

Introduction au JavaScript

1. Définition

JavaScript est un langage de script qui, incorporé aux balises HTML ou XHTML, permet d'agrémenter la présentation et l'interactivité des pages web.

Détaillons et complétons cette définition.

Le JavaScript est un langage de script, distinct du langage HTML. Le JavaScript est en quelque sorte un "petit" langage de programmation dont les lignes de code viennent s'ajouter au langage HTML. Ce langage, inspiré plus ou moins directement du langage C, ne pose pas de problèmes de compréhension à ceux qui ont déjà touché à la programmation. Par contre, les utilisateurs issus de la bureautique ou de l'infographie risquent d'être un peu déroutés devant les éléments classiques de la programmation comme les variables, les instructions conditionnelles, les boucles, les fonctions et autres méthodes.

Le HTML avec sa philosophie de structuration de contenu est essentiellement statique. Très rapidement, le besoin s'est fait sentir d'y ajouter un peu de mouvement et surtout de l'interactivité avec l'utilisateur. C'est ce que permet le JavaScript.

Citons pour exemple :

- animer du texte ou des images ;
- réagir à l'action de la souris ;
- détecter le type et la version du navigateur ;
- vérifier la saisie dans les formulaires ;
- effectuer des calculs simples ;
- demander une confirmation ;
- afficher la date et l'heure ;
- gérer les menus de navigation ;
- rediriger le visiteur vers une autre page ;
- etc.

Les lignes de code du JavaScript sont gérées et exécutées directement par le navigateur lui-même, sans devoir faire appel aux ressources du serveur qui héberge la page. C'est ce qu'on appelle une application "côté client". Il existe d'autres langages plus évolués, comme le PHP ou l'ASP, qui eux traitent les informations sur le serveur avant de les renvoyer sous forme de page HTML au navigateur de l'utilisateur. Ce sont les applications "côté serveur". Ces dernières sortent du cadre fixé à notre étude. Signalons simplement qu'il existe bien une version "côté serveur" de JavaScript mais que nous n'aborderons pas. Nous nous limitons donc à la version la plus répandue et la plus abordable, soit le JavaScript "côté client".

La tradition veut que l'on précise dans une introduction au JavaScript que celui-ci (malgré la confusion qui peut naître à partir du nom) n'a rien de commun avec le langage Java. Ce dernier est un véritable langage de programmation plus performant et bien plus complexe, dont le code est compilé sous forme d'applets Java. Celles-ci peuvent s'exécuter à partir d'une page web, pour autant que la Machine Virtuelle Java soit installée sur le poste du visiteur. Pour rappel, le code du JavaScript inclus dans la page HTML est quant à lui directement "interprété" par le navigateur au moment de son exécution.

Conséquence directe de l'inclusion du JavaScript dans les pages HTML, le code JavaScript sera visible par une simple

consultation du code source de la page. Ceci peut faire le désespoir des concepteurs débutants mais ne constitue pas vraiment un handicap, étant donné que le JavaScript n'est destiné qu'à de petites applications, finalement bien connues de tous. Pour les applications plus professionnelles, celles-ci seront réalisées avec des langages plus évolués côté serveur. Ce qui préservera alors la confidentialité du code.

Le JavaScript est un langage orienté objet. Nous y reviendrons en détail plus avant dans notre étude. Notons simplement que JavaScript permet d'accéder directement aux différents éléments du document et de les manipuler. Comme principaux éléments ou objets de la page HTML, on peut citer la fenêtre du navigateur, le document HTML, les images, les cadres, les formulaires, les éléments de formulaire ou tout autre élément que vous aurez au préalable identifié.



L'objectif de cette partie n'est pas de faire de vous des programmeurs émérites en JavaScript mais de fournir des bases solides qui vous permettront de comprendre le fonctionnement de la technique AJAX ainsi que les manipulations de données qu'elle induira.

2. Un bref historique

Le langage JavaScript est apparu en 1995 avec le navigateur Netscape 2. Le JavaScript 1.0 était né. Les possibilités offertes par ce langage ont très vite séduit les utilisateurs et les développeurs du Web, créant à l'époque un vif intérêt sinon une petite révolution.

Du côté de chez Microsoft, alors un peu à la traîne sur le plan de l'Internet, on a très rapidement (1996) fait en sorte que le JavaScript soit reconnu par Internet Explorer 3. En effet, il ne fallait pas de licence pour utiliser le JavaScript. Par contre, dans le climat de guerre froide entre les deux constructeurs, il n'était pas question d'acheter une licence à Netscape pour en adapter le code. Est né alors JScript, variante du JavaScript selon Microsoft, qui reprend la plupart des fonctions du JavaScript et en élargit le champ d'application.

C'était déjà mal parti sur le plan de la compatibilité. Heureusement, les deux firmes acceptèrent de participer à une standardisation. L'organisme choisi fut l'ECMA (*European Computer Manufacturers Association*). En 1997, le JavaScript a été standardisé sous le vocable ECMAScript sous la référence ECMA-262. En fait, cette standardisation n'a porté que sur la syntaxe de base, n'apportant qu'une compatibilité minimale et chacun est reparti vaquer à ses occupations.

Avec Netscape 3 (fin 1996) est apparue la version JavaScript 1.1. Avec Netscape 4 (1997), la version 1.2. Ces versions furent également reconnues par les versions suivantes d'Internet Explorer tandis que JScript voyait lui aussi apparaître de nouvelles versions.

Au fil des années, ce qui n'était qu'un petit langage de programmation est devenu de plus en plus évolué au fil des versions 1.3, 1.4, 1.5, 1.6, 1.7 et 1.8.

Le JavaScript a connu une avancée notoire avec l'apparition du DOM. Ce dernier permet d'accéder ou de mettre à jour le contenu, la structure ou le style de documents HTML. Le document peut ensuite être traité et les résultats de ces traitements être réincorporés dans le document tel qu'il sera présenté. Sa standardisation par le W3C fut largement adoptée par tous les navigateurs, ce qui atténua quelque peu le problème d'interopérabilité des scripts.

Le concept du Web 2.0, dans ses objectifs d'une meilleure facilité d'utilisation et d'une plus grande ergonomie, a quant à lui encore renforcé l'interactivité des pages et la demande d'applications plus riches. Et voilà le JavaScript propulsé comme un élément incontournable dans le développement d'applications web.

La meilleure preuve de ce repositionnement du JavaScript est assurément l'apparition de nouveaux moteurs JavaScript dans les navigateurs récents. Que ce soit Google Chrome avec son nouveau moteur JavaScript Open Source V8, Opera avec le projet Carakan, Firefox avec TraceMonkey, tous cherchent à améliorer (et parfois de façon assez sensible) le traitement du JavaScript. Internet Explorer a lui aussi adopté un nouveau moteur JavaScript

appelé Chakra.

Devant la demande croissante d'applications réactives et les éternels problèmes de compatibilité, on a vu fleurir des frameworks JavaScript qui facilitent l'écriture du code JavaScript tout en le rendant compatible avec les navigateurs du marché.

Dernièrement, le JavaScript a été reconnu comme un élément à part entière du concept HTML5 avec le HTML proprement dit et les feuilles de style CSS. De nombreuses API JavaScript ont vu le jour et d'autres devraient être publiées dans les prochains mois. Ces API dédiées à des tâches spécifiques comme la géolocalisation, le stockage de données, le glisser/déposer, etc. devraient aussi améliorer la compatibilité du JavaScript.

3. Les limites du JavaScript

Presque toutes les pages web intègrent quelques éléments de JavaScript. Ce qui ne semble plus créer de réels problèmes de sécurité, d'où son succès auprès des développeurs de sites web et des internautes.

Mais cette relative sécurité a un prix qui se traduit par de sévères limites techniques.

La principale limite de JavaScript est qu'il ne permet pas de lire et d'écrire sur le disque dur ou tout autre périphérique du visiteur. La seule exception est celle qui permet à JavaScript de lire et d'écrire sur le disque dur uniquement dans la zone réservée aux cookies. C'est la seule interaction que JavaScript peut avoir avec votre disque dur.

En outre, aucune instruction n'est prévue en JavaScript pour générer une connexion vers un autre ordinateur ou un serveur. Cela n'est possible que par la création d'un objet XMLHttpRequest qui est un élément indispensable à la mise en place d'applications de type AJAX (voir chapitre L'objet XMLHttpRequest).

Il ne sera donc pas possible de concevoir en JavaScript des applications telles qu'un forum de discussion, un sondage ou une boutique en ligne. Toutes ces applications nécessitent un langage plus puissant, généralement implanté côté serveur, comme le PHP ou l'ASP.

Même si cela est en partie excessif, il faut être conscient que le JavaScript est un "petit" langage de programmation et qu'il n'a été conçu que pour compléter le HTML normal.

4. Des outils pour le JavaScript

Le code source des pages web présente l'avantage de ne pas nécessiter de logiciels coûteux. JavaScript ne déroge pas à la règle.

Pour l'apprentissage du JavaScript, il vous faut :

- un simple éditeur de texte ;
- un navigateur compatible JavaScript ;
- des connaissances du langage HTML.

Le JavaScript, comme la plupart des langages de programmation, s'écrit en texte brut. Ainsi un simple éditeur de texte comme le Bloc-notes de Windows (*notepad*) fait parfaitement l'affaire. L'utilisation de la fenêtre de code d'un éditeur HTML est également possible.

Le JavaScript étant interprété par le navigateur, n'importe quel navigateur compatible JavaScript peut être adopté. Ce qui est le cas de la plupart des navigateurs du marché. Google Chrome, Internet Explorer et Firefox, retenus

dans le cadre de cet ouvrage, sont bien entendu parfaitement adaptés au JavaScript. Veillez cependant à ce que la fonction JavaScript ne soit pas désactivée dans les options de votre navigateur.


Puisque le code JavaScript vient se greffer dans le code source de la page, une connaissance approfondie des balises HTML ou XHTML est recommandée, sinon indispensable.

5. Le JavaScript et le HTML

Le JavaScript est interprété par le navigateur. Il faut donc l'informer que le code qu'il risque de rencontrer est du JavaScript.

Le HTML5 introduit une grande simplification par rapport au HTML 4.0 ou le XHTML. En effet, le JavaScript est devenu le langage de script par défaut. Le code JavaScript sera signalé au navigateur en entourant simplement celui-ci des balises `<script> ... </script>`.

```
<script>
Code JavaScript
</script>
```

 La balise `<script type="text/javascript">` utilisée précédemment est devenue inutile car le JavaScript est le langage de script par défaut du HTML5. Spécifier que le code contenu entre les balises `<script>` est du JavaScript est une information redondante.

Il en est de même avec la balise meta `<meta http-equiv="Content-Script-Type" content="text/javascript">` qui n'a plus de raison d'être.

6. Un premier script

Saisissons le code suivant dans un éditeur de texte :

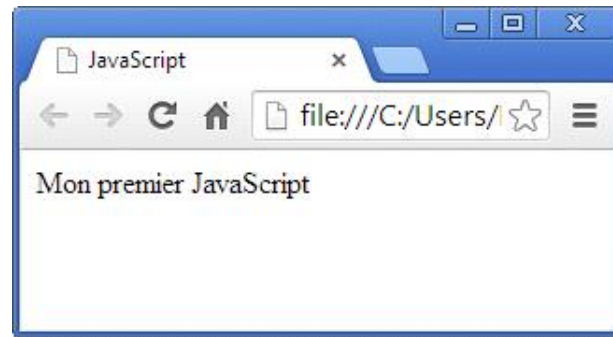
```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>JavaScript</title>
<meta charset="UTF-8">
</head>
<body>
<script>
document.write("Mon premier JavaScript");
</script>
</body>
</html>
```

Commentaires :

- Le doctype n'a aucune influence sur l'exécution du JavaScript.
- L'instruction `document.write("Mon premier JavaScript");` demande d'écrire (*write*) le texte repris entre les parenthèses dans le document.
- Remarquez que l'instruction se termine par un point-virgule.

On enregistre ce fichier au format htm.

Ensuite, on affiche la page locale dans le navigateur.



7. Le JavaScript interne

On peut se poser la question : "Où inclure les balises `<script> ... </script>` dans le document HTML ?".

Il n'y a hélas, pas de règle précise qui stipule à quel endroit d'un fichier HTML le code JavaScript doit être défini. La seule réponse est "Là où le navigateur en aura besoin" !

Pour rappel, la page HTML est chargée par le navigateur de façon séquentielle (de haut en bas). Il lit d'abord l'en-tête (le contenu des balises `<head>`) et ensuite le corps (`<body>`), ligne par ligne. Tous les éléments du "body" sont interprétés dans l'ordre d'écriture des balises.

Si on fait appel à du code JavaScript, alors que celui-ci n'a pas encore été défini dans la page et donc interprété par le navigateur, ce dernier ne peut, en toute logique, pas le prendre en charge (erreur sur la page).

On peut ainsi retrouver le code JavaScript à plusieurs endroits de la page :

- Il n'est pas rare que du code JavaScript soit inclus, totalement ou en partie, dans les balises `<head>`. En effet, ce code sera alors lu en premier par le navigateur et le concepteur peut en toute sécurité faire appel à ces fragments de code, n'importe où à l'intérieur de la balise `<body>` car il sera déjà disponible.
- On retrouve parfois du code JavaScript immédiatement après la balise `<body>` lorsque celui-ci est plus spécifique au corps du document.
- Si le JavaScript traite les données d'un formulaire, le code peut apparaître après la balise `<form>`.
- Des instructions comme `document.write("...");` peuvent être incluses dans le texte de la page à l'endroit souhaité.
- Etc.



Le code JavaScript doit être inséré avant ou au plus tard au moment où il devra être exécuté.

En outre, depuis que le JavaScript est reconnu comme le langage de script par défaut de la plupart des navigateurs, du code JavaScript peut être inclus directement dans la balise HTML, fonctionnant alors comme un attribut de celle-ci.

Cette façon de procéder n'est pas adoptée pour un code programme compliqué, mais seulement pour un appel à des événements, méthodes, fonctions déterminés.

Quelques exemples :

```
<a onclick="javascript:history.back();">Retour page précédente</a>
```

Avec l'objet `history` de JavaScript, vous avez accès aux pages web qui ont été visitées par l'utilisateur et qui sont sauvegardées dans l'historique du navigateur. Avec la méthode `back()`, on demande, au clic sur le lien, de remonter d'une position dans l'historique, revenant ainsi à la page précédente.

```
<h1 onmouseover="history.back()">Retour page précédente</h1>
```

Ici, on demande de revenir à la page précédente par le simple survol de la souris (`onmouseover`).

```
<form>
<input type="button" value="retour" onclick="history.back()">
</form>
```

Quand l'utilisateur clique sur le bouton de formulaire, on revient à la page précédente.

8. Le JavaScript externe

Il est également possible de noter le code JavaScript dans un ou des fichiers séparés de la page HTML. Ce qui présente l'avantage de clarifier le code du document HTML et de respecter la règle de séparation du contenu et de la présentation. Il est ainsi possible d'appeler le même code JavaScript à partir de fichiers HTML séparés sans avoir à le réécrire.

Ce fichier externe est un fichier de texte brut encodé, par exemple, avec le Bloc-notes de Windows. Ce type de fichier comporte l'extension `.js`. Il contient uniquement du code JavaScript, et donc sans les balises `<script>`.

Ce fichier est appelé dans la page HTML par :

```
<script src="fichier_externe.js"></script>
```

Cette balise d'appel se place généralement entre les balises `<head> ... </head>`.

Reprenons notre exemple de la section Un premier script de ce chapitre.

a. Le fichier externe

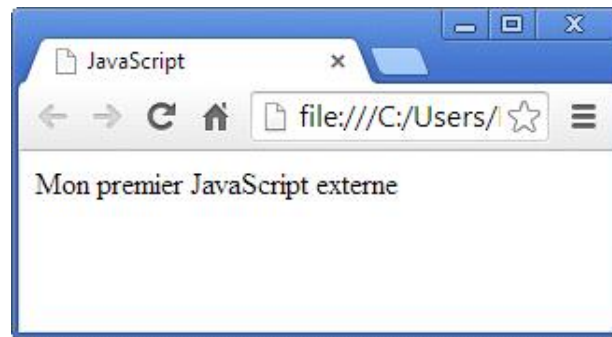
Ce fichier externe, que nous appellerons `write.js`, comporte la seule ligne de code :

```
document.write("Mon premier JavaScript externe");
```

b. La page HTML

```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>JavaScript</title>
<meta charset="UTF-8">
```

```
<script src="write.js"></script>
</head>
<body>
</body>
</html>
```



9. Quelques conseils pour le débogage

Vous serez inévitablement dérouté lors de vos premières applications en JavaScript. En effet, les navigateurs permettent beaucoup de "souplesse" dans le code HTML et les feuilles de style CSS. Avec le JavaScript, on passe à la rigueur d'un langage de programmation. La moindre erreur sera sanctionnée ! Et une erreur est si vite arrivée... Un point à la place d'une virgule, une faute de frappe, une majuscule absente ou superflue, une parenthèse manquante, un guillemet au lieu d'une apostrophe, une accolade oubliée, voilà un florilège de petites erreurs qui empêchent l'exécution d'un script.


Heureusement, les navigateurs récents intègrent des outils de développement qui sont d'une aide appréciable pour déboguer vos applications.

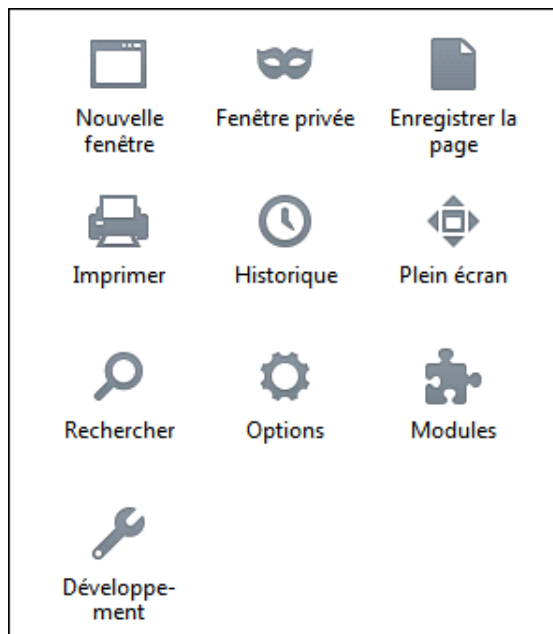
a. Google Chrome

Les outils de développement de Google Chrome sont accessibles par le raccourci-clavier [Ctrl][Shift] **I**. Ils se révèlent très complets et très pertinents. Pour ce qui nous concerne pour le JavaScript, les erreurs sont signalées dans l'onglet **Console**. On peut regretter que les erreurs soient signalées de façon peu explicite.

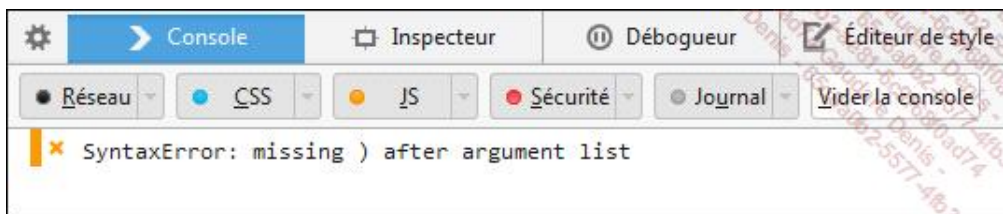


b. Firefox

Depuis quelques versions, Firefox intègre des outils de développement qui auparavant étaient fournis par l'add-on Firebug. On peut y accéder par le raccourci-clavier [Ctrl][Shift] **S** ou par l'icône  - **Développement - Débogueur**. Ici aussi, c'est l'onglet **Console** qui nous intéresse.

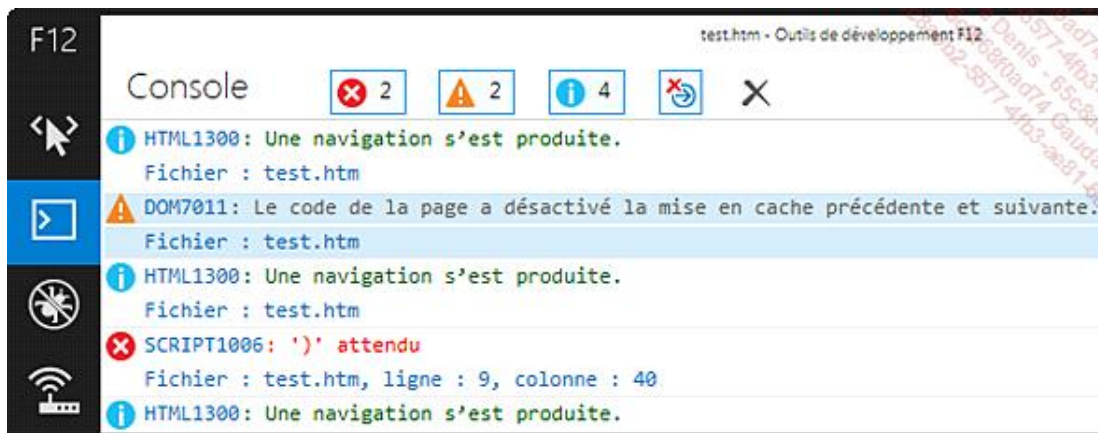


Développement web	
Outils de développement	Ctrl+Maj+I
Console web	Ctrl+Maj+K
Inspecteur	Ctrl+Maj+C
Débogueur	Ctrl+Maj+S
Éditeur de style	Maj+F7
Profileur	Maj+F5
Réseau	Ctrl+Maj+Q
Barre de développement	Maj+F2



c. Internet Explorer

Dans sa version 11 d'Internet Explorer, Microsoft a totalement remanié ses outils de développement. On y accède par la touche [F12]. C'est toujours l'onglet **Console** qui retiendra notre attention pour déboguer le JavaScript. Si sa présentation graphique est des plus réussies, son utilisation semble moins intuitive.



- Ces outils de développement ne font que vérifier la syntaxe des scripts. Il n'y a rien de plus déconcertant qu'un script sans erreurs mais qui ne réalise pas l'action souhaitée !