

# Introduction au JavaScript

par Serge P.


Date de publication : 27/11/2006

Dernière mise à jour : 16/05/2009

Cet article est une introduction au langage JavaScript. Il est destiné à ceux et celles qui souhaitent découvrir ce langage qui permet de dynamiser les pages Web. Néanmoins, de bonnes bases en HTML et CSS sont nécessaires pour aborder sereinement le JavaScript.


I - Présentation du langage JavaScript et des navigateurs.....	3
I-A - JavaScript.....	3
I-B - Navigateurs.....	3
I-C - Limites du JavaScript.....	4
II - Syntaxe.....	4
II-A - Instructions - Conditions - Boucles.....	4
II-B - Scripts dans l'en-tête du fichier HTML.....	5
II-C - Scripts externes.....	6
II-D - Commentaires.....	6
II-E - Objets internes.....	6
II-F - Accolades.....	7
II-G - Variables.....	8
III - Événements.....	8
IV - Utiliser les objets du navigateur.....	11
IV-A - L'objet window.....	11
IV-B - L'objet window.navigator.....	11
IV-C - L'objet window.document.....	11
V - Manipuler la page - Les grandes lignes du DOM.....	12
V-A - Représentation de la page.....	12
V-A-1 - Propriétés des nœuds.....	12
V-A-1-a - Propriétés.....	12
V-A-1-b - Remarques sur innerHTML et nodeValue.....	13
V-A-2 - Exemples d'arbres DOM sous divers navigateurs.....	13
V-A-2-a - Arborescence selon Firefox.....	14
V-A-2-b - Arborescence selon Firefox v2.....	14
V-A-2-c - Arborescence selon Internet Explorer v5.....	15
V-B - Création, insertion, suppression d'un nœud.....	16
V-B-1 - Méthodes JavaScript pour la gestion des nœuds.....	16
V-B-2 - Exemple.....	16
V-B-2-a - Arborescence du document.....	16
V-B-2-b - Création des éléments.....	17
V-B-2-c - Insertion des objets dans le document.....	18
V-C - Ajax.....	19
VI - Conclusion - Liens divers - Remerciements.....	19
VI-A - Conclusion.....	19
VI-B - Liens.....	19
VI-C - Remerciements.....	20

## I - Présentation du langage JavaScript et des navigateurs

 Pour aborder JavaScript, il faut déjà connaître le langage HTML ainsi que les feuilles de styles (CSS).

### I-A - JavaScript

JavaScript est un langage interprété par le navigateur. Le JavaScript est un langage « client », c'est-à-dire exécuté chez l'utilisateur lorsque la page Web est chargée. Il a pour but de dynamiser les sites Internet.

 JavaScript est à ne pas confondre avec Java !

**FAQ Qu'est-ce que le Javascript ?**

**FAQ A quoi sert le Javascript ?**

**FAQ JAVA ou Javascript ?**

**Le JavaScript est un langage sensible à la casse (« case sensitive »).**

JavaScript est un langage objet : chaque objet possède des méthodes (ou fonctions), des propriétés et .... des objets. Dans une page Web, l'objet le plus élevé dans la hiérarchie est la fenêtre du navigateur : *window*. Cet objet *window* contient entre autres l'objet *document* qui lui même contient tous les objets contenus dans la page Web (paragraphes, formulaires, etc...). En plus de ces objets, il existe des objets créés par l'utilisateur.

Les méthodes sont des fonctions qui permettent d'agir sur certaines propriétés de l'objet, les propriétés contiennent les paramètres d'un objet.

**Exemple d'un objet voiture : nous allons lui attribuer**

- des propriétés : la couleur, la marque, le numéro d'immatriculation ;
- des méthodes : tourner(), avancer(), reculer(), changer la couleur() ;
- des objets : les phares, les pneus.

Pour résumer une méthode engendre une action, un verbe qualifie cette action, une propriété est généralement qualifiée par un nom.


Dans l'exemple de la voiture une méthode permet de changer la couleur de la voiture, par contre aucune méthode ne nous autorise à changer la marque de cette voiture (ce qui entraînerait une modification des autres propriétés et éventuellement l'apparition ou la disparition de méthodes).

Il en sera ainsi également avec nos objets JavaScript : nous pourrons accéder voire modifier les propriétés (couleur du texte, style de la fonte) des objets grâce aux méthodes.

### I-B - Navigateurs

Voici une liste non exhaustive des navigateurs :

Linux / UNIX	Windows	MacOS
Firefox, Netscape, Mozilla, Konqueror, Lynx, Opéra	Internet Explorer, Firefox, Netscape, Opéra, Chrome	Internet Explorer, Konqueror, Opéra, Safari

 Lynx est un navigateur qui n'interprète pas le JavaScript.

Tout irait pour le mieux si ces navigateurs utilisaient pour un même objet, les mêmes propriétés et les mêmes méthodes pour les définir. Ce qui est loin d'être le cas. Par ailleurs, Internet Explorer interprète également le JScript, un JavaScript créé par Microsoft (ActiveX). Nous n'aborderons pas ce langage spécifique à IE.

## I-C - Limites du JavaScript

Le JavaScript est difficilement compatible entre les différents navigateurs. Il faut toujours se décider jusqu'à quel point ça doit être compatible.

Tout le monde n'a pas JavaScript : Il faut toujours que la page contienne l'ensemble de l'information, accessible même sans JavaScript. JavaScript est là pour apporter un plus (ergonomie, dynamisme), mais on doit pouvoir s'en passer.

JavaScript **n'est pas sécurisé**. Les programmes JS sont exécutés sur le client, on n'est jamais sûr de leurs résultats, il ne faut donc jamais faire confiance à une donnée provenant du client.

## II - Syntaxe

### II-A - Instructions - Conditions - Boucles

Il est fortement recommandé de terminer l'ensemble des instructions JavaScript par un point virgule (même si, en effet, ce n'est pas toujours nécessaire).

Les instructions ci-dessous ne se terminent pas par un point virgule :

#### Les définitions de fonctions

```
function maFonction()  
{  
  .....  
}
```

#### Les conditions

```
if (var1==var2)  
{  
  .....  
}  
else  
{  
  .....  
}
```

### Syntaxe des conditions :

- égalité : == (cette syntaxe est également utilisée pour comparer deux chaînes de caractères)
- différent de : != (même remarque que ci-dessus)
- inférieur ou égal à : <=
- supérieur ou égal à : >=
- inférieur à : <
- supérieur à : >
- et logique : &&
- ou logique : ||
- identique à : ===
- non identique à : !==
- et bit à bit : &
- ou bit à bit : |

#### Les boucles for

### Les boucles for

```
for (i=0; i<5; i++)  
{  
    .....  
}
```

### Les boucles while

```
while (a<b)  
{  
    .....  
}  
  
do  
{  
    .....  
}while (a<b)
```


## II-B - Scripts dans l'en-tête du fichier HTML

Tout script est encadré des balises `<script>` `</script>`, on précise également le type MIME grâce à l'attribut `type` :

### Script dans l'en-tête du fichier HTML

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">  
<html>  
  <head>  
    <title>Titre</title>  
  
    <script type="text/javascript">  
      <!--  
  
      //-->  
    </script>  
  
  </head>  
  <body>  
  
  </body>  
</html>
```

Les commentaires restent présents pour une raison historique : les premiers navigateurs n'interprétant pas le JavaScript et pour éviter un affichage du texte dans la page web, les scripts étaient encadrés de commentaires. De plus, omettre les commentaires amènent les validateurs à tenter d'interpréter le code JavaScript comme du HTML, ce qui implique des erreurs de validation non justifiées.

 **Ne pas confondre les commentaires HTML et les commentaires JavaScript.**

### Commentaire HTML

```
<!-- Ceci est un commentaire HTML -->
```

### Commentaire JavaScript

```
// Ceci est un commentaire JavaScript sur une ligne  
/* Ceci est un commentaire JavaScript  
sur plusieurs lignes */
```

Ces balises script sont généralement insérées dans le head de la page, ou entre les balises body. Dans ce dernier cas les scripts sont exécutés au fur et à mesure du chargement de la page.

Il est possible d'insérer du code JavaScript dans les balises HTML. Cependant, il faut que le code inséré soit bref pour des questions de lisibilité (dans le cas des événements).

## II-C - Scripts externes

On peut enregistrer le script dans un fichier indépendant de la page Web. Dans ce cas, on précise dans le *head* le lien vers ce fichier. L'avantage est que l'on peut ainsi réutiliser le script pour une autre page.

### Lien vers un script externe

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
  <title> </title>
  <script type="text/javascript" src="MonFichier.js"></script>
</head>
<body>

</body>
</html>
```

## II-D - Commentaires

Il existe les commentaires « multilignes » : ils commencent par `/*` et se terminent par `*/`


### Commentaire JavaScript

```
/* un commentaire
sur plusieurs lignes */
```

et des commentaires sur une ligne : ils débutent par `//`

### Commentaire Javascript

```
// un commentaire une ligne
```

 *Il semblerait que la présence des accents et des apostrophes dans ces commentaires contribuerait à une mauvaise interprétation des scripts. Cette source d'erreurs peut-être levée en précisant le charset du fichier JavaScript.*

#### **FAQ Les DOCTYPE en HTML**

*De plus, l'utilisation de commentaires multilignes peut perturber l'interprétation dans le cas d'utilisation d'expressions régulières du fait de la présence possible des caractères `/*` ou `*/` dans celles-ci.*

## II-E - Objets internes

Les objets internes JavaScript commencent par une majuscule : String, Math, Array, Boolean, Date, Number, Function (à ne pas confondre avec le mot-clef `function`), RegExp, etc...

Les méthodes ainsi que les propriétés d'un objet commencent par une minuscule. Toutes les méthodes internes à JavaScript sont sensibles à la casse (« case sensitive »).


- exemple de méthodes : `toLowerCase()` ; `getElementById()` ; `charAt()` ; `fromCharCode()` ; etc.
- exemple de propriétés : `id` ; `type` ; `innerHTML` ; `tagName` ; `style` ; etc.

L'accès à une méthode ou à une propriété d'un objet se fait en plaçant un point entre le nom de l'objet et la propriété ou la méthode.

```
var monObjet = document.getElementById("idObjet");  
monObjet.style.display = "none";
```

ou

```
document.getElementById("idObjet").style.display = "none";
```

 **Le langage JavaScript est un langage « case sensitive » : les variables, les méthodes, les attributs, les mots-clés, etc. ont une syntaxe très précise. Le non-respect de cette règle peu conduire à une mauvaise exécution des scripts.**

## II-F - Accolades

Plusieurs écoles : les accolades ouvrantes sont placées sur la même ligne que la condition ou la boucle ; ou un retour à la ligne est effectué pour l'accolade ouvrante.

### Accolades ouvrantes en fin de ligne

```
function maFonction() {  
  if (test1) {  
    .....  
    .....  
  }  
  else{  
    .....  
    .....  
  }  
  
  for (i=0; i<n; i++){  
    .....  
    .....  
  }  
}
```

### Accolades ouvrantes après un retour à la ligne

```
function maFonction()  
{  
  if (test1)  
  {  
    .....  
    .....  
  }  
  else  
  {  
    .....  
    .....  
  }  
  
  for (i=0; i<n; i++)  
  {  
    .....  
    .....  
  }  
}
```

On préconisera la seconde syntaxe à la première dans la mesure où elle offre une meilleure lisibilité dans le code.

## II-G - Variables

En JavaScript les variables ne sont pas typées. Il faut néanmoins les déclarer grâce au mot clef *var*. Une variable peut, au cours du programme, être tour à tour un entier, une chaîne de caractères, ou un booléen. Même si cette possibilité de programmation est offerte, il ne faut surtout pas s'y laisser tenter. Une variable doit garder le même type du début à la fin. Et donc ne pas hésiter à créer autant de variables que nécessaire.

### A ne pas faire

```
var i; //déclaration
i = 2; //entier
i = "bonjour"; //chaîne de caractères
i = true; //booléen
```

### Ce qu'il faut faire

```
var i, chaine, bool; //Déclaration de 3 variables
i = 2;
chaine = "bonjour";
bool = true;
```

### Bannir des noms de variables :

- du genre : truc, machin, toto, bidule... ;
- « kilométriques » : ceciEstLeNomDeMaJolieVariable ;
- avec des accents : maChaîneDeCaractères.

Les variables ne doivent pas être des mots-clefs JavaScript : *var*, *form*, *int*, *document*, etc..

## III - Evénements

Tous les événements commencent par *on* : *onclick*, *onload*, *onmouseout*, *onmouseover*.... Ils peuvent être insérés dans les balises HTML du document. Il est vivement conseillé de les écrire en minuscules.

### Syntaxe des événements

```
<body onload="maFonction()">
<input type="button" onclick="maFonction()">
```

### Pseudo-URL :

Les pseudos-URL sont insérées dans les balises de lien avec la syntaxe suivante :

### Pseudo-URL

```
<a href="javascript:alert('Coucou !!')">Mon Lien</a>
```

et ce pour faire la distinction avec un lien vers une autre page :

### URL

```
<a href="maPage.htm">Mon Lien</a>
```

**FAQ Voir à ce sujet Exemples d'événements :**

**Source** Testez sur série d'événements sur des objets de votre navigateur



Événement	Survient	Commentaires
onload	après le chargement de la page	
onunload	lors de la fermeture de la page	
onbeforeunload	juste avant la fermeture de la fenêtre	
onclick	lors d'un clic	l'événement survient lors d'un clic sur le bouton gauche
ondblclick	lors d'un double-clic	double-clic sur le bouton gauche
onmousedown	quand on enfonce le bouton de la souris	cet événement survient sur un clic gauche ou un clic droit
onmouseup	quand on relâche le bouton de la souris	cet événement survient sur un clic gauche ou un clic droit
onmousemove	lorsque la souris se déplace dans un objet du document	
onkeydown	quand on enfonce une touche du clavier	
onkeyup	quand on relâche la touche	
onkeypress	quand on enfonce une touche du clavier	<b>Firefox et IE5 sous Windows :</b> équivalent à onkeydown()
onblur	quand l'élément perd le focus	<b>IE5 sous Windows :</b> l'événement survient lorsque l'élément perd le focus. <b>Firefox sous Windows :</b> l'événement survient juste avant que l'élément ne perde le focus.
onfocus	quand l'élément a le focus	Cas de l'input de type file <input type="file" />. <b>IE5 sous Windows :</b> l'événement se produit lors du clic dans la boîte de texte ou après avoir cliqué sur « Ok » ou « Annuler » de la boîte de dialogue « Choisir fichier ». Pas d'événement lors du clic sur le bouton « Parcourir ». <b>Firefox sous Windows :</b> l'événement se produit uniquement lors du clic sur le bouton « Parcourir ».
onchange	quand l'élément perd le focus et que son contenu a changé	
onsubmit	juste avant l'envoi d'un formulaire	
onreset	lors de la réinitialisation du formulaire	
onselect	quand le contenu d'un élément est sélectionné	<b>Firefox sous Windows :</b> onselect n'est valide que pour les input de type texte (text, password, file et textarea).

		Le nombre d'appels à cet événement va dépendre de la manière dont est faite la sélection (clavier, double-clic...) et du navigateur. On ne peut pas donc se fier à cette indication pour déterminer le nombre de caractères ou de mots sélectionnés.
onscroll	lors de l'utilisation de la barre de défilement	<b>Firefox sous Windows :</b> l'événement onscroll ne fonctionne pas sur les balises textarea.
onbeforeprint	avant l'impression (après le clic sur le bouton Ok de la fenêtre d'impression)	Cet événement ne fonctionne que sous Internet Explorer.
onafterprint	après l'impression	Cet événement ne fonctionne que sous Internet Explorer.
oncopy	lors du copier vers le presse-papier	Cet événement survient lors d'un "CTRL+C" ou d'un copier via le menu contextuel.
onpaste	lors du coller depuis le presse-papier	Cet événement survient lors d'un "CTRL+V" ou d'un coller via le menu contextuel.

Il arrive souvent que l'un des arguments de la fonction appelée lors d'un événement soit l'objet *event* (gestionnaire des événements). Le cas le plus classique est lorsque l'on veut connaître la position de la souris. Avec Internet Explorer, l'objet *event* est contenu dans l'objet *window* : *window.event*. Avec Firefox ou Netscape, l'objet *event* est généré lors d'un événement (*onclick* par exemple) à la seule condition que celui-ci soit inclus dans la balise HTML. Par conséquent, pour récupérer l'événement produit, il faut que l'objet *event* soit un paramètre de la fonction.

#### Récupération de l'événement produit dans la page Web

```

<html>
<head>
<title></title>

<script type="text/javascript">
<!--
function position(ev)
{
    var Xfen, Xdoc, Yfen, Ydoc, el;

    Xfen = ev.clientX;
    Xdoc = Xfen + document.body.scrollLeft;

    Yfen = ev.clientY;
    Ydoc = Yfen + document.body.scrollTop;

    el = document.getElementById("idMouse");
    el.innerHTML = " Xdoc= "+Xdoc+" px ; Ydoc= "+Ydoc+" px<br>";
    el.innerHTML+= " Xfen= "+Xfen+" px ; Yfen= "+Yfen+" px";
}
//-->
    
```

### Récupération de l'évènement produit dans la page Web

```

</script>
</head>

<body onmousemove="position(event)">
  <div id="idMouse">
  </div>
</body>

</html>
    
```

Cependant, (et c'est un gros avantage) **ce code fonctionne également sous Internet Explorer**. Il n'est donc pas nécessaire d'ajouter des tests pour savoir s'il faut utiliser le paramètre *ev* de la fonction ou le gestionnaire d'événements *window.event* que seul IE comprend.

## IV - Utiliser les objets du navigateur

### Source Script d'affichage des propriétés des objets du document

Ce script affiche les propriétés (et la valeur de ces propriétés) d'un élément choisi dans une liste. Il permet, entre autres, pour un même élément de mettre en évidence les différentes propriétés utilisées par les navigateurs (pour cela il doit être exécuté sur différents navigateurs).

### IV-A - L'objet window

Cet objet représente le navigateur contenant l'objet *document*. Il est créé lors de l'ouverture du navigateur et contient toutes les propriétés et les méthodes de gestion de la fenêtre. Ses propriétés et ses méthodes peuvent être appelées sans devoir préciser l'objet *window*.

#### Une méthode de l'objet window

```
window.alert ("Coucou");
```

#### L'objet window est sous-entendu

```
alert ("Coucou");
```

### IV-B - L'objet window.navigator

Cet objet contient les propriétés du navigateur (nom, version, langue, etc...). On utilise de moins en moins souvent cet objet pour identifier la navigateur de l'utilisateur car certaines propriétés peuvent être modifiées (cas d'Opera). On utilisera plutôt certaines propriétés de l'objet *document* pour réaliser ces tests.

### IV-C - L'objet window.document

L'objet *document* regroupe toutes les méthodes de gestion de la page Web. Ses fonctions permettent de cibler un objet (un paragraphe par exemple) pour modifier ses attributs.

Pour modifier les attributs d'un élément (paragraphe, lien, etc...), celui-ci doit être au préalable identifié par un identifiant unique (attribut *id*). L'objet est ensuite ciblé grâce à la méthode *getElementById()* contenue dans l'objet *document*. Si l'élément possédant cet *id* n'existe pas la méthode renvoie *null*.

#### Utilisation des objets contenu dans le document


```
//objet contient toutes les propriétés de idElement
var monElement = document.getElementById("idElement");
```

### Utilisation des objets contenu dans le document

```
//pour modifier la taille de la police
monElement.style.fontSize = "12px";

//pour modifier la police
monElement.style.fontFamily = "Arial";

//pour modifier le contenu du paragraphe (balises div, span, p et body uniquement).
monElement.innerHTML = "Salut tout le monde !!";
```

 Les attributs définis dans les feuilles de styles (CSS) peuvent être modifiés par l'objet style de l'élément. Cependant, cela ne modifie pas la valeur de la feuille de style qui reste la valeur par défaut.

## V - Manipuler la page - Les grandes lignes du DOM

### V-A - Représentation de la page

Quelques liens :

 [FAQ Utilisation du DOM](#)

 [Document du W3C](#)

 [Manipulation des nœuds du document \(selfhtml.org\)](#)

### V-A-1 - Propriétés des nœuds

#### V-A-1-a - Propriétés

Le DOM (Document Object Model) est un modèle standardisé par le W3C (World Wide Web Consortium). Ce modèle propose de représenter un document sous la forme d'un arbre. Toutes les balises HTML sont donc des nœuds de l'arbre et les feuilles sont soit des balises sans contenu, soit le texte de la page HTML.

Propriétés	Commentaires
childNodes	nœuds enfants
firstChild	premier nœud enfant
lastChild	dernier nœud enfant
nextSibling	prochain nœud d'un type (nœud de même niveau)
parentNode	nœud parent
previousSibling	nœud précédent d'un type (nœud de même niveau)
nodeName	nom du nœud
nodeValue	valeur / contenu du nœud
nodeType	type du nœud (cf. ci-dessous)

Types de nœuds :

- |                                  |   |
|----------------------------------|---|
| 1 - Nœud élément                 | 7 - Nœud pour instruction de traitement |
| 2 - Nœud attribut                | 8 - Nœud pour commentaire               |
| 3 - Nœud texte                   | 9 - Nœud document                       |
| 4 - Nœud pour CDATA              | 10 - Nœud type de document              |
| 5 - Nœud pour référence d'entité | 11 - Nœud de fragment de document       |
| 6 - Nœud pour entité             | 12 - Nœud pour notation                 |

 **Document W3C sur les nœuds**

 **Plus d'informations sur le type des nœuds (selfhtml.org)**

## V-A-1-b - Remarques sur innerHTML et nodeValue

innerHTML est une instruction qui permet de modifier le contenu d'une balise ou d'insérer un objet dans la page.

### Insertion d'une image en utilisant innerHTML

```
//Ciblage du paragraphe
var MonParagraphe = document.getElementById("idPg");
//Modification de son contenu
MonParagraphe.innerHTML = "<img src='imageInseree.gif' /> Mon nouveau texte";
```

Une image sera insérée dans le paragraphe. Néanmoins, cette méthode présente quelques inconvénients lorsqu'il s'agit de modifier le contenu d'un formulaire (balise *form*). Lors de l'envoi du formulaire, les valeurs des objets créés via innerHTML ne sont pas toujours transmises au serveur. C'est pour cette raison qu'il est préférable d'utiliser les méthodes gérant les nœuds.

### Insertion d'une image en utilisant nodeValue

```
//Ciblage du paragraphe
var e = document.getElementById("idPg");

//Création de l'image
var i = document.createElement("img");

//Source de l'image
i.src = "imageInseree.gif";

//Modifiacion du texte (noeud #text)
e.firstChild.nodeValue ="mon nouveau texte";

//Ajout de l'image avant le texte
e.insertBefore(i,e.firstChild);
```

## V-A-2 - Exemples d'arbres DOM sous divers navigateurs

Soit une page Web définie par le code ci-dessous.

### Représentation de la page Web

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html>
  <head>
    <title>DOM</title>
  </head>

  <body>

    <div id="idP1" style="border:1px solid #AAAAAA">
      <h4>DIV 1</h4>
      bla bla bla bla bla bla bla bla bla bla bla
      <span>bla bla bla bla bla bla bla</span>
      bla bla bla bla bla bla bla bla
      <a href="">bla bla bla bla bla bla bla bla bla</a>
    </div>

    <div id="idP2">
      <h4>DIV 2</h4>
      bla bla bla bla bla bla bla bla
    </div>
```

Représentation de la page Web

```
</body>

</html>
```

V-A-2-a - Arborescence selon Firefox

La page ci-dessus est représentée sous Firefox selon ce schéma :

Arbre DOM de Firefox v1

```
|-- HTML
|-- HEAD
|  |-- TITLE
|  |-- #text
|-- #text (0)
|-- BODY
|  |-- #text (1)
|  |-- DIV id = "idP1"
|    |-- H4
|    |-- #text
|    |-- #text (2)
|    |-- SPAN
|    |-- #text
|    |-- A
|    |-- #text (3)
|    |-- #text (4)
|  |-- DIV id = "idP2"
|    |-- H4
|    |-- #text
|    |-- #text (5)
|    |-- #text (6)
```

Donc d'après cet arbre DOM, nous avons :

	BODY	DIV (idP1)	DIV (idP2)
<b>childNodes</b>	#text (1) ; DIV ; #text (4) ; DIV ; #text (6)	H4 ; #text (2) ; SPAN ; #text ; A ; #text (3)	H4 ; #text (5)
<b>firstChild</b>	#text (1)	H4	H4
<b>lastChild</b>	#text (6)	#text (3)	#text (5)
<b>nextSibling</b>	<b>inexistant</b>	#text (4)	#text (6)
<b>parentNode</b>	HTML	BODY	BODY
<b>previousSibling</b>	#text (0)	#text (1)	#text (4)

V-A-2-b - Arborescence selon Firefox v2

Arbre DOM de Firefox v2

```
|-- HTML
|-- HEAD
|  |-- TITLE
|  |-- #text (0)
|-- BODY
|  |-- #text (1)
|  |-- DIV id = "idP1"
|    |-- #text (2)
|    |-- H4
|    |-- #text
|    |-- #text (3)
|    |-- SPAN
|    |-- #text
|    |-- #text (4)
```

**Arbre DOM de Firefox v2**

```

|-- A
  |-- #text
  |-- #text (5)
  |-- #text (6)
  |-- DIV id = "idP2"
    |-- #text (7)
    |-- H4
    |-- #text
    |-- #text (8)
    |-- #text (9)
    
```

D'après cet arbre :

	BODY	DIV (idP1)	DIV (idP2)
<b>childNodes</b>	#text (1) ; DIV (idP1) ; #text (6) ; DIV (idP2) ; #text (9)	#text (2) ; H4 ; #text (3) ; SPAN ; #text (4) ; A ; #text (5)	#text (7) ; H4 ; #text (8)
<b>firstChild</b>	#text (1)	#text (2)	#text (7)
<b>lastChild</b>	#text (9)	#text (5)	#text (8)
<b>nextSibling</b>	<b>inexistant</b>	#text (6)	#text (9)
<b>parentNode</b>	HTML	BODY	BODY
<b>previousSibling</b>	HEAD	#text (1)	#text (6)

**V-A-2-c - Arborescence selon Internet Explorer v5**

Internet Explorer supprime les nœuds vides au moment de la création de la page.

**Arbre DOM d'Internet Explorer**

```

|-- HTML
  |-- HEAD
  |-- TITLE
  |-- BODY
    |-- DIV id = "idP1"
      |-- H4
      |-- #text (1)
      |-- SPAN
      |-- #text (2)
      |-- A
      |-- #text (3)
    |-- DIV id = "idP2"
      |-- H4
      |-- #text (4)
    
```

Donc d'après cet arbre, nous avons :

	BODY	DIV (idP1)	DIV (idP2)
<b>childNodes</b>	DIV ; DIV	H4 ; #text (1) ; SPAN ; #text (2) ; A ; #text (3)	H4 ; #text (4)
<b>firstChild</b>	DIV (idP1)	H4	H4
<b>lastChild</b>	DIV (idP2)	#text (3)	#text (4)
<b>nextSibling</b>	<b>inexistant</b>	DIV (idP2)	<b>inexistant</b>
<b>parentNode</b>	HTML	BODY	BODY
<b>previousSibling</b>	HEAD	<b>inexistant</b>	DIV (idP1)

## V-B - Création, insertion, suppression d'un nœud

### V-B-1 - Méthodes JavaScript pour la gestion des nœuds

Quelques fonctions permettant de gérer les nœuds du document.

Méthodes	Commentaires
createElement()	Méthode pour créer un nouvel élément HTML dans le document (div, p, span, a, form, input, etc...).
createTextNode()	Méthode pour créer un nœud texte.
appendChild()	Pour ajouter l'élément créé dans le document. L'élément sera ajouté comme étant le dernier nœud enfant d'un élément parent.
insertBefore()	Pour ajouter l'élément créé avant un autre nœud.
removeChild()	Pour supprimer un nœud.

### V-B-2 - Exemple

**Source** [Lien vers le script complet](#)

Soit un formulaire dans lequel on trouve un groupe d'éléments :

- trois textes : « Votre texte » ; « Vos options » ; « La suite »
- deux boîtes de textes ;
- une liste d'options.

ET un bouton qui permet d'ajouter à ce formulaire un nouveau groupe d'éléments identique au précédent. Ce nouveau groupe sera inséré avant le bouton.

Aspect final du formulaire :

Votre texte :  Vos options :  La suite :

*Aspect final du formulaire*

### V-B-2-a - Arborescence du document

#### Arborescence du document HTML

```

|-- BODY
|-- FORM id = "idFormulaire"
  |-- #text : Votre texte :
  |-- INPUT value = "" type = "text"
  |-- #text : Vos options :
  |-- SELECT size = "1"
    |-- OPTION value = "" text="Votre choix"
    |-- OPTION value = "valeur1" text="Option 1"
    |-- OPTION value = "valeur2" text="Option 2"
    |-- OPTION value = "valeur3" text="Option 3"
    |-- OPTION value = "valeur4" text="Option 4"
  |-- #text : La suite :
  
```



## Arborescence du document HTML

```
|-- INPUT value = "" type = "text"  
|-- BR  
|-- BR  
|-- INPUT value = "Ajouter un élément" type = "button" id = "idBouton"  
    onclick = "addLigne()"  
|-- BR
```

## V-B-2-b - Création des éléments

Dans un premier temps nous allons créer tous les éléments qui seront dans la page :

### Création de deux *input* de type texte.

Nous utilisons la fonction *createElement()*.

#### Syntaxe de la méthode *createElement()*

```
document.createElement("élément HTML à créer");
```

#### Création des *input* de type text

```
var elInput = new Array();  
for (i=0;i<2;i++)  
{  
    elInput[i] = document.createElement("input");  
    elInput[i].type = "text";  
}
```

**Création des trois *TextNode*** « Votre texte » ; « Vos options » ; « La suite » : nous utilisons la fonction *createTextNode()*.

#### Syntaxe de la méthode *createTextNode()*

```
document.createTextNode("Texte du n°ud");
```

#### Création des *TextNode*

```
// Tableau dans lequel seront stockés les éléments  
var elTxt = new Array();  
  
//Textes des éléments  
var tabTxt = new Array("Votre texte : ", "Vos options : ", "La suite : ");  
  
for (i=0; i<tabTxt.length; i++)  
{  
    elTxt[i] = document.createTextNode(tabTxt[i]);  
}
```

**Création de la liste déroulante** : le code ci-dessous ne permet de créer **que** la balise *select*. Nous verrons plus loin comment créer et ajouter les options à la liste.

#### Création de la liste déroulante

```
//création de l'élément select  
var elSelect = document.createElement("select");  
  
//nombre d'éléments visibles
```

### Création de la liste déroulante


```
elSelect.size = "1";
```

### Création des options de la liste déroulante.

Les options sont des objets de la liste. Pour les créer nous n'utilisons pas la méthode `createElement()`. Nous allons créer ces objets en utilisant la syntaxe suivante :

### Créer une option

```
new Option("Text", "Value", "defaultSelected true / false", "selected true / false");
```

 Les objets sont créés mais ne sont pas ajoutés à la liste pour autant.

### Création des options d'une liste

```
//Tableau contenant les options de la liste
var elOption = new Array(
    new Option("Votre choix", "", false, false),
    new Option("Option 1", "valeur1", false, false),
    new Option("Option 2", "valeur2", false, false),
    new Option("Option 3", "valeur3", false, false),
    new Option("Option 4", "valeur4", false, false)
);
```

**Création d'une ligne** pour séparer chaque groupe. Une feuille de style s'appliquera sur cette ligne.

### Création d'une ligne

```
var ligne = document.createElement("hr");
ligne.className= "styleLigne";
```

## V-B-2-c - Insertion des objets dans le document

Maintenant nous allons ajouter ces éléments au document. Ces éléments seront ajoutés dans le formulaire juste au-dessus du bouton "Ajouter un élément". Nous allons avoir besoin de l'objet formulaire (élément parent) et de l'objet bouton (référence).

**La fonction utilisée est `insertBefore(e1, e2)` avec :**

- `e1` : le nouvel élément enfant à insérer ;
- `e2` : un élément enfant avant lequel le nouvel élément enfant doit être inséré.

### Ajout des éléments dans le document

```
//Appel des objets formulaire et bouton
var elForm = document.getElementById("idFormulaire");
var objBouton = document.getElementById("idBouton");

//Ajout de la ligne de séparation
elForm.insertBefore(ligne, objBouton);

elForm.insertBefore(elTxt[0], objBouton); //1er texte
elForm.insertBefore(elInput[0], objBouton); //1er INPUT

elForm.insertBefore(elTxt[1], objBouton); //2ème texte
```


### Ajout des éléments dans le document

```
elForm.insertBefore(elSelect, objBouton); //Ajout du select

//Ajout dans le select des options (1)
for (i=0;i<elOption.length;i++)
{
    elSelect.options.add(elOption[i]);
}

elForm.insertBefore(elTxt[2], objBouton); //3ème texte
elForm.insertBefore(elInput[1], objBouton); //2ème INPUT

//Saut de ligne 1
elForm.insertBefore(document.createElement("br"), objBouton);
//Saut de ligne 2
elForm.insertBefore(document.createElement("br"), objBouton);
```

 (1) Internet Explorer : il faut insérer le select dans le document avant d'ajouter les options.

## V-C - Ajax

AJAX est une méthode de programmation des pages Web qui s'appuie sur des scripts JavaScript en utilisant l'objet XMLHttpRequest (aussi appelé XHR). Celui-ci permet de réaliser des requêtes vers le serveur de manière asynchrone et ainsi de mettre à jour tout ou partie du contenu d'une page Web.

 [FAQ Qu'est-ce qu'Ajax ?](#)

[FAQ La rubrique Ajax](#)

 [Les ressources Ajax sur developpez.com](#)

## VI - Conclusion - Liens divers - Remerciements

### VI-A - Conclusion

Le JavaScript facilite la navigation au sein d'un site Web. Au fil du temps, des améliorations et des fonctionnalités sont apportées (Ajax, par exemple). Cependant, compte-tenu des différences d'interprétation d'un navigateur à un autre, il est conseillé de tester chaque script sur différents navigateurs voire sous différents systèmes d'exploitation pour assurer une portabilité du code.

### VI-B - Liens

#### Les liens developpez.com

-  [La section JavaScript de www.developpez.com](#)
- [Source Les codes sources JavaScript](#)
- [FAQ La FAQ JavaScript](#)

#### Autres liens

-  [Le site de l'ECMA](#)
-  [Le site du World Web Consortium \(W3C\)](#)

## VI-C - Remerciements

Merci à **DenisC** et **Yogui** pour la relecture et leurs contributions dans la rédaction de cet article.  
Merci à **Bisûnûrs** pour sa relecture.