LANGAGE JAVASCRIPT

1 - Introduction

Le Javascript est un langage interprété et orienté objet qui s'exécute côté client ou côté serveur. Il est particulièrement adapté au World Wide Web puisqu'il fonctionne avec le HTML. Ce langage prend une place prépondérante dans l'environnement Internet. C'est grâce à lui que les pages Web s'enrichissent avec un contenu interactif et animé.

Les possibilités de H.T.M.L étant limitées, Javascript s'impose comme un complément indispensable. Il donne le contrôle à la fois sur le document et sur le navigateur. Il faut donc apprendre le langage et connaitre l'environnement dans lequel il s'exécute (dans notre cas le navigateur).

1 – 1 - A quoi sert javascript ?

Les principales caractéristiques du langage :

- Orienté objet (On manipule des objets tels que des fenêtres, des boutons, des images… on peut même en créer de nouveaux objets)
- Interprété (A l'inverse du C, Javascript est interprété au moment de l'exécution du script)
- Dérivé du C++ (Les fonctions, les opérateurs sont les mêmes mais simplifiés et plus tolérants)
- Portable (Indépendant du matériel, il peut être exécuté partout dès que le système est pourvu d'un interpréteur Javascript.
- Inséré dans des applications hôtes (Navigateur, applis propriétaires)

1 – 2 - Javascript côté client et côté serveur

Javascript peut s'exécuter côté serveur et côté client.

Le côté client est utilisé pour agrémenter des pages HTML d'un contenu plus dynamique et interactif, piloter le navigateur, ou très souvent valider les informations issues d'un formulaire avant de les diriger vers un serveur, ...

Le côté serveur, est réservé à des applications en relation avec JAVA ou la technologie ASP de Microsoft.

1 – 3 - Javascript et HTML

Un script peut se trouver au sein d'une page HTML grâce à la présence de la balise <SCRIPT>. Le corps du script sera soit directement dans la page HTML ou à l'extérieur. Dans ce cas un lien extérieur vers le fichier contenant le programme sera inséré dans la page HTML.

Balise Script

Cette balise marque le début d'un script dans la page *HTML*. Elle peut s'insérer autant de fois que nécessaire dans les balises *HEAD* ou *BODY*. Elle contient plusieurs attributs:

Attribut	Signification
type	Spécifie le langage de script utilisé. Cet attribut doit être un type MIME Internet
langage	Spécifie également le langage mais sous forme d'un identificateur texte simple et non standard
src	Indique la localisation du script lorsqu'il est séparé du document

Attributs de la balise SCRIPT

1er Exemple d'utilisation de la balise *SCRIPT* :

```
<SCRIPT type="text/javascript">
    Contenu du script
</SCRIPT>
```

2ème Exemple d'utilisation de la balise *SCRIPT* avec référence à un fichier externe :

```
<SCRIPT type="text/Javascript" src="mon_script.js"> </SCRIPT>
```

Le fichier « mon_script.js » doit, dans ce cas, se trouver à coté de la page dans laquelle on écrit ce script.

2 - Variables et Données

2 – 1 - Déclaration de variables (var)

La déclaration d'une variable peut se faire avant son utilisation grâce au mot-clé **var**. C'est la déclaration explicite.

```
var test ;
```

La déclaration peut être implicite. Il suffit d'affecter une valeur à la variable pour qu'elle soit créée.

```
test = true ;
```

On ne peut pas spécifier le type de la variable. Javascript est langage non typé. La variable a le type de la valeur qu'on lui a affecté.

Remarque :

Javascript est un langage non typé. C'est-à-dire qu'on ne déclare pas le type des variables. Les variables ont le type de la valeur qu'on leur affecte.

2 – 2 - Types primitifs

Booléen (Boolean)

Le type booléen ne peur prendre que 2 valeurs (True et False)

```
test = true;
test = false;
```

Nombre (Number)

Il n'y a pas de distinction entre les réels, les entiers. Un nombre peut-être codé de différentes manières :

```
nombre1 = 10.25;
nombre2 = 17;
nombre3 = 1.05e+14 ;
nombre4 = 0xFF52 ;  // Nombre hexadécimal
```

Le nombre est limité aux valeurs inférieures à 10^{308} environ.

Chaîne (String)

Les chaînes doivent être délimitées par des guillemets simples ou doubles. On utilise les guillemets simples dans le cas où une instruction Javascript est exécutée au sein d'une propriété d'une balise HTML (Appel à une fonction par exemple)

```
chaine = "Bonjour";
```

Ou

```
<A HREF="javascript:afficher('Bonjour')"> Cliquez sur ce lien </A>
```

Dans cet exemple le lien permet d'exécuter le script « afficher('Bonjour') ». La fonction « afficher » admet pour argument une chaine de caractère. On doit entourer la chaine « Bonjour » par des guillemets simples pour éviter le conflit avec les guillemets de l'attribut HTML « href ».

Les chaine de caractères sont représentées par un objet « String » qui a des attributs et méthodes. Voyez un peu la souplesse du langage javascript :

```
// crée un nouveau objet
var   s = new String("Le monde est petit") ;
/* affiche une boite de message (La méthode toString() est la
méthode par défaut) */
alert(s) ;
```

Ou :

```
// crée un objet String implicitement
s = "Le monde est petit" ;
//transforme la chine en majuscule puis l'affiche.
alert(s.toUpperCase()) ;
```

Ou tout simplement :

```
Alert("Le monde est petit".toUpperCase()) ;
```

Nombres et types particuliers

Javascript reconnaît trois nombres particuliers (derniers navigateurs) :

Infinity, **-Infinity** et **NaN** (Not a Number),

Et deux types particuliers :

Null (variable non initialisée) et **Undefined** (variable non définie)

Tableaux

Un tableau s'utilise pour stocker des données différentes dans une seule variable dont l'indice varie. Pour déclarer un tableau, on utilise le mot clé **new** :

```
var mois= new Array(12);
mois[0] = "Janvier";
mois[1] = "Fevrier";
mois[2] = "Mars";
…
```

ou encore :

```
var mois= new Array("Janvier", "Fevrier",  "Mars", … );
```

Les éléments d'un tableau ne sont pas forcément de même type. Un tableau peut très bien contenir des chaînes et des nombres ou même des références à des objets.

Date

Il s'agit d'un type d'objet permettant de traiter les dates et les heures. Cet objet permet connaître l'heure système de l'ordinateur client et possède des méthodes pouvant traiter les formats des dates et des heures.

```
// Crée un objet de type Date qui contient la date du jour.
Var d = new Date() ;
// affiche la date.
alert(d) ;
```

Conversion de types

Il existe des fonctions de conversion de type qui permettent de spécifier explicitement la conversion à effectuer. Voici des exemples de conversion explicites ou implicites :

```
var a = String(21.34)     // Donne a = "21.34"
var b = Number ("12.56")  // b = 12.56
var c = parseInt("12.56") // c = 12
var d = ParseFloat("12.56")     // d = 12.56

var e = + ("0xFE99");     // e = 65177
var f = "14.1" – 5 ;      // d = 9.1
```

Opérateurs

Les opérateurs arithmétiques, relationnels, logiques, bit à bit ou spéciaux obéissent aux règles rencontrées en langage C. Les opérateurs sont utilisés pour effectuer des tests ou manipuler des données. Il est possible de combiner plusieurs opérations en une seule. Voici un récapitulatif des principaux opérateurs :

Contrôle de flux

Héritées du langage C , les mécanismes de contrôle de flux donnent au programmeur la possibilité d'effectuer des boucles ou des clauses if else. (Voir les références du langage)

3 - Les fonctions

Les fonctions permettent de subdiviser les tâches à effectuer en petits sous-programmes. Ces fonctions peuvent prendre des paramètres et renvoyer tout type de données.

```
function ma_fonction(mes_arguments)
{
  // instructions
  //et éventuellement return pour renvoyer une données
}
```

Voici une fonction renvoyant une chaîne dépendant du nom passé en paramètres

```
function bonjour (nom)
  {
  return (" Bonjour  " + nom);
  }
```

Dans javascript les fonctions sont des objets. Voyez l'exemple suivant :

```
// Définition de la fonction
function somme()
{
var total = 0;
for (i=0;i<somme.arguments.length;i++)
        total+=somme.arguments[i];
return (total);
}
alert(somme(1,6,9));
```

Dans cet exemple, on ne connaît pas à l'avance le nombre d'arguments de la fonction. L'objet « **Function** » a un attribut « arguments » qui contient un tableau de tous les arguments envoyés à la fonction.

4 - Les classes dans Javascript

On peut créer des classes en javascript en créant directement le constructeur de la classe. Voici un exemple simple de classe :

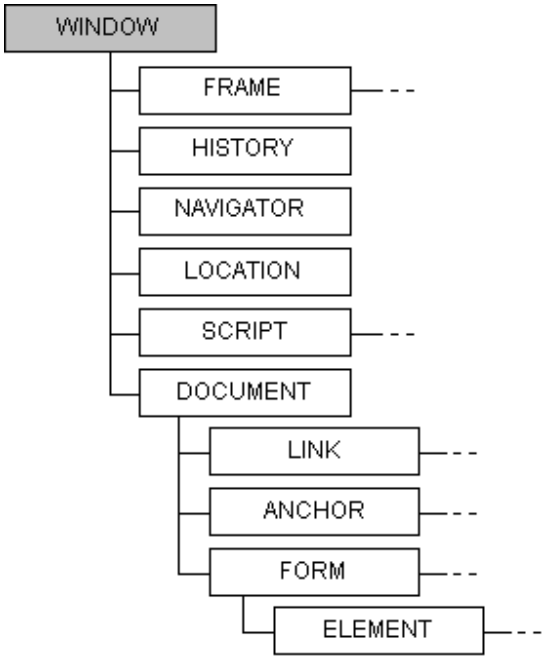
```
// Définition du constructeur
function  ma_classe(n)
{
this.nom = n ;  // Une propriété.
this.afficher = afficher_le_nom ; // Une méthode.
}

// Définition de la méthode.
function afficher_le_nom ()
{
alert(this.nom) ;
}
// Création de l'objet
var personne = new ma_classe("azer") ;
// Appel de la méthode
personne.afficher() ;
```

5 - Modèle objet du navigateur

5 – Généralités

L'environnement d'exécution de javascript est le navigateur Web. Il est alors important de connaître le modèle objet du navigateur pour pouvoir interagir avec ses objets. La plupart des navigateurs exposent un modèle qui ressemble à celui-ci :



L'objet « **window** » est l'objet de premier niveau. Il expose un certain nombre de méthodes, de propriétés et d'événements.

Méthodes :

alert, confirm, prompt, open, close, setTimeout, clearTimeout, navigate.

Evénements:

onload, onUnload

Propriétés :

name, parent, opener, self, top, location, defaultStatus, status, frames, history, navigator, document.

Pour invoquer une méthode de l'objet « **window** » on peut écrire par exemple :

```
window.alert("le monde est petit!") ;
```

Ou tout simplement :

```
alert("le monde est petit!") ;
```

Car l'objet « **window** » est global.

Pour écrire un message dans la barre de statut :

```
status = "Un message !" ;
```

Pour ouvrir une nouvelle fenêtre :

```
open("http://123.ma");
```

Bien sûr, la méthode « open » a d'autres arguments qui permettent, entre autre, de personnaliser la fenêtre « **popup** ».

Les événements sont utilisés en conjonction avec le code HTML. Les événements de l'objet « **window** » s'utilisent dans la balise « **body** » de la façon suivante :

```
<body onload="code_javascript_a_executer">
```

Dans cet exemple "code_javascript_a_executer" peut représenter une fonction ou une suite d'instructions javascript séparées par un « ; ».

L'objet « **document** » va nous permettre d'accéder et de manipuler les données de notre page HTML. Il expose lui aussi des propriétés, méthodes et événements. Vu que les fichiers HTML sont hiérarchiques, le navigateur web construit un arbre d'objets dont le nœud racine est l'objet « **document** ». Ce dernier aura alors un ensemble de propriétés sous forme de collections (tableau d'objets) qui représenteront par exemple l'ensemble des liens de la page (Links qui est tableau d'objet Link) ou l'ensemble des Images de la page (Images qui est un tableau d'objets Image).

Voyons un exemple de page HTML.

```
<html>
<head> <title>Ma page</title></head>
<body bgcolor="yellow">
<p>
   <img src="mon_image.jpg" width="50" height="50">
   Le monde est petit….<br />
   <a href="ma_page.html"> Aller à ma page. </a><br />
   <a href=""> Encore un lien </a>
</p>
<hr>
<script type="text/javascript">
  document.images[0].width=200;  // ligne 1
  document.bgColor="white";      // ligne 2
  document.links[1].href="http://yahoo.fr";
  document.write(document.images[0].src );
</script>
</body>
</html>
```

Dans cet exemple, on accède au contenu html et on le modifie en agissant sur les attributs de balises. « **bgColor** » est un attribut de body et au même temps une propriété de l'objet « document ». Dans la dernière ligne, nous avons utilisé la méthode « write » qui permet d'insérer du contenu dans le document.

5 – 2 – Gestion des événements

Pour créer de l'interactivité dans une application web, on a besoin de gérer les différents événements provoqués en général par l'utilisateur par exemple en cliquant sur un bouton HTML. Voici un exemple simple de gestion d'événement.

```
<input type="button" onclick="alert('La vie est belle')"
value="Cliquez">
```

Ici « **onclick** » est un attribut HTML de la balise « input » de type « **button** ». La valeur de cet attribut est du code javascript. « **alert** » est une méthode l'objet « **window** ».

Un deuxième exemple :

```
<input type="text" onfocus="this.value=''" value="Rechercher">
```

Dans cet exemple, quand le champ de texte reçoit le focus, le texte qu'il contient s'efface automatiquement. Par opposition à « **onfocus** » l'événement « **onblur** » est déclenché quand le focus

quitte le champ de texte. A l'instar d'autre langage orienté objet, le mot clé « **this** » désigne l'objet courant. Ici c'est le champ de texte.

Un troisième exemple :

```
<form name="mon_formulaire">
A : <input type="text" name="a" value=""><br>
B : <input type="text" name="b" value=""><br>
R : <input type="text" name="r" value=""><br>
<input type="button" value="Somme"
onclick="this.form.r.value=this.form.a.value+this.form.b.value">
</form>
```

Dans cet exemple, nous voulons calculer la somme des nombres saisis dans les champs « **a** » et « **b** » puis afficher le résultat dans le champ « **r** ». L'opération s'effectue quand on clique sur le bouton. L'événement est donc associé au bouton.

Pour accéder aux trois champs de texte à partir de l'objet « **button** », on a utilisé la propriété « **form** » qui désigne le formulaire dans lequel se trouve.

Voici d'autres manières pour accéder au contenu du champ « a » à partir de l'objet « **button** » :

```
this.form.a.value ;  // déjà vue
document.mon_formulaire.a.value ;
document.forms[0].a.value ;
document.forms[0].elements[0].value ;
```

Le champ « **a** » étant le premier élément du formulaire et « **mon_formulaire** » étant le premier formulaire du document.

6 – Applications de javascript

6 – 1 – Validation de formulaires.

Elle consiste en la vérification du contenu des champs obligatoires avant la soumission du formulaire. La balise html « **form** » réagit à l'événement « **onsubmit** » qui est déclenché quand on appuie sur le bouton « **submit** ». En gérant cet événement on peut faire des traitements sur le client avant la soumission du formulaire.

Voici un exemple simple :

```
…
<script type="text/javascript">
function valider()
{
  if (document.mon_formulaire.mon_text.length>=5)
     retur true ;
  else
   {
     alert("Au moins 4 caractères !");
      return false;
  }
…
<body>
<form name="mon_formulaire" onsubmit="return valider()">
<input type="text" name="mon_text"> <br>
<input type="submit" value="envoyer">
</form>
…
```

Remarquez que la fonction doit retourner une valeur booléenne. Et que dans l'attribut « **onsubmit** » on a utilisé le mot clé « **return** ». Si la longueur de la chaine saisie est supérieure ou égale à cinq le formulaire est soumis, sinon une boite de message (alert) apparait.

L'objet « **form** » a des propriétés qui correspondent aux attributs html de la balise « **form** ». Il a aussi des méthodes parmi lesquelles on cite la méthode « **submit** » qui soumet le formulaire. Pour soumettre un formulaire, on n'est pas obligé d'avoir un bouton « **submit** ». On peut utiliser un bouton standard « **button** » ou même un lien hypertexte qui appelle la méthode « **submit** » du formulaire concerné.

6 – 2 – Javascript et les styles CSS

Les objets (**DOM** =Document Object Model) construits par le navigateur ont une propriété « **style** » qui correspond à l'attribut « **style** » des balises HTML .

Prenons un exemple :

```
<p name="mon_paragraphe"> La vie est belle.</p>
<input type="button" value="colorer P"
onclick="javascript:document.getElementById('mon_paragraphe').styl
e.background='red'">
<br>
<input type="button" value="Masquer P"
onclick="javascript:document.getElementById('mon_paragraphe').styl
e.visibility='hidden'">
```

Cet exemple présente deux boutons. Le premier permet de changer la couleur d'arrière plan du paragraphe. L'autre permet de masquer le paragraphe. On note ici l'utilisation de la puissante méthode

« **getElementById** » de l'objet « **document** ». Celle-ci nous permet d'accéder à n'importe quelle balise html pourvues d'un attribut html « **id** ». L'attribut « d » doit donc être unique dans la page.

Connaissant la syntaxe des attributs CSS, on peut deviner facilement les propriétés javascript correspondantes :

CSS :

```
font-color :red ;
forder-left-style :solid ;
```

Javascript :

```
document.getElementById(ma_balise').style.fontColor="red" ;
document.getElementById(ma_balise').style.borderLeftStyle="solid"
;
```

On enlève les tirets des noms d'attributs CSS et met la lettre après le tiret en majuscules.