Les bases du JavaScript (JS)

Introduction :

Le *JavaScript* (officiellement basé sur *ECMAscript*) est un ***langage*** de programmation *interprété orienté objet à prototype*. Il est le frère *d'ActionScript* du logiciel *Flash*, souvent utilisé pour animer les pages web, afficher de la vidéo, etc. *JavaScript* (pour être exact sa version en *C* appelée *SpiderMonkey*) est également au cœur du moteur *Gecko*, qui fait tourné la plupart des logiciels *Mozilla* (comme *Firefox* ou *Thunderbird*).

JavaScript permet de rédiger des ***scripts*** principalement dédiés à une *exécution* (sans *compilation* préalable) *côté client*, plutôt que *côté serveur*. Comme *l'exécution* des ***scripts*** se fait par les *navigateurs*, il existe donc de petites variantes de ***méthode***, ce qui nécessite parfois de prendre quelques précautions dans la rédaction d'un ***script*** pour s'assurer d'une bonne exécution sur tous les *navigateurs*. Bref, rien de nouveau par rapport à l'utilisation *d'HTML5* ou de la *CSS*.

Il s'agit d'un langage *asynchrone*, c'est-à-dire que plusieurs ***instructions*** peuvent être exécutées de manière parallèle, sans qu'il ne faille attendre la fin des précédentes pour continuer.

Principes de base :

Comme le *HTML5*, le *JavaScript* est un ***langage*** relativement permissif, cela peut induire parfois des effets étranges lorsqu'on ne respecte pas totalement une syntaxe stricte. Par exemple chaque ***instruction*** est sensée se terminer par un **;**, mais il est aussi possible (mais non recommandé) de l'omettre, à condition de *passer à la ligne* suivante. Par contre, le *JS* est sensible à la *casse*.

Le *JS* permet de manipuler le *HTML* (le *SVG* et la *CSS*) de manière dynamique. En fait, chaque ***balise*** du *HTML* est considérée comme un ***objet*** manipulable par *JavaScript* par l'intermédiaire du ***DOM*** (*Document Object Model*). Ce dernier crée une ***arborescence*** de tous les ***éléments***, dont la *racine* est le mot-clé **document**, lui-même inclus dans ***l'objet* window** (qui correspond à l'onglet du *navigateur*, mais il peut être omis la plupart du temps). La ***notation pointée*** permet d'atteindre les différents ***paramètres*** de chaque ***objet***, selon ***l'arborescence*** du ***DOM***.

Exemple : modification dynamique d'un texte au survol de la souris

```
<p id="modif" onmouseover="changement(true);" onmouseout="changement(false);"> Texte initialement affiché.</p>

<!-- les attributs d'événement lancent la fonction javascript qui est définie ci-dessous -->

<script type="text/javascript">

        var monObjet = document.getElementById('modif'); //on associe l'élément à une variable

        function changement(vrai) {   // on définit la fonction appelée précédemment

                if (vrai) {monObjet.innerHTML="Texte au survol de la souris" ;}

                else {monObjet.innerHTML="Texte lorsque la souris est partie" ;}

                return false;  // cette instruction facultative met fin à l'exécution de la fonction.

        } // il n'y a pas de ; à la fin d'une définition de fonction, car elle n'est pas exécutée ici.

</script>
```

Où intégrer son script ?

Il est possible d'ajouter directement dans l'*entête* (**<head>**) et/ou le *corps* (**<body>**) du fichier *HTML*, le *javascript* souhaité, à l'intérieur des balises **<script type="text/javascript">…</script>**.

Il est également possible d'appeler un *script* situé dans un fichier **.js** en utilisant *l'attribut* **src** inclus dans la *balise* **script** : **<script type="text/javascript" src="fichier.js"></script>**

REMARQUE : Lorsque le *code* fait référence à *l'identité* de *balises*, il est obligatoire de déclarer ce *code JS* après lesdites *balises*, sans quoi le *DOM* n'étant pas encore correctement généré, des erreurs se produiront.

Comment lancer le script ?

JavaScript est principalement utilisé pour ajouter de l'interactivité au fichier *HTML*. Le lancement conditionnel du *script* (ou d'une portion de celui-ci) est ce qu'on recherche le plus souvent. On utilise pour cela le *gestionnaire d'événements DOM* et des *fonctions JS* (comme dans l'exemple précédent). Cependant, il est aussi possible d'exécuter automatiquement du *code* au chargement du fichier. En fait, tout est une question de syntaxe.

Les *instructions* sont immédiatement exécutées, à moins d'être situées dans une *fonction*. Il est possible d'appeler une *fonction* à la suite d'un *événement* à l'aide du *gestionnaire d'événement*. Celui-ci est déclaré soit dans la *balise HTML* à l'aide de *l'attribut* correspondant (**<***balise événement***="***fonction***(***argument***);">**), soit dans le *script*, à l'aide de la *notation pointée* (*objetSurveillé*.*événement* **=function(){***instruction* **;};**). Par convention, pour exécuter immédiatement des *instructions* au chargement de la *page*, *l'objet surveillé* et *l'événement* sont alors **window.onload**.

REMARQUE : il ne peut y avoir qu'une seule *déclaration* pour chaque *événement*. Ainsi si on souhaite lancer 2 *instructions* au chargement de la *page*, il faut les grouper dans la même *déclaration*. Dans le cas suivant, seule la seconde *instruction* est exécutée : **window.onload = function(){** *instruction1***;};** **window.onload = function(){** *instruction2***;};**

Les principaux événements :

onblur : *L'élément* (possédant un **tabindex**) perd le focus.

onchange : *L'élément* de *formulaire* est changé.

onclick : Un clic gauche a été réalisé sur *l'élément*.

oncontextmenu : Un menu contextuel (résultant généralement d'un clic droit) est affiché.

ondblclick : Un double clic est réalisé sur *l'élément*.

onfocus : *L'élément* (possédant un **tabindex**) prend le focus.

oninput : Une saisie a été réalisée dans *l'élément* du *formulaire*.

onkeydown : Une touche du clavier a été pressée.

onkeypress : Une touche du clavier a été pressée puis relâchée. (Ne gère pas toutes les touches).

onkeyup : Une touche du clavier est relâchée.

onload : ***L'élément*** est en cours de chargement.

onmousedown : Un bouton de la souris est pressé

onmousemove : La souris se déplace.

onmouseout : La souris sort de ***l'élément***.

onmouseover : La souris survole ***l'élément***.

onmouseup : Le bouton de la souris est relâché.

onresize : La fenêtre est redimensionnée.

onselect : ***L'élément*** est sélectionné.

onunload : Le document est quitté.

Indenter et commenter :

Comme pour les autres ***langages***, il est fortement conseiller ***d'indenter*** et de ***commenter*** son ***script***. Comme il y a souvent plusieurs façons d'obtenir le même résultat, les ***commentaires*** sont donc essentiels.

Il y a deux façons d'introduire un ***commentaire*** dans un ***script*** : soit on ajoute **//** suivi du commentaire sans changement de ligne (!) ; soit on utilise la même syntaxe qu'en *CSS* (**/*** *commentaire* *****/**).

```
instruction ; // commentaire de fin de ligne pouvant être situé après une instruction.
/* commentaire pouvant
    s'étaler sur plusieurs lignes */
```

Les variables et les fonctions

Variable locale ou variable globale :

Les ***variables*** peuvent être *globale* (utilisables partout dans le ***script***) ou *locale* à une ***fonction***. Il est recommandé de ne pas abuser des ***variables*** *globales*, car elles restent stockées en mémoire, alors qu'une ***variable*** *locale* est « détruite » après son utilisation. Dans l'exemple précédent, la ***variable*** ayant été déclarée en dehors de la ***fonction***, elle est *globale*. Si on avait placé la déclaration dans la ***fonction***, elle aurait été *locale*.

Déclarer une ***variable*** commence par le *mot-clé* **var** et se termine par **;**. On peut déclarer plusieurs ***variables*** sur la même ligne, en les séparant par une *virgule*. Une ***valeur*** peut leur être immédiatement attribuée en utilisant le signe **=**. Alternativement, si une ***variable*** de ***fonction*** est destinée à recevoir un ***argument*** transmis à la ***fonction*** au moment de son appel, cette ***variable*** peut être définie directement dans les **()** suivant le *nom* de la ***fonction*** (**function** *nom* **(***variable***)** **{***instruction* **;}**).

Le nom d'une ***variable*** (ou d'une ***fonction***) ne peut pas être identique à celui d'un *mot-clé* du *JavaScript*. Par convention, le nom commence toujours par une lettre *minuscule*. Toujours par convention, la séparation des « mots » dans un nom se fait par l'usage d'une *majuscule* plutôt que d'un **_**, mais en aucun cas par un *espace*. En *JS*, la *casse* est respectée, par conséquent les ***variables***

nom, *nOm* et *nOM* sont 3 ***variables*** différentes. Il est possible d'utiliser des *chiffres* dans les noms (mais pas commencer par un *chiffre*), mais pas d'autres *caractères* (sauf **_** et **$**).

Les types de variables :

Les ***variables*** *JS* ne sont pas limitées à contenir qu'un seul *type* de donnée. En fait, une même ***variable*** peut contenir successivement un *nombre*, puis une *chaine de caractère*, puis un *élément* du ***DOM***. Les principaux types sont : *numérique*, *texte*, *booléen*, *tableau* et des *objets* (notamment du ***DOM***).

Le type *numérique* (*number*) est simple. Comme souvent en informatique, la *virgule* des nombres décimaux est substituée par un *point*. Les opérations mathématiques classiques (**+**, **-**, *****, **/**) peuvent être réalisées sur ce type de ***variables***.

Le type *texte* (*string*) est également assez simple. En *JavaScript*, on peut utiliser les *guillemets* **"** ou les *apostrophes* **'** pour délimiter une valeur *textuelle*. Lorsque le texte contient des *apostrophes* ou des *guillemets*, il faut les précéder d'un *backslash* **** pour éviter de mettre fin prématurément à la valeur. De fait, pour mettre un **** dans la valeur, il faut le précédé lui-même d'un **** (donnant donc ****).

Le type *booléen* est très simple, il accepte seulement 2 valeurs : **true** (*vrai*) et **false** (*faux*).

En tant que ***langage*** orienté ***objet***, le *JavaScript* permet de créer des ***objets***. Ceux-ci peuvent être notamment des *tableaux* (*array*), qui sont des ***variables*** à plusieurs ***valeurs***. Pour déclarer un *tableau*, il suffit de mettre les différentes ***valeurs*** (séparées par une *virgule*) entre *crochets* : **var** *monTableau* **= [***valeur1***,** *valeur2***,** *valeur3***];**. Pour plus de formalisme, on devrait déclarer le *tableau* avant de lui assigner des valeurs, ce qui donnerait **var** *monTableau* **= new Array (***valeur1***,** *valeur2***,** *valeur3***);**, mais le *JavaScript* est un ***langage*** souple, ce n'est donc pas nécessaire. Pour accéder à une ***valeur*** particulière, on l'appelle en indiquant *l'indice* de la valeur moins un. Ainsi *monTableau***[1]** permet d'obtenir la *valeur2*. Finalement, il est possible d'imbriquer un *tableau* dans un autre *tableau*.

Les *indices* des valeurs des *tableaux* peuvent être définis par des *étiquettes* pour éviter d'utiliser des *chiffres*. Pour cela il faut commencer par déclarer le *tableau* (**var** *monTableau* **= new Array();**) puis d'attribuer les valeurs pour chaque étiquette, que l'on crée au fur et à mesure (*monTableau***["***etiquette1***"] =** *valeur1***;** *monTableau***["***etiquette2***"] =** *valeur2***;**).

Finalement, il est possible d'associer une ***variable*** à un ***objet*** du ***DOM***. Pour cela, on utilise la ***méthode*** la plus adaptée. Il en existe plusieurs, mais dans cette introduction nous ne retiendrons que la ***méthode*** **.getElementById()** qui permet de récupérer la ***balise*** qui possède la bonne ***identité*** (**id**) :

var *monObjet* **= document.getElementById("***identité***") ;**

Les fonctions :

Les ***fonctions*** sont des sous-***scripts***, qui peuvent être exécutées à la demande. Les ***fonctions*** peuvent être imbriquées les unes dans les autres. Elles peuvent être *anonyme* ou porter un *nom*. On ne peut pas appeler une ***fonction*** *anonyme*, mais celles-ci permettent de regrouper plusieurs ***instructions*** et de les « exécuter » en dehors des autres ***éléments***. Lorsqu'on définit une ***fonction***, celle-ci n'est pas exécutée et ne se termine donc pas par un **;**. Pour lancer son exécution, soit on met son *nom* dans une ***instruction*** (*maFonction***();**) soit on l'assigne à une ***variable*** (*maVariable***=***maFonction***();**).

Fonction anonyme :

Les ***fonctions*** *anonymes* sont utilisées (entre autre) pour exécuter immédiatement une série ***d'instructions*** comme par exemple au chargement de la *page* : **window.onload = function () {***instruction1* **;** *instruction2* **;};**

Une ***fonction*** *anonyme* peut être lancée sans devoir l'associer à un ***événement*** ou à une ***variable***. Il suffit pour cela de la placer entre **()** et de terminer ***l'instruction*** par **() ;** :

(function () {*instruction* **;}) () ;**

Créer une fonction.

Créer une ***fonction*** consiste à la déclarer après le *mot-clé* **function**. Après le *nom* de la ***fonction***, on place entre *parenthèses* les ***arguments*** qui doivent être transmis au moment de son appel. S'il n'y en a pas, on laisse les *parenthèses* vides. Ces ***arguments*** sont des ***variables*** *locales* de la ***fonction***. Si des ***arguments*** sont *optionnels*, ils sont placés à la fin de leur déclaration. Les ***instructions*** sont ensuite placées entre des *accolades*. La ***déclaration*** ne se termine pas avec un **;** :

function *nomFonction* **(***argumentObligatoire***,** *argumentFacultatif***) {***instruction1* **;** *instruction2* **;}**

Les ***instructions*** d'une ***fonction*** peuvent en appeler d'autres. Une ***fonction*** (y compris *anonyme*) peut également renvoyer une ***valeur*** (notamment lorsqu'elle est appelée lors d'une assignation à une ***variable***) grâce à ***l'instruction*** **return**. Toutes les ***instructions*** situées en aval de ***l'instruction*** **return** seront ignorées ! Si on souhaite retourner plusieurs ***valeurs***, il faut alors recourir à un *tableau*. Il est d'usage de terminer une ***fonction*** qui ne retourne pas de ***valeurs*** par ***l'instruction*** **return false;**.

Il est possible de créer des nouveaux *types* ***d'objets*** en respectant la même *syntaxe*, mais nous n'en utiliserons pas d'autres.

Quelques fonctions (et méthodes) préexistantes :

Affichage d'une boite de dialogue : alert("texte à afficher");

La ***fonction*** **alert** permet d'ouvrir une boite de *dialogue* qui affiche ***l'argument***.

REMARQUE : En réalité, **alert** (comme les suivantes) est une ***méthode*** de ***l'objet*** **window** et non une ***fonction***. Comme il est d'usage de ne pas indiquer **window**, cette ***méthode*** se lance donc comme une ***fonction***, raison pour laquelle elle est appelée ici ***fonction***.

Affichage d'une boite de confirmation : confirm("texte de la demande de confirmation");

La ***fonction*** **confirm** permet de demander à l'utilisateur de confirmer quelque chose dont la question est passée en ***argument***. La ***fonction*** **confirm** retourne **true** si l'utilisateur clique sur *OK* et **false** s'il ferme la boite de dialogue ou clique sur *Annuler*.

Affichage d'une boite à question : prompt("texte de la question");

Lorsqu'on attend une valeur plus précise qu'un *booléen*, on utilise la ***fonction*** **prompt**, qui retourne le *texte* entré (même si c'est un *chiffre*). Cette ***fonction*** accepte également un second ***argument*** facultatif, qui est le texte affiché *par défaut* dans la zone de saisie.

Conversion d'un texte en nombre : parseInt("texte");

La ***fonction*** **prompt** renvoie un *texte*, ce qui complique les opérations mathématiques subséquentes. En effet, en additionnant les textes '2' et '3' (**total='2'+'3';**) on obtient '23' et non 5. Il peut donc être nécessaire de convertir un type *texte* en un type *numérique*, ce que fait la ***fonction*** **parseInt**.

Exercices :

1. Créez une ***fonction*** helloWorld qui ouvre une *boite de dialogue* contenant « Hello World ! », lorsqu'on clique sur le titre de la *page HTML*.
2. Demandez à l'utilisateur de vous donner deux *chiffres* puis afficher dans une *boite de dialogue* le résultat de l'addition de ceux-ci.
3. Demandez à l'utilisateur de vous donner un *chiffre* entre 1 et 7 et afficher dans une *boite de dialogue* le jour de la semaine auquel ce chiffre correspond, en utilisant un *tableau*.

Principales références :

Site du W3C : http://www.w3.org

DÉVELOPPER VOTRE SITE WEB, **F. Basmaison** *et coll.*, MicroApplication **2012**

DYNAMISER VOS SITES WEB AVEC JAVASCRIPT, **J. Pardanaud et S. De La Marck**, Site du Zéro **2012**