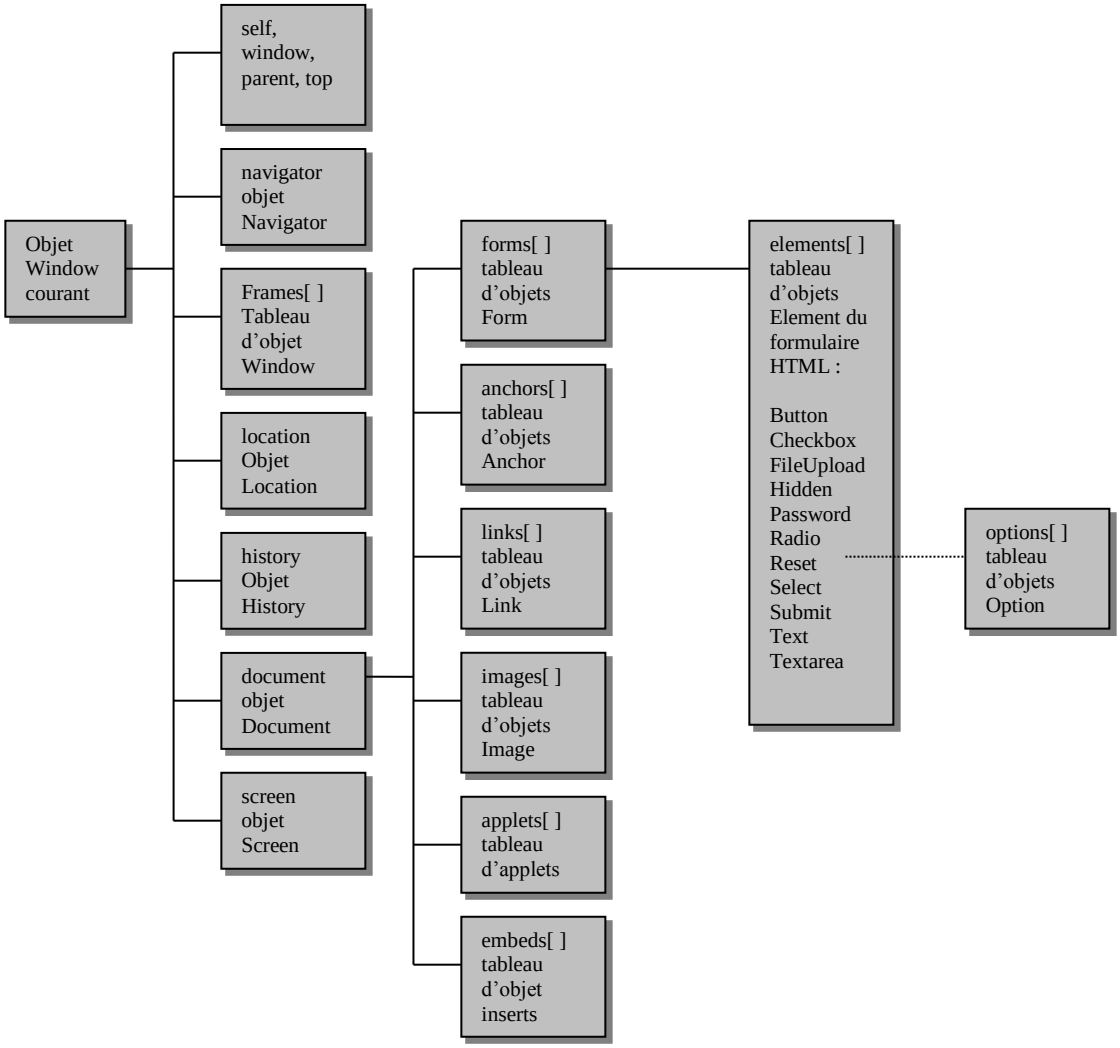
EvalError	Invoquée lorsque eval() est mal utilisé.
RangeError	Invoquée si un nombre ne fait pas partie d'une plage légale comme un intervalle de tableau par exemple.
ReferenceError	Lecture d'une variable qui n'existe pas.
SyntaxError	Lorsqu'une mauvaise syntaxe Javascript est utilisée.
TypeError	Une valeur est de type erroné.
URLError	Levée par decodeURL() et decodeURLComponent()

9. Utilisation du DOM : interaction avec l'interface HTML

Ce document décrit la structure du DOM niveau 0 (Document Object Model) et donne les propriétés et méthodes de quelques nœuds de cette structure. Plus particulièrement, les propriétés de Window, Document et Form.

Hiérarchie des objets côté client; niveau 0

L'objet Window en Javascript représente le contexte d'exécution global. Puisqu'il existe aussi une série d'objets exprimant les propriétés du navigateur et les éléments HTML d'un document, tous ces objets seront considérés comme des propriétés de Window. Ainsi une hiérarchie existe et elle se schématise de la façon suivante.



Accéder aux objets du modèle

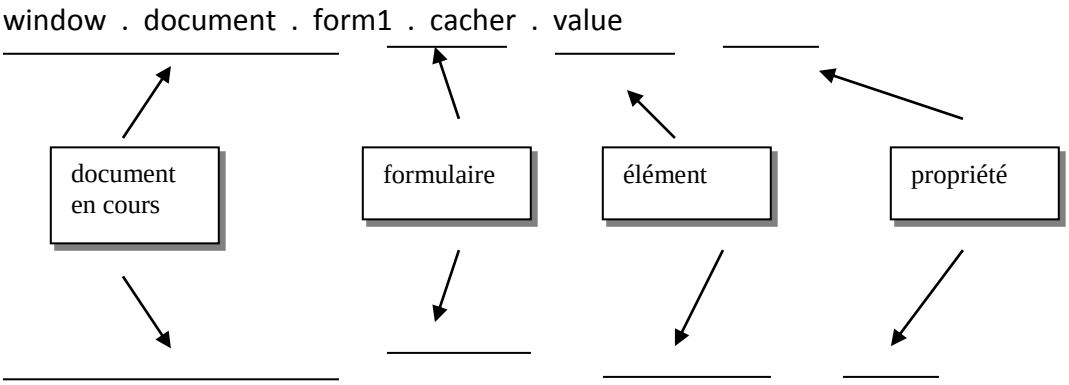
Les objets des documents HTML sont accessibles soit directement par leur nom, soit via les tableaux d'objets (conteneurs) construits par le navigateur. Ces tableaux contiennent les objets présents dans le document.

Soit le document HTML suivant :

```
<html>
<head>
<title>Titre</title>
</head>

<body>
        <form name="form1" method="post" action="">
                <input type="cacher" name="hiddenField">
        </form>
</body>
</html>
```

Les deux méthodes d'accès mènent au même résultat



Référence de l'objet Window

Propriétés

closed	Booléen : true si la fenêtre est fermée
defaultStatus, status	Texte de la ligne de statut du navigateur
document	Référence au document HTML
frames[ ]	Tableau d'objet Window de cadres, s'il y en a dans le document
history	Référence à l'historique de navigation
location	Référence à l'URL du document affiché
name	Nom de la fenêtre
opener	Référence à l'objet Window qui a ouvert la fenêtre courante
parent	Si la fenêtre est un cadre, référence au cadre qui la contient
self	synonyme de window
top	Si la fenêtre est un cadre, référence à l'objet Window au niveau le plus élevé contenant le cadre
window	Référence à l'objet Window courant

Méthodes

alert(), confirm(), prompt()	Boîtes de dialogue
close()	Ferme la fenêtre
focus(), blur()	Focus place la fenêtre en avant-plan, blur retire le focus.
moveBy(), moveTo()	Déplace la fenêtre
open()	Ouvre une nouvelle fenêtre avec un URL spécifié
print()	Imprime la fenêtre ou le cadre
resizeBy(), resizeTo()	Redimensionne la fenêtre
scrollBy(), scrollTo()	Fait défiler le document affiché dans la fenêtre
setInterval(), clearInterval()	Programme ou annule une fonction à invoquer de façon répétitive avec un délai spécifié en millisecondes

setTimeout(), clearTimeout()	Programme ou annule une fonction à invoquer après un délai spécifié en millisecondes

Référence de l'objet Document

Propriétés

anchors[ ]	Tableau des objets ancre dans le document.
alinkColor, linkColor, vlinkColor	Attributs alink, link et vlink du document HTML.
applets [ ]	Tableau des applets Java dans le document.
bgColor, fgColor	Couleur du fond et couleur du texte du document HTML.
cookie	Cookies http.
domain	Permet au serveur d'assouplir les interactions entre pages de même domaine.
forms[ ]	Tableau d'objets Form.
images [ ]	Tableau d'objets Image.
lastModified	Chaîne contenant la date de dernière modification du document.
links[ ]	Tableau d'objets Link.
location	Propriété URL (déclassée)
referrer	URL du document qui contient le lien qui a conduit au document présent. (Fenêtre qui a commandé l'ouverture de la fenêtre mise en focus)
title	Texte du conteneur <title>.
URL	URL du document.

Méthodes

close()	Ferme un document qui a été ouvert avec open().
open()	Ouvre un nouveau document effaçant le contenu du document en cours.
write()	Ajoute du texte au document actuellement ouvert.
writeln()	Ajoute du texte et un retour à la ligne.

Référence de l'objet Form

Propriétés

action	Chaîne en lecture/écriture. URL de destination où envoyer les données.
elements[]	Tableaux d'éléments HTML contenus dans le formulaire.
encoding	Chaîne en lecture/écriture. Spécifie l'encodage du formulaire.
length	Longueur du tableau d'élément.
name	Nom du formulaire en lecture/écriture.
target	Nom du cadre de destination en lecture/écriture.

Méthodes

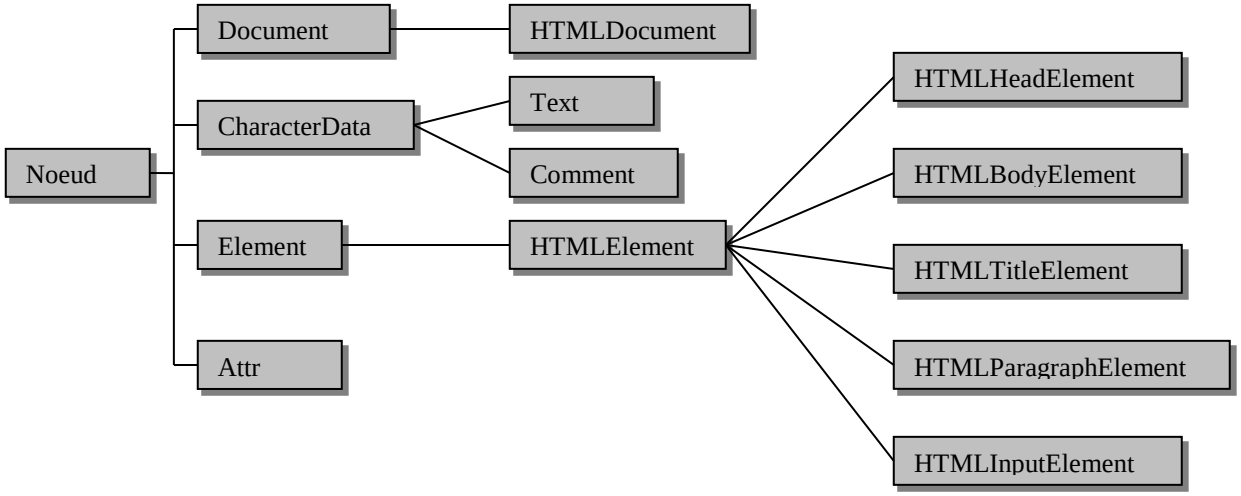
reset()	Réinitialise chacun des éléments à sa valeur par défaut.
submit()	Soumet le formulaire.

Gestionnaires d'évènements

onreset	Invoquée juste avant que les éléments du formulaire soit réinitialisés.
onsubmit	Invoqué juste avant que le formulaire soit soumis.

[4] Flanagan David ; JavaScript La référence ; O'Reilly ; 2002 ; 955 pages.

[4] Pour les méthodes et propriétés de *element*, voir:
Flanagan David ; JavaScript La référence ; O'Reilly ; 2002 ; pages 623 à 634.


Noeud
Document
CharacterData
Element
Attr
HTMLDocument
Text
Comment
HTMLElement
HTMLHeadElement
HTMLBodyElement
HTMLTitleElement
HTMLParagraphElement
HTMLInputElement