Cours JavaScript - Partie 1

Les caractéristiques du langage JavaScript

- C'est un **script** (un script, c'est tout simplement *un bout de code qui a une tâche précise*).
- Le **script** n'est **pas compilé** (pas besoin de compilateur)
- Relativement **limité** : il se limite plus ou moins à la page web sur laquelle il se trouve.
- Il est **interprété par le navigateur du visiteur**

Insertion du JavaScript dans une page HTML

Il y a plusieurs méthodes pour insérer des scripts JavaScript dans une page Web :

1- Directement dans les balises

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

Pour insérer le code dans une balise, une nouvelle propriété est nécessaire. Il s'agit du **gestionnaire d'événements**.

Exemple : Test JS

ou Test JS

Autres gestionnaires d'événements

- ondblclick : lors d'un double clic
- onmouseover : au passage de la souris
- onmouseout : lorsque la souris "sort" de cet élément
- …

Gestionnaires d'événements de la balise <body> :

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

Exemple : <body **onload="alert('Bonjour');" onunload="alert('Au revoir');"**>

2- Entre les balises <script> et </script>

La seconde solution consiste à écrire le script entre deux balises spécifiques, <script> et </script>.
<script type="text/javascript"> ou <script language="javascript"> ou <script>

```
  // Script JS
</script>
```

Exemple :

```
<body>

<!-- script pour le début du chargement -->

        <script type="text/javascript">

                alert('Début du chargement de la page');

        </script>

 <!-- ici se trouve le contenu de la page web -->

------------

 <!-- script pour la fin du chargement -->

        <script type="text/javascript">

                alert('Fin du chargement de la page');

        </script>

</body>
```

3- Via un fichier externe

Comme pour le CSS, on peut très bien placer du code JavaScript dans un fichier externe d'extension ".js".

Syntaxe d'importation : <script type="text/javascript" src="script.js"></script>

Les variables

Une variable est un objet repéré par son nom, pouvant contenir tout type de données, qui pourront être modifiées lors de l'exécution du programme.

En JavaSript les noms de variables doivent répondre à certains critères :

1. un nom de variable doit commencer par une lettre (majuscule ou minuscule) ou un _ "underscore".
2. un nom de variable peut comporter des lettres, des chiffres et les caractères _ et - .
3. les espaces ne sont pas autorisés
4. Les noms de variables ne peuvent pas être les noms suivants, qui sont des noms réservés : abstract, boolean, break, byte, case, catch, char, class, const, continue, debugger, default, delete, do, double, else, export, extends, false, final, finally, float, for, function, goto, if, implements, import, in, infinity, instanceof, int, interface, label, long, native, new, null, package, private, protected, public, return, short, static, super, switch, synchronized, this, throw, throws, transient, true, try, typeof, var, void, volatile, while, with …

Créer & lire une variable

Les variables JavaScript peuvent être déclarées de 4 manières :

1. Automatiquement
2. En utilisant le mot clé : var
3. En utilisant le mot clé : let
4. En utilisant le mot clé : const (pour les constantes)

Créer la variable : nom-variable, var nom-variable; ou let nom-variable;
Lire une variable var nom-variable = "Javascript";
alert(nom-variable);
ou vous pouvez afficher le contenu de la variable dans la console de votre navigateur.
console.log(nom-variable);

Les types de variables

Les chaînes de caractères
var msg = "Ceci est un petit test";
alert(msg);
Les nombres

- les nombres entiers
- les nombres à virgule

Les booléens
Désigne un paramètre qui peut avoir comme valeur true ou false.

Le type de données objet peut contenir :

- Un objet
- Un tableau
- Une date

Récupération de type d'une variable (typeof)

var number = 2;
alert(typeof number); // Affiche : « number »

```
var text = 'Mon texte';
alert(typeof text); // Affiche : « string »

var aBoolean = false;
alert(typeof aBoolean); // Affiche : « boolean »

alert(typeof nothing); // Affiche : «undefined» (variable inexistante ou déclarée mais ne contient rien)
```

La conversion des types

Conversion type "string" en" number"

Le concept est simple : il suffit de convertir la chaîne de caractères en nombre. Pour cela, vous allez avoir besoin de la fonction parseInt() qui s'utilise de cette manière :

```
var text = '2017';
var number;
number = parseInt(text);
alert(typeof number); // Affiche : « number »
alert(number); // Affiche : « 2017 »
```

Conversion type "number" en "string

Pour convertir une chaîne de caractères en nombre, nous pouvons utiliser la méthode toString() :

```
var number = 2,
text = number.toString();
alert(typeof text); // Affiche : « string »
alert(text); // Affiche : "2017"
```

Exemples :

- parseInt("12.9 Dhs") Affiche 12
- parseFloat(" 12.9 Dhs ") Affiche 12.9
- parseFloat("3,14") Affiche 3 : il faut utiliser le point au lieu de la virgule. La conversion va donc s'arrêter après le "3".

Les opérateurs arithmétiques

Opérateur	Signe
addition	+
soustraction	-
multiplication	*
division	/
modulo	%

Les autres opérateurs arithmétiques :

- +=
- -=
- *=
- /=
- %=

Les opérateurs d'incrémentation

Incrémentation X++

Décrémentation X—

Les caractères spéciaux

On peut également insérer des retours à la ligne, ainsi que des tabulations ou autres.
Voici les caractères spéciaux les plus courants :

- **\n** qui insère un retour à la ligne.
- **\t** pour insérer une tabulation
- **\r** le retour de chariot
- **\b** revenir d'un caractère (backspace)

La concaténation

Une concaténation consiste à ajouter une chaîne de caractères à la fin d'une autre, on utilise simplement le symbole de concaténation **+** entre chaque morceau

```
var age = 18;
alert("Vous avez " + age + " ans");
```

Demander une chaîne de caractère au visiteur

On demande l'âge du visiteur pour afficher ensuite dans une phrase.

```
var age = prompt("Quel âge avez-vous ?"); // on demande l'âge
alert("Vous avez " + age + " ans");          // on affiche la phrase
```

Les conditions

Les opérateurs de comparaison

Permet de comparer diverses valeurs entre elles.

Opérateur	Signification
==	égal à
!=	différent de
>	supérieur à
>=	supérieur ou égal à
<	inférieur à
<=	inférieur ou égal à

Les opérateurs logiques

Opérateur	Type de logique	Utilisation
&&	ET	valeur1 && valeur2
||	OU	valeur1 || valeur2
!	NON	!valeur

Instructions de controle

Instruction if

Elle permet d'exécuter une série d'instructions lorsqu'une condition est réalisée.
La syntaxe de cette expression est la suivante.

```
if ( si condition réalisée ) {
```

```
//liste d'instructions
}
```

Instruction if else

L'expression if … else permet d'exécuter une autre série d'instructions en cas de non-réalisation de la condition.
La syntaxe de cette expression est la suivante.

```
if ( si condition réalisée ) {
//liste d'instructions
}
else {
// si non autre série d'instructions
}
```

Instruction else if

La syntaxe de cette expression est la suivante.

```
if ( si condition réalisée ) {
//liste d'instructions
}
else if ( sinon si condition réalisée ) {
// liste d'instructions
}
else {
// si non autre série d'instructions
}
```

Exemple :

```
var nombre = prompt("Entrez un nombre");
if(nombre == 20)
        alert("Nombre correct");
else
        alert("Nombre incorrect");
```

Instruction switch...case

L'instruction switch permet de faire plusieurs tests de valeurs sur le contenu d'une même variable.
La syntaxe de cette expression est la suivante :

```
switch (Variable) {
          case Valeur1:
          Liste d'instructions;
          break;
          case Valeur2:
          Liste d'instructions;
          break;
          default:
          Liste d'instructions;
          break;
}
```

Exemple :

```
var nombre = prompt("Entrez un nombre");
switch(nombre)
{
        case "20":
                alert("Nombre correct");
```

```
            break;
        default:
            alert("Nombre incorrect");
            break;
}
```

La fonction confirm()

L'utilisation de la fonction confirm() est simple, on lui passe en paramètre une chaîne de caractères qui sera affichée à l'écran et elle retourne un booléen en fonction de l'action de l'utilisateur.

```
if (confirm('Voulez-vous exécuter le code JavaScript de cette page ?')) {
   alert('Le code a bien été exécuté !');
}
```

Les fonctions

Les fonctions regroupent un ensemble d'instructions permettant d'accomplir un ensemble d'opérations.

La création d'une fonction, se fait par l'indication du nom de la fonction précédé du mot clé **function**

Syntaxe :

```
function Nom-Fonction (param1, param2, ...)
        {
        ... instructions ...
        }
```

L'appel d'une fonction se fait dans le programme principal après avoir déclarer la fonction. Une fonction est appelée grâce à son nom suivi des paramètres.

Exemple :

```
<html>
<head>
    <title></title>
    <script type="text/javascript">
        function message()
            {
             alert("Bonjour");
            }
    </script>
</head>
<body>
    <p><a href="#" onclick="message();"> Lien </a> </p>
</body>
</html>
```

Fonctions de base pour traiter les chaines de caractères :

- String length
- String charAt()
- String charCodeAt()
- String at()
- String [ ]
- String slice()
- String substring()
- String substr()

- String toUpperCase()
- String toLowerCase()
- String concat()
- String trim()
- String trimStart()
- String trimEnd()
- String padStart()
- String padEnd()
- String repeat()
- String replace()
- String replaceAll()
- String split()

Fonctions de recherche des chaines de caractères :

- String indexOf()
- String lastIndexOf()
- String search()
- String match()
- String matchAll()
- String includes()
- String startsWith()
- String endsWith()

Les boucles

Les boucles permettent de répéter des opérations élémentaires un grand nombre de fois sans avoir à réécrire le même code. Selon l'instruction de boucle utilisée, le nombre d'itérations peut être défini à l'avance ou être déterminé par une condition particulière.

La boucle while

```
while (condition) {
    instruction_1;
    instruction_2;
    instruction_3;
}
```

La boucle do while

```
do {
    instruction_1;
    instruction_2;
    instruction_3;
}
while (condition)
```

La boucle for

```
for (initialisation; condition; incrémentation) {
    instruction_1;
    instruction_2;
    instruction_3;
}
```

Exemple :

```
for(i=0;i<10;i++){

        document.write(i+"<br>");

}
```

Mot-clé "break" : pour arrêter la boucle d'un seul coup

```
compt=1;
while(compt<5){
        if(compt==4)
        break;
        document.write(compt+"<br>");
        compt++;
}
```

Mot-clé "continue" : Permet de mettre fin à une itération, mais attention, elle ne provoque pas la fin de la boucle : l'itération en cours est stoppée, et la boucle passe à l'itération suivante

```
var i = 0;
while (i < 5) {
  i++;
  if (i==3)
    continue;
  document.write(i+"<br>");
}
```

Les tableaux

Créer un tableau

Pour créer un tableau, on utilise l'istruction **new Array()**
Exemple : var table = new Array();

Initialiser un tableau

```
var tab1 = new Array("Taha", "Yahya", " Fatima Zahra");
var tab2 = new Array(2.5, 4.7, 3.14);
```

Longueur d'un tableau (Length)

```
var tab1 = new Array("Taha","Yahya"," Fatima Zahra ");
alert(tab1.length);
```

Trier un tableau (sort)

```
var tab1 = new Array("Taha", "Yahya", " Fatima Zahra ");
tab1.sort();
```

Inversé le contenu d'un tableau (reverse)

```
var tab1 = new Array("Taha", "Yahya", " Fatima Zahra ");
tab1.reverse();
```

Retourner un sous tableau (slice(début,fin))

```
var tab1 = new Array("Taha", "Yahya", " Fatima Zahra ");
tab1.slice(1,2);
```

Concaténer deux tableaux (concat(tableau))

```
var tab1 = new Array("Taha", "Yahya", " Fatima Zahra");
var tab2 = new Array(2.5, 4.7, 3.14);
tab1.concat(tab2);
```

join(séparateur) retourne une chaîne de caractères contenant les éléments d'un tableau séparés par la chaîne séparateur

```
var tab2 = new Array(2.5, 4.7, 3.14);
tab2.join('-------');
```

Lire un tableau de dimension 1

```
var tab = ["Taha","Yahya","Fatima Zahra"]; ou var tab = new Array("Taha","Yahya","Fatima Zahra ");
function lire(tab)
{
	var chaine = "Le tableau contient :"
	for(var i=0; i<tab.length; i++)
		chaine += "\n" + i + " -> " + tab[i];
	return chaine;
}
var test=lire(tab);
 document.write (test);
```

Pour les tableaux associatifs

```
var tab1 = {"nom":"Taha", "prenom":"Yahya", "adresse":"Fès"};
function lire(tab)
{
        var chaine = "Le tableau contient :";
        for(var indice in tab)
                chaine += "\n" + indice + " -> " + tab[indice];
        return chaine;
}
 var test=lire(tab1);
 document.write (test);
```

Lire un tableau de dimension 2

```
var matrice = [["Taha", "Yahya", "Fatima Zahra"],
                                ["Taha1", "Yahya1", "Fatima Zahra1"],
                                ["Taha2", "Yahya2", "Fatima Zahra2"]
                                ]; ou

var matrice = new Array (["Taha", "Yahya", "Fatima Zahra"],
                                ["Taha1", "Yahya1", "Fatima Zahra1"],
                                ["Taha2", "Yahya2", "Fatima Zahra2"]
                                );
function lire(grille)
{
for(var i=0; i<grille.length; i++){
   for(var j=0; j<grille[i].length; j++)
   {
      document.write("\n" + grille[i][j]);
   }
      document.write("<br>");
}
}
lire(matrice);
```

Pour les tableaux associatifs de dimension 2

```
var tab1 = {
   "ETD001": {"nom":"Taha", "prenom":"Yahya", "adresse":"fes"},
   "ETD002": {"nom":"Tah1a", "prenom":"Yahya1", "adresse":"fes1" },
   "ETD003": {"nom":"Taha2", "prenom":"Yahya2", "adresse":"fes2"}
} ou
var tab1 = {
   ETD001: {nom:"Taha", prenom:"Yahya", adresse:"fes"},
   ETD002: {nom:"Taha1", prenom:"Yahya1", adresse:"fes1" },
   ETD003: {nom:"Taha2", prenom:"Yahya2", adresse:"fes2"}
}
for (var key in tab1) {
         document.write(key +"<br>");
                                for(var subkey in tab1[key]){
           document.write( "\n" + subkey + " -> " + tab1[key][subkey]);
                                }
 document.write("<br>");
}
```