Bases de Javascript

Présentation :

Le JavaScript est un langage de programmation.
Pour être plus précis, c'est un **langage orienté objet** : quand on code en JavaScript, on se base sur des *objets*.

Le JavaScript est un langage de script, Il s'insère dans le code (x)HTML d'une page web, et permet d'en augmenter le spectre des possibilités.

Ce langage de POO *[Programmation Orientée Objet]*, faiblement typé, est exécuté côté client.

De quelle manière s'utilise-t-il ?

A l'intérieur des pages web : on dit alors que le JavaScript est **une extension du (x)HTML**. Il permet de rendre celles-ci *interactives*.

Voici quelques exemples :

- ouvrir des *pop-up* (les petites fenêtres qui s'ouvrent de manière intempestive)
- faire défiler un texte
- insérer un menu dynamique (qui se développe au passage de la souris)
- proposer un diaporama (changement d'image toute les X secondes, boutons pour mettre en pause, aller à l'image précédente / suivante, etc.)
- avoir une horloge "à aiguilles"
- faire en sorte que des images suivent le pointeur de la souris
- créer de petits jeux (comme le classique "*Plus ou Moins*", cf. TP)
- insérer des balises du zCode (les balises qui apparaissent en cliquant sur le bouton)
- faire un aperçu du zCode en direct.

(Il suffit d'un navigateur et d'un éditeur de texte tel que Bloc-notes).

Historique du langage :

Le JavaScript, inventé par un certain *Brendan Eich* et développé par *Netscape*, fait son apparition en *1995*, sous le nom de *LiveScript*, dans le but de dynamiser les pages web.
Son utilisation s'est largement répandue, et il se fait rapidement accepter par d'autres navigateurs.

Aujourd'hui très présent sur les sites web, mieux en mieux accepté, à la fois par les navigateurs et par les visiteurs.

Les caractéristiques du JavaScript

- C'est un **script**
- C'est un langage **orienté objet**
- Le code n'est **pas compilé** :
- il est donc **plus rapide à produire** (pas besoin de compilateur, un seul fichier, ...),
- mais il est **moins puissant** qu'un programme en C, par exemple,
- et relativement **limité** : il se limite plus ou moins à la page web sur laquelle il se trouve. Il ne permet donc pas de faire des choses comme manipuler des fichiers sur votre disque dur seulement des choses assez simples.
- Il est **exécuté par le navigateur du visiteur** (le *client*), et dépend donc de celui-ci.

- Il est **déterminé par une norme**, nommée *ECMA-262* ou ***ECMAScript***. De la même manière que le W3C se charge de définir clairement le (x)HTML.

Fonctionnement du JavaScript :

Un script, c'est tout simplement *un bout de code JavaScript qui a une tâche précise*. Dès que l'on parlera de script, ce sera pour désigner le code que l'on aura inséré à notre page Web.

Par opposition à un langage compilé, un langage qui ***s'interprète***. Ici, l'interprète du JavaScript, c'est *le navigateur du visiteur* (le client).

Les scripts ne sont pas obligatoirement exécutés au chargement de la page. Ils sont lancés lorsqu'un événement spécifique se produit.

Exemples d'événements qui peuvent se dérouler lors de la visite d'une page Web :

- lorsque vous chargez la page (exemple : ouvrir un pop-up)
- lorsque vous changez de page (un autre pop-up)
- lorsque vous cliquez sur un lien (vous aimez les pop-up ?)
- lorsque vous sélectionnez un élément d'un menu déroulant (vous serez par exemple redirigé vers une autre page)
- lorsque vous validez un formulaire (avant qu'il soit envoyé : vous pouvez alors vérifier si les champs sont correctement remplis).

Exemple :

En PHP, vous devriez savoir que celui-ci est exécuté côté serveur. Autrement dit, vous demandez au serveur de vous donner une page. Il va lire cette page, exécuter le code PHP et vous donner la page ainsi générée.

Eh bien en JavaScript, ce n'est **pas** comme ça.

Vous demandez la page au serveur : il vous la donne (après avoir exécuté les éventuels scripts PHP), et c'est **votre navigateur** qui exécute le script (soit immédiatement, soit lorsqu'un événement précis se produit).

- votre ordinateur récupère le code source de la page en question.
- Votre navigateur interprète la page et les scripts qu'elle contient.
- La page formatée s'affiche sur votre écran. Les scripts, quant à eux, sont mis en mémoire et seront lancés dès que l'événement attendu se produira.

Création de notre premier Script :

Logiciels nécessaires :

De même que pour créer une page (x)HTML, il vous suffit :

- d'un éditeur de texte : Bloc-notes est suffisant, mais d'autres logiciels offrent plus d'options, comme la coloration syntaxique, qui est très appréciable.
  Notepad++ (que vous possédez peut-être déjà...) en est un bon exemple.

- d'un navigateur : **FireFox**, avec le plugin **Web Developer**, est parfaitement approprié pour cela. Cependant, pour vérifier la compatibilité de vos scripts, il peut être utile de les tester sur plusieurs des navigateurs les plus courants (*Internet Explorer 6*, *Opera* et *Firefox* étant les références pour les PC).

Voici quelques liens :

http://notepad-plus-plus.org/download/v6.6.8.html

https://www.mozilla.org/fr/firefox/new/

http://chrispederick.com/work/web-developer/

Où se place le code JavaScript ?

On écrit le JS dans notre page HTML, et c'est le navigateur qui lit et exécute ce code.
Mais comment fait-il pour différencier les deux ?

Si on ne précise pas à notre navigateur que notre code est du JS, il ne le devinera pas tout seul.

Structure de la page xHTML (code minimal) :

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="fr" lang="fr">
	<head>
		<title>Titre de la page</title>
		<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
		 <!-- en-tete du document -->
	</head>
	<body>
		<!-- corps du document -->
	</body>
</html>
```

Il y a deux méthodes différentes pour insérer du JavaScript dans une page Web.

Directement dans les balises

La première méthode consiste à écrire le code directement **à l'intérieur de la balise** qui va être concernée par le script.

Pour insérer le code dans une balise, une nouvelle *propriété* est nécessaire. Il s'agit du **gestionnaire d'événements** (en anglais : *event handler*).

Exercice 01 :

Rajoutez ce script à la page xHTML entre les balises <head> </head>

```
< a href= "#" onclick = "alert('Bonjour !');" > lien </a>
```

Exercice 02:

Essayez de faire la même chose, mais avec une image cette fois.

****

Exemple 03 :

Il est possible d'écrire du JavaScript directement à la place de l'adresse d'un lien, en le faisant précéder de javascript:

 Cliquez ici

Autres gestionnaires d'événements

Il existe d'autres événements que le clic de souris

En voici quelques-uns :

- onclick : au clic (gauche) de la souris
- ondblclick : lors d'un double clic
- onmouseover : au passage de la souris
- onmouseout : lorsque la souris "sort" de cet élément (en quelque sorte, le contraire de onmouseover).

Gestionnaires d'événements de la balise <body> :

- onload : au chargement de la page
- onunload : lorsqu'on quitte la page (ou qu'on change de page).

Ecrire du JavaScript Entre les balises <script> et </script>

On peut les placer soit dans l'en-tête <head> ... </head>, soit dans le corps <body> ... </body> de la page (x)HTML.
Il faut commencer par préciser au navigateur que notre script est du JavaScript.
Pour cela, on rajoute la propriété ***type="text/javascript"***, ce qui nous donne ceci :

```
<script type="text/javascript">
        /* votre code javascript se trouve ici
         C'est deja plus pratique pour un script de plusieurs lignes */
</script>
```

Pour les vieux navigateurs on l'écrit comme ceci :

```
<script type="text/javascript">
<!--
        /* et vous pouvez placer votre code JS ici sans être inquiété par les vieux navigateurs */
//-->
</script>
```

Exemple d'application

Essayez de réaliser cet exemple : dans notre page web, on veut :

- une boîte de dialogue indiquant le début du chargement de la page (donc, le code est à placer au début du corps de la page),
- une autre indiquant la fin du chargement de celle-ci (donc, à la fin du corps).

Voici le code complet de la page :

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="fr" lang="fr">

<head>

<!-- en-tete du document -->

  <title>Un exemple</title>

  <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
</head>

<body>

  <!-- script pour le debut du chargement -->
  <script type="text/javascript">
  <!--
    Alert ('Debut du chargement de la page');
  //-->
  </script>

  <!-- ici se trouve le contenu de la page web -->
  <p>
    Vous testez un script...<br />
    Enjoy ;)
  </p>

  <!-- script pour la fin du chargement -->
  <script type="text/javascript">
  <!--
    Alert ('Fin du chargement de la page');
  //-->
  </script>

</body>

</html>
```

Le navigateur exécute le JS au fur et à mesure du chargement de la page : il attend donc que le script soit terminé avant de charger la suite.

Placer le code dans un fichier séparé

Importer un fichier externe

Comme pour le CSS, on peut très bien placer notre code dans un fichier indépendant. On dit que **le code est importé depuis un fichier externe**.

L'extension du fichier externe contenant du code JavaScript est **.js**.
On va indiquer aux balises le nom et l'emplacement du fichier contenant notre (ou nos) script(s), grâce à la propriété src (comme pour les images).

Exemple :

Créer le fichier et son script :

Si vous ne savez pas créer un fichier ayant l'extension **.js**, voici comment faire.

Avec *Notepad*++

- Créez un nouveau fichier avec le menu **Fichier**, puis **Nouveau**.
- Dans le menu **Langage**, sélectionnez... **Javascript** ! 
- Enregistrez votre fichier, en lui donnant le nom que vous voulez, suivi de **.js**.
- Pour ouvrir à nouveau ce fichier, clic droit, puis ***Open in Notepad***++.

Voici le script à écrire dans votre page xhtml

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

Voici le contenu de votre fichier .js à créer :

```
alert('Azul fellawen');
```

De quoi se compose le code ?

Une instruction est un "ordre" que l'on donne à l'ordinateur, comme on pourrait taper "format C:" dans sa console.
Quand on donne une instruction à l'ordinateur, il faut également lui dire où est la fin de cette instruction.
Utiliser un **point-virgule (;)** à la fin de chaque instruction.

Les fonctions :

Une fonction est une suite d'instructions ayant un rôle précis.

alert(‘Bonjour’);

Cette fonction affiche, lorsqu'on l'appelle (*terme à retenir*), une boîte de dialogue contenant le texte entre les apostrophes (ici : ‘Bonjour’).

Rôle des parenthèses

Exemple mathématique : entre les parenthèses, on précise la valeur de l'argument x.

f(x) = 15x - 4
Avec x = 5, on aura f(5) = 15*5 - 4 = 71.

C'est le même principe en JavaScript : par exemple, avec alert(), l'argument est le texte à afficher. S'il y a plusieurs arguments, ils sont séparés par des virgules.

Le navigateur va alors chercher la définition de cette fonction et l'exécuter, ce qui aura pour effet de faire apparaître notre message.

Mais il est également possible de créer nos propres fonctions (par exemple, une fonction qui convertit des euros en francs), ce sera l'objet d'un chapitre.

Applications 01 :

```
<HTML>
<HEAD>
<TITLE> Mon titre de page </TITLE>
<SCRIPT type="text/javascript" >
function Init()
{
Alert ("Message dans une fenêtre");
Boite de message au chargement d'une page HTML
Alert ("Message dans une fenêtre");
}
</SCRIPT>
</HEAD>
<BODY onload="Init()" >
Texte sur la page <BR>
<SCRIPT type="text/javascript">
document.writeln("un texte imprimer avec
javascript
</SCRIPT>
Suite du texte.
</BODY>
</HTML>
```

Application 2 :

```
<HTML>
<HEAD>
<TITLE> Boite de dialogue confirm</TITLE>
<SCRIPT type="text/javascript" >

function Init()
{
var chaine;
chaine = prompt("saisir un texte \nMerci ...","texte par défaut.....");
if ( chaine == null )
{
alert("Pas de saisie !!!!");
}
else
{
chaine = "Voici le texte >> [" + chaine + "]";
alert(chaine);
}
}
</SCRIPT>
</HEAD>
<BODY onload="Init()">
Texte sur le page
</BODY>
</HTML>
```

Application 3 :

```
<HTML>
<HEAD>
<TITLE> Méthode getElementById </TITLE>
<SCRIPT type="text/javascript" >
```

```
function ChangeNom()
{
var chaine;
var AncienNom = document.getElementById("rien").value;
chaine = prompt("saisir le nom du bouton", AncienNom);
document.getElementById("rien").value = chaine;
}

</SCRIPT>
</HEAD>
<BODY>
        <CENTER>Bouton modifiable <BR><BR><BR>
<INPUT type="button"id="rien"value="mmmmm"onclick="ChangeNom()"></INPUT></CENTER>
</BODY>
</HTML>
```