

## Présentation rapide du JavaScript

### Ressources utiles :

- Developpez.com : <http://javascript.developpez.com/>
- SelfHTML : <http://fr.selfhtml.org/javascript/index.htm>
- Comparatifs :
  - o <http://www.ja-sig.org/wiki/display/UP3/Javascript+Toolkit+Comparison>
  - o <http://ntt.cc/2008/02/13/the-most-complete-ajax-framework-and-javascript-libraries-list.html>
- Le DOM :
  - o <http://www.quirksmode.org/dom/intro.html>
  - o <http://domscripting.com>
  - o <http://fr.selfhtml.org/javascript/objets/index.htm>
  - o <http://fr.selfhtml.org/javascript/langage/gestevenements.htm>
- Débugage sous Firefox : <http://getfirebug.com/>

### Introduction

Le JavaScript est un langage de script introduit en 1995 par le navigateur NetScape 2.0, pour offrir des fonctionnalités de programmation côté client (navigateur). Aujourd'hui présent dans tous les navigateurs et relativement normalisé, on peut le considérer comme un complément essentiel, côté client, des applications web développées avec des langages côté serveur (PHP, ASP).

C'est un langage orienté objet, mais selon une philosophie très différente de Java ou C++, beaucoup plus libérale (pas de types forcés par exemple). Au-delà des fonctions offertes par les interpréteurs JS présents dans les navigateurs, qui sont nombreuses mais de bas niveau, l'utilisation de JS se développe désormais autour de bibliothèques de fonctions.

Ces bibliothèques vont offrir des fonctions de plus haut niveau, pour faciliter la création d'applications : interfaces, formulaires, design graphiques, effets visuels, interactions, widgets, etc.

### 1- Le JavaScript, langage de programmation interne à un navigateur Internet.

#### a- Insertion du code

Un script JS peut s'intégrer à une page HTML de deux façons : dans la page ou en lien. Naturellement la seconde solution est intéressante pour les scripts de grande taille, ou devant être générés / modifiés par ailleurs (en PHP par exemple).

Dans la page :

```
<script type="text/javascript">
    //votre code javascript ici
</script>
```

En lien :

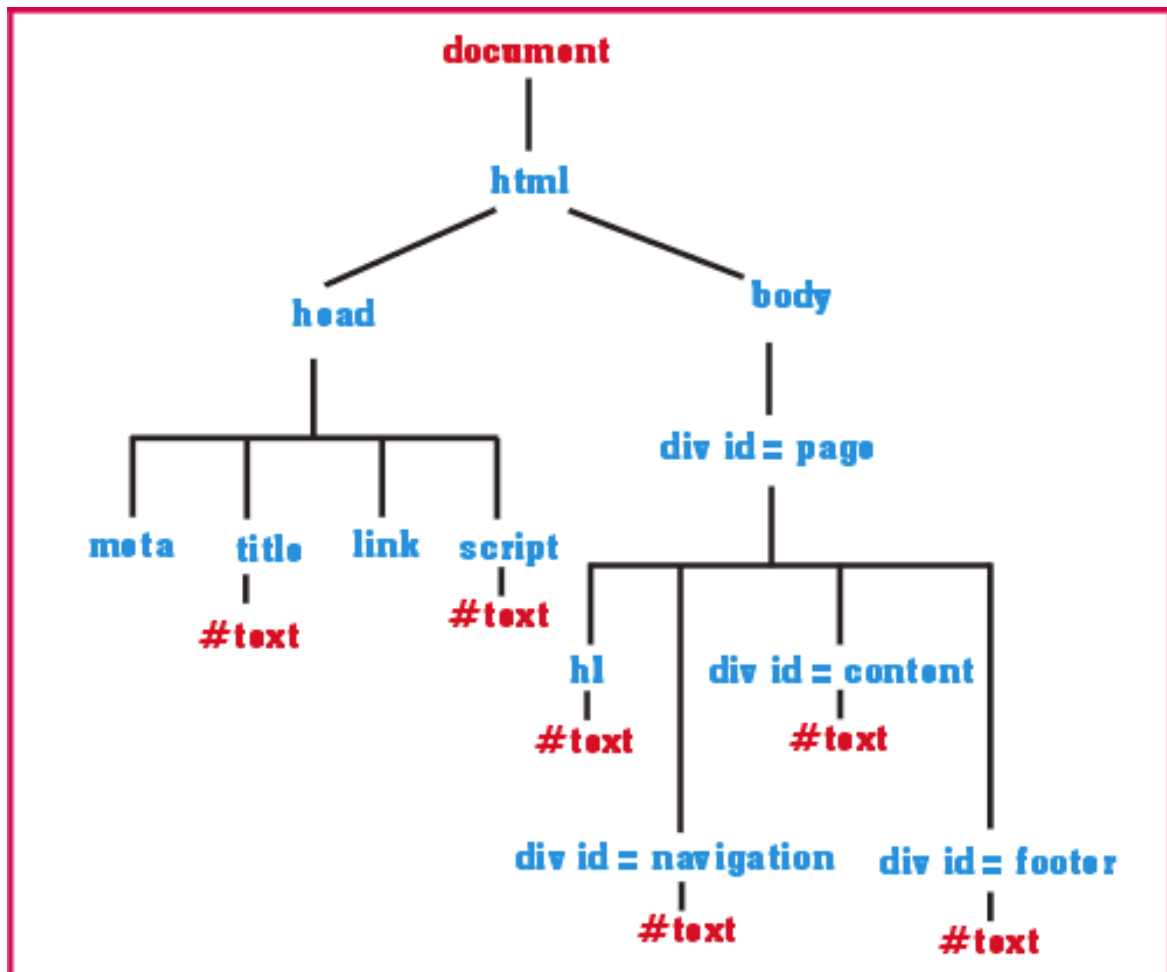
```
<script src="mon_script.js" type="text/javascript" />
```

Suivant l'endroit où le script est placé dans la page, il sera interprété pendant ou après le chargement de la page HTML. On trouve donc souvent des fonctions JS dans le bloc <HEAD> </HEAD>, en fin de fichier HTML, ou appellés par l'événement body.onload() ;

### **b- Interaction HTML / JS**

Le principe de fonctionnement du code JS c'est qu'il va interagir avec des éléments de la page HTML. C'est l'aspect du JS qui est très souvent négligé dans les présentations de ce langage : il s'appuie sur un modèle objet de la page HTML, nommé le DOM pour Document Object Model, qui est une spécification du W3C.

Ainsi, on va pouvoir utiliser des fonctions JS pour générer dynamiquement du contenu HTML, ou plus simplement, modifier ses caractéristiques (style, contenu).



Le DOM est un modèle objet hiérarchique qui va décrire et rendre accessible les éléments d'une page web (cf. schéma ci-dessus).

Par exemple pour écrire un texte dans une page HTML, on va utiliser la méthode write sur l'objet document :

```
<script language="javascript">
    document.write('bonjour');
</script>
```

Les fonctions JS peuvent être lancées directement comme ci-dessus, au moment où le script est interprété et exécuté, mais on peut aussi les faire réagir à des événements, c'est à dire des changements d'état d'objets et des actions de l'utilisateur (sur des formulaires par exemple).

### c- Bases du langage JS

Impossible ici de faire une vraie présentation des bases du langage JS, des ouvrages entiers y sont consacrés, on va simplement lister les principes de fonctionnement et les structures principales. Les instructions sont séparées par des points-virgules, comme en PHP.

#### Les variables

JS n'est pas un langage fortement typé, c'est à dire où il faut déclarer les variables d'un certain type avant de pouvoir les utiliser. Cette souplesse permet une plus grande facilité d'utilisation, mais elle peut parfois être source de confusion.

```
<script language="javascript">
    var s; // déclaration optionelle
    s = 'bonjour'; // pas de type
    document.write(s);
</script>
```

Les tableaux sont en JS des objets de type Array. On peut les déclarer sans les dimensionner (définir a priori la taille maximale de leur contenu). On accède au contenu d'un tableau en indiquant l'indice de l'élément entre crochets [].

```
<script language="JavaScript">
    var T = new Array(3);
    var S = "";
    for (var i = 0; i < T.length; i++){
        T[i] = i;
    }
    T[5] = "Pourtant, j'y suis";
    for (var i = 0; i < T.length; i++){
        S += "En " + i + " on trouve " + T[i] + "\n";
    }
    alert(S);
</script>
```

La classe JS Array possède des méthodes intéressantes pour trier et organiser les variables d'un tableau.

## Les fonctions

JS permet la définition de fonctions, qui peuvent ensuite être appelées dans le code, comme les fonctions du langage lui-même.

```
<script language="javascript">
function afficherSomme(a, b) {
    var s ;
    s = a + b ;
    document.write(s);
}
afficherSomme(1, 2) ;
</script>
```

De la même manière, on peut faire appel à des fonctions systèmes, comme la date et l'heure, ou récupérer des propriétés des éléments de la page par la biais du DOM.

Exemple, une image en rollover sur un lien :

```
<a href="url.php"
onMouseOver="document.ex1.src='../images/milieu2.gif'"
onMouseOut="document.ex1.src='../images/milieu.gif'">
    
</a>
```

Notez l'utilisation du DOM en allant chercher la propriété « src » de l'objet « ex1 » qui est une image, et des événements OnMouseOver et OnMouseOut.

Exemple : modification du style d'un élément :

```
<HTML>
<HEAD>
<script language="javascript">
function couleur(destination) {
    var p;
    p = document.getElementById(destination);
    p.style.background = 'blue';
}
</SCRIPT>
</HEAD>
<BODY>
    <p id='paragraphe'>
        Bonjour !
    </p>
    <FORM NAME="form4">
        <INPUT TYPE="button" NAME="but" VALUE="Action"
            onClick="couleur('paragraphe') ">
    </FORM>
</BODY>
</HTML>
```

## Les boucles et structures de contrôle

On retrouve en JS les structures de contrôle en boucle classiques :

- boucle for : `for (var i = 1 ; i < 9; i += 3) { corps boucle }`
- boucle `do { corps boucle } while (expression booléenne) ;`
- structure if :

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

Exemple : le contrôle de saisie dans un formulaire :

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

## 2- Le JavaScript, sous la forme de bibliothèques de fonctions

Le développement actuel de Javascript, et la course à la vitesse entre les navigateurs récents, tient surtout au développement d'applications Internet client-serveur dont la partie cliente est en fait une paeweb dans un navigateur. JS est alors le langage qui permet de faire fonctionner l'interface avec l'utilisateur, et de proposer un aspect graphique interactif plus développé que ce que HTML/CSS peuvent proposer.

Dans ce cadre, se sont construites des bibliothèques de fonctions qui rassemblent de manière conviviale différents outils graphiques et d'automatisation d'interfaces écrits en JS.

On parle aussi parfois de cadres d'application (frameworks en anglais), de boîtes à outils (toolboxes ou toolkits), et d'ensembles de widgets (éléments d'interface graphique).

L'idée est de proposer des fonctions avancées dans des fichiers JS liés aux pages HTML, ces derrières pouvant naturellement être générées en PHP par le serveur. Parmi ces fonctions, une des plus célèbres est l'AJAX, qui permet la communication entre le navigateur client et le serveur sans rechargement de la page, par l'utilisation d'appels JS invisibles pour l'utilisateur.

=> Aperçu de certaines bibliothèques (Prototype / script.aculo.us, ExtJS).

Certaines de ces bibliothèques sont orientées WebMapping, la plus connue aujourd'hui étant <http://openlayers.org/> Elle est de plus incluse dans une autre bibliothèque, GeoExt, qui vise à la coupler avec ExtJS (<http://extjs.com/>), une bibliothèque généraliste de développement d'applications web.

Le problème de ce fonctionnement avec des bibliothèques de fonctions, c'est que cela consiste en fait à utiliser des possibilités qui n'ont rien à voir avec les fonctions JS de base, et qui sont généralement uniques à ces bibliothèques. Donc il faut qu'elles soient bien documentées et présentées, pour que leur utilisation soit accessible aux débutants.

Par exemple les grands fournisseurs de sites cartographiques interactifs offrent un accès programmatif à leurs données et fonctions par le biais d'API en JS : Google Maps et Google Earth, Microsoft Live, Yahoo Maps. De même les sociétés qui proposent des logiciels serveurs commencent à offrir la possibilité de les coupler à des clients léger (navigateurs web) par l'utilisation de bibliothèques de fonctions JS : ESRI ArcGIS server et ArcWeb Services.

A côté d'OpenLayers et des API des fournisseurs de cartes interactives, qui se veut généraliste dans le domaine du webmapping, on peut trouver des bibliothèques dédiées à un type de source de données ou de présentation :

- WMS JS lib : <http://wms-map.sourceforge.net/>
- TimeMap (Google Maps + ligne temporelle) : <http://code.google.com/p/timemap/>

La bibliothèque MapStraction propose même une API unifiée pour se connecter à diverses fournisseurs de cartographie en ligne : <http://mapstraction.com/>