Support de cours

Introduction au JavaScript

Réaliser par : Nene SYLLA

Introduction au Javascript

Comme tout langage de programmation, le Javascript possède quelques particularités : sa syntaxe, son modèle d'objet, etc. En clair, tout ce qui permet de différencier un langage d'un autre. D'ailleurs, vous découvrirez rapidement que le Javascript est un langage relativement spécial dans sa manière d'aborder les choses. Cette partie est indispensable pour tout débutant en programmation *et même pour ceux qui connaissent déjà un langage de programmation* car les différences avec les autres langages sont nombreuses.

Avant d'entrer directement dans le vif du sujet, ce chapitre va vous apprendre ce qu'est le Javascript, ce qu'il permet de faire, quand il peut ou doit être utilisé et comment il a évolué depuis sa création en 1995.

Qu'est-ce que le Javascript ?

Le Javascript est un langage de programmation de scripts orienté objet.

Un langage de programmation

Tout d'abord, un **langage de programmation** est un langage qui permet aux développeurs d'écrire du **code source** qui sera analysé par l'ordinateur.

Un **développeur**, ou un programmeur, est une personne qui développe des programmes. Ça peut être un professionnel (un ingénieur, un informaticien ou un analyste programmeur) ou bien un amateur.

Le code source est écrit par le développeur. C'est un ensemble d'actions, appelées **instructions**, qui vont permettre de donner des ordres à l'ordinateur afin de faire fonctionner le programme. Le code source est quelque chose de caché, un peu comme un moteur dans une voiture : le moteur est caché, mais il est bien là, et c'est lui qui fait en sorte que la voiture puisse être propulsée. Dans le cas d'un programme, c'est pareil, c'est le code source qui régit le fonctionnement du programme.

En fonction du code source, l'ordinateur exécute différentes actions, comme ouvrir un menu, démarrer une application, effectuer une recherche, enfin bref, tout ce que l'ordinateur est capable de faire. Il existe énormément de langages de programmation, la plupart étant listés sur cette page.

Programmer des scripts

Le Javascript permet de programmer des **scripts**. Comme dit plus haut, un langage de programmation permet d'écrire du code source qui sera analysé par l'ordinateur.

Il existe trois manières d'utiliser du code source :

- **Langage compilé** : le code source est donné à un programme appelé **compilateur** qui va lire le code source et le convertir dans un langage que l'ordinateur sera capable d'interpréter : c'est le langage binaire, fait de 0 et de 1. Les langages comme le C ou le C++ sont des langages dits compilés.
- **Langage précompilé** : ici, le code source est compilé partiellement, généralement dans un code plus simple à lire pour l'ordinateur, mais qui n'est pas encore du binaire. Ce code intermédiaire devra être lu par ce que l'on appelle une « machine virtuelle », qui exécutera ce code. Les langages comme le C# ou le Java sont dits précompilés.
- **Langage interprété** : dans ce cas, il n'y a pas de compilation. Le code source reste tel quel, et si on veut exécuter ce code, on doit le fournir à un interpréteur qui se chargera de le lire et de réaliser les actions demandées.

Les scripts sont majoritairement interprétés. Et quand on dit que le Javascript est un langage de scripts, cela signifie qu'il s'agit d'un langage interprété ! Il est donc nécessaire de posséder un interpréteur pour faire fonctionner du code Javascript, et un interpréteur, vous en utilisez un fréquemment : il est inclus dans votre navigateur Web !

Chaque navigateur possède un interpréteur Javascript, qui diffère selon le navigateur. Si vous utilisez Internet Explorer, son interpréteur Javascript s'appelle *JScript* (l'interpréteur de la version 9 s'appelle *Chakra*), celui de Mozilla Firefox se nomme *SpiderMonkey* et celui de Google Chrome est *V8*.

Le Javascript, le langage de scripts

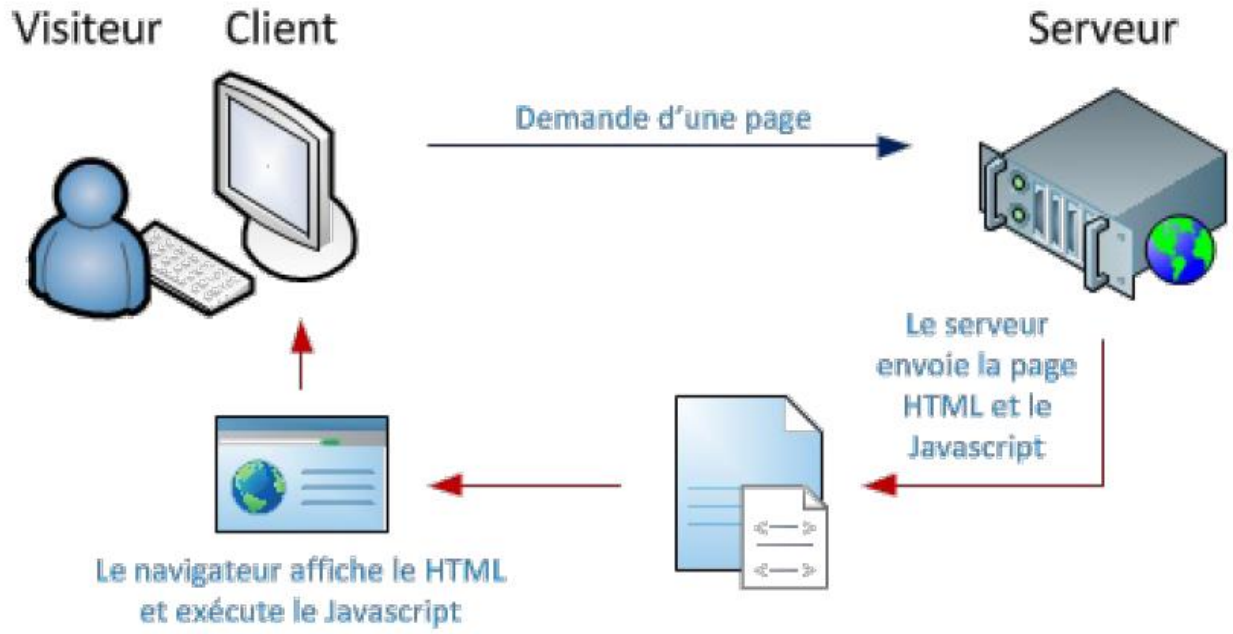
Le Javascript est à ce jour utilisé majoritairement sur Internet, conjointement avec les pages Web (HTML ou XHTML). Le Javascript s'inclut directement dans la page Web (ou dans un fichier externe) et permet de *dynamiser* une page HTML, en ajoutant des interactions avec l'utilisateur, des animations, de l'aide à la navigation, comme par exemple :

- Afficher/masquer du texte ;
- Faire défiler des images ;
- Créer un diaporama avec un aperçu « en grand » des images ;
- Créer des infobulles.

Le Javascript est un langage dit ***client-side***, c'est-à-dire que les scripts sont exécutés par le navigateur chez l'internaute (le **client**). Cela diffère des langages

de scripts dits ***server-side*** qui sont exécutés par le serveur Web. C'est le cas des langages comme le PHP.

C'est important, car la finalité des scripts *client-side* et *server-side* n'est pas la même. Un script *server-side* va s'occuper de « créer » la page Web qui sera envoyée au navigateur. Ce dernier va alors afficher la page puis exécuter les scripts *client-side* tel que le Javascript. Voici un schéma reprenant ce fonctionnement :



Le Javascript, pas que le Web

Si le Javascript a été conçu pour être utilisé conjointement avec le HTML, le langage a depuis évolué vers d'autres destinées. Le Javascript est régulièrement utilisé pour réaliser des extensions pour différents programmes, un peu comme les scripts codés en Lua ou en Python.
Le Javascript peut aussi être utilisé pour réaliser des applications. Mozilla Firefox est l'exemple le plus connu : l'interface du navigateur est créée avec une sorte de HTML appelé XUL et c'est le Javascript qui est utilisé pour animer l'interface. D'autres logiciels reposent également sur cette technologie, comme TomTom HOME qui sert à gérer votre GPS TomTom via votre PC.

Petit historique du langage

En 1995, Brendan Eich travaille chez Netscape Communication Corporation, la société qui éditait le célèbre navigateur Netscape Navigator, alors principal concurrent d'Internet Explorer.

Brendan développe le LiveScript, un langage de script qui s'inspire du langage Java, et qui est destiné à être installé sur les serveurs développés par Netscape. Netscape se met à développer une version client du LiveScript, qui sera renommée JavaScript en hommage au langage Java créé par la société Sun Microsystems. En effet, à cette époque, le langage Java était de plus en plus populaire, et appeler le LiveScript JavaScript était une manière de faire de la publicité, et au Java, et au JavaScript lui-même. Mais attention, au final, **ces deux langages sont radicalement différents** ! N'allez pas confondre le Java et le Javascript car ces deux langages n'ont clairement pas le même fonctionnement.

Le Javascript sort en décembre 1995 et est embarqué dans le navigateur Netscape 2. Le langage est alors un succès, si bien que Microsoft développe une version semblable, appelée JScript, qu'il embarque dans Internet Explorer 3, en 1996.

Netscape décide d'envoyer sa version de Javascript à l'ECMA International (*European Computer Manufacturers Association* à l'époque, aujourd'hui *European association for standardizing information and communication systems*) pour que le langage soit standardisé, c'est-à-dire pour qu'une référence du langage soit créée et que le langage puisse ainsi être utilisé par d'autres personnes et embarqué dans d'autres logiciels. L'ECMA International standardise le langage sous le nom d'*ECMAScript*.

Depuis, les versions de l'ECMAScript ont évolué. La version la plus connue et mondialement utilisée est la version ECMAScript 3, parue en décembre 1999.

L'ECMAScript et ses dérivés

L'ECMAScript est la référence de base. De cette référence découlent des **implémentations**. On peut évidemment citer le Javascript, qui est implémenté dans la plupart des navigateurs, mais aussi :

- **JScript**, qui est l'implémentation embarquée dans Internet Explorer. C'est aussi le nom de l'interpréteur d'Internet Explorer ;
- **JScript.NET**, qui est embarqué dans le framework .NET de Microsoft ;
- **ActionScript**, qui est l'implémentation faite par Adobe au sein de Flash ;
- **EX4**, qui est l'implémentation de la gestion du XML d'ECMAScript au sein de SpiderMonkey, l'interpréteur Javascript de Firefox.

Les versions du Javascript

Les versions du Javascript sont basées sur celles de l'ECMAScript (que nous abrégerons ES). Ainsi, il existe :

- ES 1 et ES 2, qui sont les prémices du langage Javascript ;

- ES 3 (sorti en décembre 1999), qui est fonctionnel sur tous les navigateurs (sauf les vieilles versions d'Internet Explorer) ;
- ES 4, qui a été abandonné en raison de modifications trop importantes qui ne furent pas appréciées ;
- ES 5 (sorti en décembre 2009), qui est la version la plus récemment sortie ;
- ES 6, qui est actuellement en cours de conception.

En résumé :

- Le Javascript est un langage de programmation interprété, c'est-à-dire qu'il a besoin d'un interpréteur pour pouvoir être exécuté.
- Le Javascript est utilisé majoritairement au sein des pages Web.
- Tout comme le HTML, le Javascript est exécuté par le navigateur de l'internaute : on parle d'un comportement *client-side*, par opposition au *server-side* lorsque le code est exécuté par le serveur.
- Le Javascript est standardisé par l'ECMA International sous le nom d'ECMAScript qui constitue la référence du langage. D'autres langages découlent de l'ECMAScript, comme ActionScript, EX4 ou encore JScript.NET.
- La dernière version standardisée du Javascript est basée sur l'ECMAScript 5, sorti en 2009.

Premiers pas en Javascript

Comme indiqué précédemment, le Javascript est un langage essentiellement utilisé avec le HTML, vous allez donc apprendre dans ce chapitre comment intégrer ce langage à vos pages Web, découvrir sa syntaxe de base et afficher un message sur l'écran de l'utilisateur.

Concernant l'éditeur de texte à utiliser (dans lequel vous allez écrire vos codes Javascript), celui que vous avez l'habitude d'utiliser avec le HTML supporte très probablement le Javascript aussi.

Afficher une boîte de dialogue
Le Hello World!

Voici un code HTML simple contenant une instruction Javascript, placée au sein d'un élément **<script>** :

```
<html>
   <head>
      <title>Hello World!</title>
   </head>
```

```
  <body>
    <script>
        alert('Hello world!');
    </script>
  </body>
</html>
```

Écrivez ce code dans un fichier HTML, et ouvrez ce dernier avec votre navigateur habituel. Une boîte de dialogue s'ouvre, vous présentant le texte « Hello World! ».

Explication :

Tout d'abord, un élément **<script>** est présent : c'est lui qui contient le code Javascript que voici :

```
alert('Hello world!');
```

Il s'agit d'une instruction, c'est-à-dire une commande, un ordre, ou plutôt une action que l'ordinateur va devoir réaliser. Les langages de programmation sont constitués d'une suite d'instructions qui, mises bout à bout, permettent d'obtenir un programme ou un script complet.

Dans cet exemple, il n'y a qu'une instruction : l'appel de la fonction alert().

La boîte de dialogue alert()

alert() est une instruction simple, appelée **fonction**, qui permet d'afficher une boîte de dialogue contenant un message. Ce message est placé entre apostrophes, elles-mêmes placées entre les parenthèses de la fonction alert().

La syntaxe du Javascript

La syntaxe du Javascript n'est pas compliquée. De manière générale, les instructions doivent être séparées par un point-virgule que l'on place à la fin de chaque instruction :

```
instruction_1;
instruction_2;
instruction_3;
```

En réalité le point-virgule n'est pas obligatoire si l'instruction qui suit se trouve sur la ligne suivante, comme dans notre exemple.

En revanche, si vous écrivez plusieurs instructions sur une même ligne, comme dans l'exemple suivant, le point-virgule est obligatoire. Si le point-virgule n'est pas mis, l'interpréteur ne va pas comprendre qu'il s'agit d'une autre instruction et risque de retourner une erreur.

```
Instruction_1;Instruction_2
Instruction_3
```

NB : Mais attention ! Ne pas mettre les points-virgules est considéré comme une *mauvaise pratique*, c'est quelque chose que les développeurs Javascript évitent de faire, même si le langage le permet.

La compression des scripts

Certains scripts sont disponibles sous une forme dite compressée, c'est-à-dire que tout le code est écrit à la suite, sans retours à la ligne. Cela permet d'alléger considérablement le poids d'un script et ainsi de faire en sorte que la page soit chargée plus rapidement. Des programmes existent pour « compresser » un code Javascript. Mais si vous avez oublié un seul point-virgule, votre code compressé ne fonctionnera plus, puisque les instructions ne seront pas correctement séparées. C'est aussi une des raisons qui fait qu'il faut *toujours* mettre les points-virgules en fin d'instruction.

Les espaces

Le Javascript n'est pas sensible aux espaces. Cela veut dire que vous pouvez aligner des instructions comme vous le voulez, sans que cela ne gêne en rien l'exécution du script. Par exemple, ceci est correct :

```
instruction_1;
    instruction_1_1;
    instruction_1_2;
instruction_2;      instruction_3;
```

Indentation et présentation

L'indentation, en informatique, est une façon de structurer du code pour le rendre plus lisible. Les instructions sont hiérarchisées en plusieurs niveaux et on utilise des espaces ou des tabulations pour les décaler vers la droite et ainsi créer une hiérarchie. Voici un exemple de code *indenté* :

```
function toggle(elemID) {
      var elem = document.getElementById(elemID);
      if (elem.style.display == 'block') {
         elem.style.display = 'none';
      } else {
         elem.style.display = 'block';
      }
}
```

Ce code est indenté de quatre espaces, c'est-à-dire que le décalage est chaque fois un multiple de quatre. Un décalage de quatre espaces est courant, tout comme un décalage de deux. Il est possible d'utiliser des tabulations pour indenter du code. Les tabulations présentent l'avantage d'être affichées différemment suivant l'éditeur utilisé, et de cette façon, si vous donnez votre code à quelqu'un, l'indentation qu'il verra dépendra de son éditeur et il ne sera pas perturbé par une indentation qu'il n'apprécie pas (par exemple, nous n'aimons pas les indentations de deux, nous préférons celles de quatre).
Voici le même code, mais non indenté, pour vous montrer que l'indentation est une aide à la lecture :

```
function toggle(elemID) {
var elem = document.getElementById(elemID);
if (elem.style.display == 'block') {
elem.style.display = 'none';
} else {
elem.style.display = 'block';
}
}
```

Les commentaires

Les commentaires sont des annotations faites par le développeur pour expliquer le fonctionnement d'un script, d'une instruction ou même d'un groupe d'instructions. Les commentaires ne gênent pas l'exécution d'un script.
Il existe deux types de commentaires : les commentaires de fin de ligne, et les commentaires multilignes.

Commentaires de fin de ligne

Ils servent à commenter une instruction. Un tel commentaire commence par deux slashs :

```
instruction_1; // Ceci est ma première instruction
instruction_2;
// La troisième instruction ci-dessous :
instruction_3;
```

Le texte placé dans un commentaire est ignoré lors de l'exécution du script, ce qui veut dire que vous pouvez mettre ce que bon vous semble en commentaire, même une instruction (qui ne sera évidemment pas exécutée) :

```
instruction_1; // Ceci est ma première instruction
instruction_2;
// La troisième instruction ci-dessous pose problème, je l'annule temporairement
// instruction_3;
```

Commentaires multilignes

Ce type de commentaires permet les retours à la ligne. Un commentaire multiligne commence par /* et se termine par */ :

```
/* Ce script comporte 3 instructions :
- Instruction 1 qui fait telle chose
- Instruction 2 qui fait autre chose
- Instruction 3 qui termine le script
*/
instruction_1;
instruction_2;
instruction_3; // Fin du script
```

Remarquez qu'un commentaire multiligne peut aussi être affiché sur une seule ligne :

```
instruction_1; /* Ceci est ma première instruction */
instruction_2;
```

Les fonctions

Dans l'exemple du Hello world!, nous avons utilisé la fonction alert(). Nous reviendrons en détail sur le fonctionnement des fonctions, mais pour les chapitres suivants, il sera nécessaire de connaître sommairement leur syntaxe.
Une fonction se compose de deux choses : son nom, suivi d'un couple de parenthèses (une ouvrante et une fermante) :

```
myFunction(); // « function » veut dire « fonction » en anglais
```

Entre les parenthèses se trouvent les **arguments**, que l'on appelle aussi **paramètres**. Ceux-ci contiennent des valeurs qui sont transmises à la fonction. Dans le cas du Hello world!, ce sont les mots « Hello world! » qui sont passés en paramètre :

```
alert('Hello world!');
```

Où placer le code dans la page

Les codes Javascript sont insérés au moyen de l'élément **<script>**. Cet élément possède un attribut type qui sert à indiquer le type de langage que l'on va utiliser. Dans notre cas, il s'agit de Javascript, mais ça pourrait être autre chose, comme du VBScript, bien que ce soit extrêmement rare.
NB : Si vous n'utilisez pas le HTML5, sachez que l'attribut type prend comme valeur text/javascript, qui est en fait le type MIME d'un code Javascript.

Le type MIME est un identifiant qui décrit un format de données. Ici, avec text/javascript, il s'agit de données textuelles et c'est du Javascript.

Le Javascript « dans la page »

Pour placer du code Javascript directement dans votre page Web, rien de plus simple, on fait comme dans l'exemple du Hello world! : on place le code au sein de l'élément **<script>** :

```
<html>
   <head>
      <title>Hello World!</title>
   </head>
   <body>
```

```
    <script>
        alert('Hello world!');
    </script>
  </body>
</html>
```

L'encadrement des caractères réservés

Si vous utilisez les normes HTML 4.01 et XHTML 1.x, il est souvent nécessaire d'utiliser des **commentaires d'encadrement** pour que votre page soit conforme à ces normes. Si par contre, comme dans ce cours, vous utilisez la norme HTML5, les commentaires d'encadrement sont inutiles.

Les commentaires d'encadrement servent à isoler le code Javascript pour que le validateur du W3C (*World Wide Web Consortium*) ne l'interprète pas. Si par exemple votre code Javascript contient des chevrons < et >, le validateur va croire qu'il s'agit de balises HTML mal fermées, et donc va invalider la page. Ce n'est pas grave en soi, mais une page sans erreurs, c'est toujours mieux !

Les commentaires d'encadrement ressemblent à des commentaires HTML et se placent comme ceci :

```
<body>
    <script>
<!--
    valeur_1 > valeur_2;
//-->
  </script>
</body>
```

Le Javascript externe

Il est possible, et même conseillé, d'écrire le code Javascript dans un fichier externe, portant l'extension .js. Ce fichier est ensuite appelé depuis la page Web au moyen de l'élément **<script>** et de son attribut src qui contient l'URL du fichier .js.

Voici tout de suite un petit exemple :

Hello.js :

```
alert('Hello world!');
```

Test.html :

```
<html>
   <head>
      <title>Hello World!</title>
   </head>
   <body>
     <script src="Hello.js"> </script>
   </body>
</html>
```

On suppose ici que le fichier *hello.js* se trouve dans le même répertoire que la page Web.

NB : Il vaut mieux privilégier un fichier externe plutôt que d'inclure le code Javascript directement dans la page, pour la simple et bonne raison que le fichier externe est mis en cache par le navigateur, et n'est donc pas rechargé à chaque chargement de page, ce qui accélère l'affichage de la page.

POSITIONNER L'ÉLÉMENT <SCRIPT>

Une page Web est lue par le navigateur de façon linéaire, c'est-à-dire qu'il lit d'abord le **<head>**, puis les éléments de **<body>** les uns à la suite des autres. Si vous appelez un fichier Javascript dès le début du chargement de la page, le navigateur va donc charger ce fichier, et si ce dernier est volumineux, le chargement de la page s'en trouvera ralenti. C'est normal puisque le navigateur va charger le fichier avant de commencer à afficher le contenu de la page.

Pour pallier ce problème, il est conseillé de placer les éléments **<script>** juste avant la fermeture de l'élément **<body>**, comme ceci :

```
<html>
  <head>
     <title>Hello World!</title>
  </head>
  <body>
    <p>
       <!--
 Contenu de la page Web
 ...
 -->
```

```
    </p>
    <script>
        // Un peu de code Javascript...
    </script>
    <script src="hello.js"></script>
  </body>
</html>
```

Il est à noter que certains navigateurs modernes chargent automatiquement les fichiers Javascript en dernier, mais ce n'est pas toujours le cas. C'est pour cela qu'il vaut mieux s'en tenir à cette méthode.

En résumé :

- Les instructions doivent être séparées par un point-virgule.
- Un code Javascript bien présenté est plus lisible et plus facilement modifiable.
- Il est possible d'inclure des commentaires au moyen des caractères //, /* et */.
- Les codes Javascript se placent dans un élément **<script>**.
- Il est possible d'inclure un fichier Javascript grâce à l'attribut src de l'élément **<script>**.