

De la bonne utilisation de

JAVASCRIPT

Ou comment ne pas transformer vos pages web
en sapin de Noël ;-)

Vincent MAZENOD – IGE CNRS
Webmestre de www.cerdi.org
Centre d'Etude et de Recherche sur le
Développement International



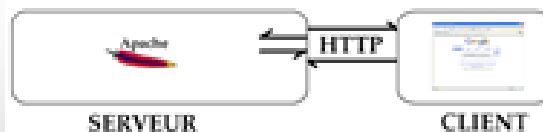
vmazenod@free.fr
Site de référence
<http://fr.selfhtml.org/javascript/>

Le WEB

Le World Wide Web :

- Architecture réseau permettant d'accéder à des ressources (pages web) liées entre elles (hypertexte)
- Ces ressources sont situées sur des machines (serveur web) reliées par le réseau des réseaux Internet
- La consultation de ces ressources est basée sur le modèle Client/Serveur et nécessite un protocole réseau spécifique : HTTP

Le modèle Client/Serveur [www]



- le navigateur (browser) demande un document au serveur Web (Apache, IIS, roxen, Zope, ...)
- celui-ci envoie le document (après vérification)
- le navigateur interprète l'information reçue (HTML, texte, image...)

HTTP (Hyper Text Transfer Protocol)

Protocole de Dialogue entre le Navigateur et le serveur Web

Langages et protocoles

HTML (Hyper Text Markup Language)

Langage à base de balises de type <BALISE> pour la description de documents hypertexte (liens) et hypermédia (images, sons...)

URL (Uniform Resource Locator)

Adresse universelle de ressource en 3 parties :

- > le protocole (par quelle méthode accéder ?)
- > l'adresse DNS du site : www.monsite.com (où trouver l'information ?)
- > le chemin pour y accéder et le nom du document (quel document récupérer ?)

Le web statique

> Pour une ressource donnée le serveur renvoie toujours la même réponse

> HTML permet seulement de présenter du texte, des liens et des images

> Une page HTML simple est appelée page statique, elle n'offre que peu d'interaction à l'utilisateur

Le web dynamique

> Pour palier à ce manque d'interactivité il existe 3 méthodes :

- > Java permet d'écrire des applets (petites applications) interprétables par le navigateur du client via une machine virtuelle installée sur le poste client
 - ⊗ Nécessite une machine virtuelle, gestion de la mémoire
- > La technologie SERVER-SIDE (PHP, ASP, CGI): langage de script interprété par le logiciel serveur (Apache,...) en fonction de paramètres passés par le client
 - ⊗ Chaque interaction du Client nécessite une nouvelle requête vers le serveur
- > La technologie CLIENT-SIDE (JavaScript): langage de script à placer au sein du code HTML et interprété par le client (navigateur)
 - ⊗ Code disponible au niveau du client, Problème de compatibilité selon les navigateurs

Pourquoi le CLIENT-SIDE?

- > Améliorer l'interactivité (temps de réponse plus court)
- > Améliorer les débits sur le réseau (éviter des envois erronés, économie de requête au serveur web)
- > Proposer des pages dynamiques (ergonomie, personnalisation, animation...)
- > Aucun environnement ni compilateur nécessaire au développement : un éditeur de texte et un (des) navigateur(s) sont suffisants

Exemples

- Test d'un [formulaire](#) avant envoi, pour vérifier la validité d'une adresse e-mail par exemple:
Attention: JavaScript peut être désactivé ou non supporté par le navigateur ...
- Application numérique : Le [Panier](#)
- Affichage dynamique : Le [roll Over](#)
Attention toutes les images doivent être chargées ... la taille est x2
- Génération de HTML [Le calendrier](#)

Caractéristiques

JavaScript est un langage

- interprété (pas de compilation) → Langage de script
- sensible à la casse
- à base d'objets (euh ... not a real one!)
- N'a pas accès aux fichiers locaux! Pour des raisons de sécurité
- multi-plateforme (ne dépend pas du système d'exploitation)
- développé par Netscape (nom d'origine LiveScript)
- Microsoft (de son côté) a développé Jscript
- Problèmes de compatibilité entre JavaScript et Jscript (propriétaire Microsoft)



<http://fr.selfhtml.org/introduction/technologies/javascript.htm>

Le noyau Javascript

Au niveau du langage, on distingue :

- le noyau JavaScript (le coeur du langage) comportant les objets de base, les opérateurs, les structures de contrôle...
- un ensemble d'objets prédéfinis associés au navigateur (fenêtres, documents, images...)

Il existe aussi la possibilité d'exécuter du code JavaScript du côté du serveur (communications avec une BD, des fichiers...) mais l'usage est plus restreint.

ECMA Script

L'organisme de standardisation ECMA (European Computer Manufacturers Association) a défini un standard ECMAScript basé sur JavaScript 1.1.

Ce standard ECMA-262, repris par l'ISO définit les caractéristiques du noyau du langage

<http://www.ecma-international.org/publications/standards/Ecma-262.htm>

Aujourd'hui, JavaScript 1.3 et JScript 3.0 sont conformes à cette norme ECMA-262 mais ils ont aussi tous les deux

- quelques extensions
- quelques différences au niveau du modèle objet du navigateur

Javascript et HTML

- Code JavaScript contenu dans les documents HTML entre les balises `<script>` et `</script>`
- Les fonctions JavaScript peuvent être appelées :
 - depuis des événements associés aux balises
 - le click de la souris sur un lien
 - `Ouvre une fenêtre`
 - comme une URL classique
 - `Call my Function`
 - en insérant le code JavaScript au milieu du code HTML
 - `
<script>AffCalendar();</script>`

Gestion des evenements

JavaScript reconnaît certains **événements** :

- Actions de la souris : déplacement, click...
- Chargement d'une page, d'une image
- Autres : erreur, soumission de formulaire...

et permet de leur associer une action :

- Une fonction
- ou directement du code JavaScript

en arguments de Mots-clefs onEvenement (onClick, onLoad...) associés aux balises HTML

Gestionnaire d'évenements

onBlur : désélection (PASSWORD, TEXT , TEXTAREA, SELECT)
onChange : modification (PASSWORD, TEXT , TEXTAREA, SELECT)
onFocus : sélection de l'élément (PASSWORD, TEXT , TEXTAREA, SELECT)
onSelect : sélection de valeur (PASSWORD, TEXT, TEXTAREA)
onClick : Clique sur l'élément (A HREF, BUTTON, CHECKBOX, RADIO, RESET, SUBMIT)
onMouseover : souris pointe (A HREF, AREA)
onMouseout : souris hors du lien (A HREF, AREA)
onLoad : chargement page (BODY)
onUnload : déchargement (BODY)
onError : erreur au chargement (BODY, IMG)
onAbort : arrêt du chargement (IMG)
onSubmit : action submit (FORM)
onReset : action reset (FORM)

La balise <SCRIPT>

Le code JavaScript peut être placé :

- dans l'en-tête `<head>...</head>` (déclaration de l'ensemble de fonctions appelées à partir d'événements, d'URL)
- dans le corps du document `<body>...</body>` (pour un traitement synchrone lors de l'interprétation du document)
- dans un fichier séparé

```
<SCRIPT SRC="monprog.js">
```

Il est également possible de spécifier la version de Javascript utilisée pour écrire le script via l'attribut language (par défaut JavaScript)

```
<SCRIPT language="JavaScript 1.0">
```

Exemple

```
<HTML> <HEAD>...
<SCRIPT LANGUAGE="JavaScript">
<!--
//Certains navigateurs n'interprètent pas JavaScript
//on peut mettre le code entre commentaires
Function fonction1(){
    alert("ca court!! Javascript");
}
//-->
</SCRIPT>
</HEAD>
<BODY>
...
<SCRIPT>fonction1(); </SCRIPT>
...
<BALISE onEvenement="fonction1();alert("fonction1 exécutée"); ">
...
</BODY>
</HTML>
```

Le langage Javascript

- ❑ Toute instruction (ligne de code se termine par un point virgule)
- ❑ Variables (faiblement typées) : l'utilisation ne nécessite pas de déclaration
- ❑ Opérateurs et Instructions (ceux du langage C)
- ❑ Méthodes : globales (associées à tous les objets), fonctions définies par l'utilisateur
- ❑ Objets : prédéfinis (String, Date, Math...) , liés à l'environnement (window, document...)
- ❑ Les Entrées/Sorties sont déléguées à l'hébergeant (le navigateur)
- ❑ Commentaires comme en langage C : //commente une ligne, /*comment une portion de code multi-ligne */

Opérateurs

Ceux du langage C

- > arithmétiques : + - * / %
- > Incrémentation / décrémentation : i++ , i- , ...
- > logiques : && (ET) || (OU) ! (NON)
- > bit à bit :
 - ❖ & (AND) | (OR) ^ (XOR) ~ (Not)
 - ❖ décalages : >> (à droite) << (à gauche) >>> (non signé)
- > comparaisons: ==, !=, <=, >=, < > ,
- > concaténation de chaînes : +

L'opérateur + est l'addition ou la concaténation selon qu'il agit sur un numérique ou sur une chaîne de caractère (qui est le type d'une variable par défaut!)



Affectations

affectation simple : =

```
MyString="valeur";
```

affectation conditionnelle :

```
var = (condition) ? exp_alors : exp_sinon
```

```
MyOtherString = (a > b) ? "plus" : "moins";
```

Affectation avec opération : += -= *= ...

```
MyInteger +=3; // équivaut à MyInteger = MyInteger +3;
```



distinguer l'affectation (=) et la comparaison (==)

Types primitifs

- Number: Entier, décimal ou hexadécimal ou octal, réel (-2.3452E-12)
- Booléen (Boolean) : true ou false
- Chaîne de caractères (String) : 'chaine' ou "chaine"
 - Caractères séparateurs
 - \t (tabulation)
 - \n (à laligne)
 - \r (retour chariot)
 - \b (backspace)
 - \f (saut de page)

Conversions

- Typage faible - Type String = type dominant
- JavaScript fait des conversions implicites selon les besoins

```
N=12; // N numérique
T="34"; // T chaîne de caractères
X=N+T; // X est la chaîne de caractères "1234"
```
- Il existe des types particuliers :
 - Null (null)
 - Undefined (undefined)
- des nombres particuliers :
 - Infinity,
 - NaN (Not a Number)
- var : déclaration d'une variable

Structure conditionnelle

```
if (condition) {           // instruction conditionnelle
  instructions
}
[ else {instructions } ] //Clause else optionnelle
```

Exemple

```
if (choix=="Calcul"){
  alert("Vous avez demandé la calcullette!");
}
```

Structures itératives

```
for (i=1 ; i<N ; i++){ // itérations
  instructions;
}
```

```
while (condition){ // itérations
  instructions;
}
```

break; //sortie d'une boucle

continue; //itération suivante d'une boucle

Fonctions

```
// Déclaration de la fonction
function nom(p1,p2...){
  instructions ;
  return code;
}
// Appel de la fonction
X = nom(5, "jours");
```

Notion d'objet Javascript

Un objet est une collection de propriétés (pool de variables associées à un même objet) et de méthodes (pool de fonctions associées à un même objet)

Un objet dérive d'une classe (Sorte de moule à objets ;-))

JavaScript n'a pas de classe (pseudo-classe), pas de sous-classe, ni d'héritage ...

JavaScript permet par contre de définir des propriétés après définition via l'objet "prototype"

JavaScript met à disposition des objets prédéfinis (arborescence d'objets) et permet de créer ces propres objets

Création d'un objet par définition de son constructeur (fonction du nom de la pseudo-classe avec affectation des propriétés à partir des paramètres et déclaration des méthodes)

Exemple de creation

```
// Définition d'une méthode d'objet (équivalent d'une fonction)
function surface(){
  return (this.longueur * this.largeur);
}
// Définition d'un Constructeur de l'objet Rectangle
function Rectangle(L, l){
  this.longueur = L; // propriétés
  this.largeur = l; // propriétés
  this.surface=function surface(){
    return (this.longueur * this.largeur);
  } // méthode
}
var rectangle1 = new Rectangle(20,10,0,0);
Exemple et détail http://minilien.com/?hgkumrZxcP
```

Instructions spécifiques Objets

Création d'une instance : **new** // à utiliser dans constructeur
`MonObjet = new objet(x1,x2);`

Référence de l'objet : **this** //référence l'objet courant
`this.nom=valeur; //affecte la valeur "valeur" à la propriété nom`

Parcours : **for** //parcours des propriétés d'un objet
`for (variable in objet) { instructions }`

Références à un objet : **with** //Evite l'utilisation du **this**
`with (MonObjet){
 y1=zz; // raccourci de MonObjet.zz
 y2=vv; // raccourci de MonObjet.vv
}`

Suppression d'objet : **delete**
`delete MonObjet;`

Les objets ECMAScript

Structure de données en arbre avec au sommet l'objet **Global** qui définit un ensemble de méthodes communes à tous les objets.

Les objets "prédéfinis" sont

1. Object (création d'un objet)
2. Function (création d'une fonction)
3. Array (création d'un tableau)
4. String (création d'une chaîne de caractères)
5. Boolean (création d'un booléen)
6. Number (création d'un nombre)
7. Date (création d'une date)
8. Math (création de structures mathématiques)

A ces objets ECMAScript, s'ajouteront une hiérarchie d'objets liée à l'hébergeant (selon le navigateur)



<http://selfhtml.selfhtml.com/fr/javascript/objets/index.htm>

Objet String

Type par défaut d'une variable en JavaScript

- Propriété
length
- Méthodes de manipulation de chaînes :
indexOf("chaîne", "s/chn.pos), substring(début,fin+1), charAt(n),
lastIndexOf("chaîne"), toLowerCase(), toUpperCase(), split(), toString()

Exemple :

```
var T = Bonjour;  
T.indexOf("o"); // 1  
T.lastIndexOf("o"); // 4  
T.charAt(3); // j  
T.substring(3,7); // jour  
T.toUpperCase(); // BONJOUR
```

Méthodes globales de l'objet String

- **escape** ("car") → code ASCII
`escape("é"); // %E9`
- **unescape** ("%exp") → caractère de code ASCII
`unescape("%26"); // &`
- **eval** ("exp") → évaluation d'une expression et retour d'un nombre
`eval("15"+"15"); // Marignan ;-)`
`eval("MyTab"+"["+2+"]"); // 3ème valeur du tableau MyTab`

Objet String : méthodes HTML

Créer un lien en JavaScript : `anchor()`, `link()`

```
"Institut".link("http://www.univ-bpclermont.fr");  
<a href="http://www.univ-bpclermont.fr">Institut </a>
```

Même chose avec `big`, `sub`, `sup`, `blink`, `bold`, `fixed`, `italics`, `small`, `strike`, `fontcolor`, `fontsize`

```
"Bonjour".big;  
<big> Bonjour </big>
```

```
"Bonjour".fontsize(5);  
<font size=5>Bonjour</font>
```

Objet Number

Propriétés :

MAX_VALUE + grand nombre pouvant être sauvegardé
MIN_VALUE + petit nombre pouvant être sauvegardé
NaN Not A Number
POSITIVE_INFINITY +infini
NEGATIVE_INFINITY -infini

Méthodes (communes à tous les objets)

toString() retourne une chaîne
valueOf() génère une erreur si l'objet n'est pas un nombre

Objet Boolean (simple encapsulation d'un booléen)

Propriété

Boolean renvoie un booléen

Méthodes :

toString(), valueOf()

Méthodes globales de l'objet Number

isNaN(k) retourne VRAI si k est un NaN

isFinite(k) retourne faux si k est un NaN, +infini ou -infini et vrai sinon

parseInt("chn", b) → entier chn dans la base b (0 ou NaN sinon)

parseFloat("chn") → flottant chn (0 ou NaN sinon)

```
parseInt("15",10);//15
parseInt("F",16);//15
parseInt("15.99",10);//15
parseFloat("15,99E-12");//1.599e-11
```

Objet Math

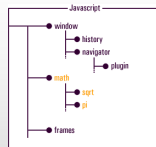
Structures mathématiques

Propriétés

➤ constantes mathématiques : E, PI, SQRT2, SQRT1_2, LN2, LN10, LOG2E, LOG10E

Méthodes : fonctions usuelles

- Trigonométriques : cos, sin, tg, acos, atan, asin, atan2
- abs (valeur absolue)
- ceil (entier inf), floor (entier sup), round (entier + proche)
- exp, log, sqrt, pow(x,a)
- max (a,b), min (a,b)
- random : nombre entre 0 et 1



Objet Date

Représentation des dates

Propriété

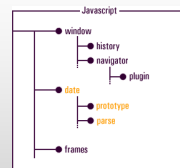
length (valeur initiale 7)

Nécessité d'instancier avec new

```
X = new Date() --> date du jour
Y = new Date(args)
```

MoisJour,An [Heure:Minute:Seconde] avec Mois en lettres
ou An,Mois,Jour[,Heure,Minute,Seconde] tout en chiffres

Hier = new Date(1998, 01, 06, 09, 15, 00) // 6.1.98 à 9H15



Objet Date : methodes

Accès	Affectation	Valeurs
getDate()	SetDate ()	1-31
getDay()	SetDay()	0-6
getMonth()	SetMonth()	0-11
getYear()	SetYear()	19XX ou 20XX
getFullYear()	SetFullYear()	
getHours()	SetHours()	0-23
getMinutes()	SetMinutes()	0-59
getSeconds()	SetSeconds()	0-59
getMilliseconds()	SetMilliseconds()	0-999
getTime()	SetTime()	Nombre de ms depuis 1.1.1970
getTimezoneOffset()	SetTimezoneOffset()	Différence en minutes avec GMT

Objet Date : méthodes

- toGMTString() : *conversion en date GMT*
- toLocaleString() : *conversion en heure locale*
- UTC(an,mois,jour,H,M,S) : *conversion en variable date (GMT)*
- Parse(« December 25, 1998 ») : *conversion en variable Date*

Objet Array

Tableau d'éléments

Propriété

length nombre d'éléments

Méthodes

join() concatène tous les éléments en une chaîne de caractères séparés par une virgule ou un séparateur passé en argument

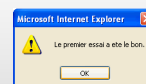
reverse() transpose le tableau (1<->N 2<-> N-1...)

sort() trie les éléments dans l'ordre lexicographique ou selon la relation d'ordre passée en argument

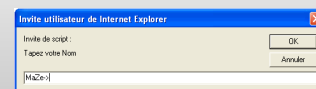
Boîtes de dialogue

Parmi les objets JavaScript du navigateur, on distingue 3 objets de type fenêtre qui proposent des fenêtres prédéfinies pour faire des Entrées/Sorties :

➤ **alert** : fenêtre d'affichage d'un message avec un bouton de fermeture



➤ **prompt** : fenêtre de saisie d'une information avec un champ de saisie, et 3 boutons pour valider, effacer et annuler



➤ **confirm** : fenêtre de test avec 2 boutons pour valider ou annuler



Debuggage

Toujours délicate avec les langages interprétés (messages peu explicites)

Quand une instruction plante, tout le javascript de la page plante ☹

Console JavaScript (mozilla, Netscape):

en tapant javascript: comme adresse, le navigateur ouvre une fenêtre avec les messages d'erreur.

Possibilité de tester du code directement

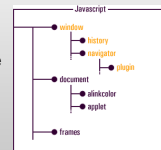
L'objet Window



<http://fr.selfhtml.org/javascript/objets/window.htm>

➤ L'objet window (fenêtre d'affichage) est l'objet par excellence dans Javascript, car il est le parent de chaque objet qui compose la page web, il contient donc:

- l'objet **document**: la page en elle-même
- l'objet **location**: le lieu de stockage de la page
- l'objet **history**: historique des pages visitées
- l'objet **frames**: les cadres



L'objet Window - proprietes

closed (fenêtre fermée)
defaultStatus (affichage normal dans la barre d'état)
innerHeight (hauteur du domaine d'affichage)
innerWidth (largeur du domaine d'affichage)
locationbar (barre d'adresse)
menubar (barre de menus)
name (nom de fenêtre)
outerHeight (hauteur de la fenêtre globale)
outerWidth (largeur de la fenêtre globale)
pageXOffset (position de départ de la fenêtre à partir de la gauche)
pageYOffset (position de départ de la fenêtre à partir du haut)
personalbar (barre pour les adresses favorites)
statusbar (barre d'état)
status (Contenu de la barre d'état)

L'objet Window - proprietes

alert() (boite de dialogue avec infos)
blur() (quitter la fenêtre)
close() (fermer la fenêtre)
confirm() (boite de dialogue pour confirmer)
find() (chercher du texte)
focus() (activer la fenêtre)
moveBy() (se mouvoir avec des mentions relatives)
moveTo() (se mouvoir avec des mentions absolues)
print() (imprimer)
prompt() (fenêtre de dialogue pour la saisie de valeur)
releaseEvents() (fermer un événement)
resizeBy() (modifier la taille avec des mentions relatives)
resizeTo() (modifier la taille avec des mentions absolues)
routeEvent() (parcourir la hiérarchie des gestionnaires d'événement)
scrollBy() (défiler un certain nombre de pixels)
scrollTo() (défiler jusqu'à la position)
setTimeout() (entamer le compte à rebours)
stop() (interrompre)

L'objet Window - Methode open : les pop up

```
window.open("URL", "Nouvelle fenetre","options_de_la_fenetre");
```

•URL : l'url de la page qui sera affichée dans la nouvelle fenêtre, c'est-à-dire l'emplacement physique de celle-ci.

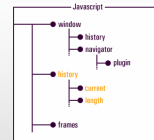
Permet d'initialiser les propriétés suivantes de l'objet Windows créée:

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

L'objet history

history : propriété de l'objet document.

Contient l'historique du navigateur,



Propriétés et méthodes

length : nombre d'objets dans l'historique

méthode **back** : aller à l'URL précédent dans l'historique

méthode **forward** : aller à l'URL suivant dans l'historique

méthode **go(variable)** : aller à un des URL de l'historique.

Où variable : nombre entier qui détermine le nombre de pages relatif auquel se trouve l'URL désiré, ou chaîne de caractère

L'objet navigator

navigator : permet de récupérer des informations sur le navigateur

propriétés en lecture seule :

- **navigator.appCodeName** : code du navigateur. Un navigateur a pour nom de code Mozilla
- **navigator.appName** : nom du navigateur
- **navigator.appVersion** : version du navigateur. Cette propriété prend la forme suivante: Numéro de version(plateforme (système d'exploitation), nationalité)
- **navigator.userAgent** : chaîne de caractère qui comprend toutes les informations.

Les navigateurs ne supportent pas tous le Javascript de la même façon, on peut faire du « spaghetti code » pour avoir le même effet sur tous les navigateurs ... Cette pratique est toutefois déconseillée



<http://cybercodeur.net/weblog/presentations/avantages/avantages.html>

L'objet document



<http://fr.selfhtml.org/javascript/objets/document.htm>

Contient des informations concernant le document courant et les objets (balises HTML qui le compose)

Quelques propriétés

cookie (chaîne de caractères pouvant être sauvegardée d

defaultCharSet (jeu de caractères normal)

lastModified (dernière modification du document)

referrer (pages déjà visitées)

title (titre du fichier)

URL (adresse URL du fichier)

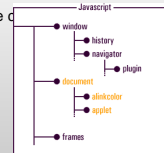
Quelques méthodes

captureEvents() (surveiller les événements)

getSelection() (texte sélectionné)

write() (écriture du HTML bout à bout dans la fenêtre du document)

writeln() (écriture du HTML bout à bout ligne par ligne)



Le DOM



<http://www.quirksmode.org/dom/intro.html>

- Le DOM permet de se représenter le document sous forme d'arborescence de balises
- Il permet de manipuler n'importe quel élément (balise) de notre page Web via les propriétés et les méthodes suivantes
- Le DOM est LA méthode d'accès aux éléments d'une page Web
- Le DOM fait très bon ménage avec CSS et XHTML
- Le DOM est lié à l'objet document (Premier nœud)

Méthodes et attributs liés au DOM

Propriétés

parentNode (nœud parent du nœud courant)
childNodes (Tableau)
firstChild (premier nœud enfant)
LastChild (premier nœud enfant)

Méthodes

createAttribute() (créer un nœud d'attributs)
createElement() (créer un nœud d'éléments)
createTextNode() (créer un nœud de texte)
getElementById() (Accès à l'élément HTML par l'attribut Id)
getElementByName() (Accès à l'élément HTML par l'attribut name)
getElementsByTagName() (Accès à l'élément HTML par liste d'éléments)
Pour supprimer un élément il suffit de lui affecter la valeur spéciale **null**

[Exemple](#)

CSS (cascading style sheet)



<http://fr.selfhtml.org/css/index.htm>
<http://brainjar.com/css/using/default.asp>

- Idée de base séparé la structure de la présentation d'un document
 - HTML/XHTML structure le document (paragraphe, grand titre, éventuellement table ...)
 - CSS regroupe toutes les déclarations liées aux styles des éléments du document
- Dans les balises HTML on n'utilise plus que les attributs suivant : id, name, class
- Les tableaux ne sont pas des élément de mise en forme ... on préférera les calques (balises DIV)

Utilisation de CSS

- On peut regrouper les déclarations CSS dans un fichier.css ou entre les balises `<style>...</style>` dans l'entête de la page web (`<head>...</head>`)
- On peut redéfinir le style par défaut d'une balise HTML:

```
P {  
  color: #666666;  
  margin-left: 2 cm;  
  border-top: 1px solid gray;  
}
```

Utilisation de CSS

- On peut définir un style pour une balise HTML ayant un id spécifique:

```
#BoiteVerte {  
  position : absolute;  
  top : 20px;  
  left : 100px  
  color: green;  
  border: 1px solid gray;  
}
```

Appeler ce style en HTML

```
<div id="BoiteVerte"> le contenu de la boite verte </div>
```

[Exemple](#)

Utilisation de CSS

- On peut définir un style pour une balise HTML ayant un id spécifique:

```
.BoiteVerte {  
  position : absolute;  
  top : 20px;  
  left : 100px  
  color: green;  
  border: 1px solid gray;  
}
```

- Appeler ce style en HTML

```
<div class="BoiteVerte"> le contenu de la boite verte </div>
```

- Le DOM permet entre autre de manipuler les styles CSS via la propriété style de chaque balise

```
document.getElementById("MyP").style.color="red";
```