Table des matières

Chapitre 1 : Initiation à Javascript

I. Introduction

Brendan Eich

Membre du conseil d'administration de la fondation Mozilla

JavaScript (souvent abrégé **JS**) est un langage de programmation de scripts utilisé dans les pages web interactives. Le langage a été créé en *1995* par ***Brendan Eich*** pour le compte de *Netscape Communications Corporation*.

Le langage, actuellement a la version 1.8.5.

- Le Javascript est sensible à la casse, c'est-à-dire qu'il fait une différence entre un nom de variable contenant ou non des majuscules.

  Ainsi la fonction bonjour(); n'est pas la même fonction que Bonjour();.

- Comme en langage C, chaque instruction se termine par un point-virgule (;)

  Le point-virgule n'est pas obligatoire si l'instruction qui suit se trouve sur la ligne suivante mais, de préférence, il est conseillé de les utiliser

II. JavaScript & Java

Il ne faut pas confondre le **JavaScript** et le **Java**. En effet contrairement au langage Java, le code est directement écrit dans la page HTML, c'est un langage peu évolué qui ne permet aucune confidentialité au niveau des codes. D'autre part l'applet Java a été préalablement compilée, et une machine virtuelle permettant d'interpréter le *pseudocode* doit être chargée en mémoire (du côté du client) à chaque chargement de la page, d'où un important ralentissement pour les applets Java contrairement au JavaScript

Javascript	Java
Langage interprété	Langage pseudo-compilé (chargement d'une machine virtuelle)
Code intégré au HTML	Code (applet) à part du document HTML, appelé à partir de la page
Langage peu typé	Langage fortement typé (déclaration du type de variable)
Liaisons dynamiques : les références des objets sont vérifiées au chargement	Liaisons statiques : les objets doivent exister au chargement (compilation)
Accessibilité du code	Confidentialité du code

III. Insertion de code JavaScript dans une page HTML

1. Utilisation de la balise "script"

Un script est une portion de code qui vient s'insérer dans une page HTML. Le code du script n'est toutefois pas visible dans la fenêtre du navigateur car il est compris entre des balises (ou tags) spécifiques qui signalent au navigateur qu'il s'agit d'un script écrit en langage JavaScript. Les balises annonçant un code Javascript sont les suivantes :

```
<script  type="text/javascript"    language="javascript" >
                //Placez ici le code de votre script
</script>
```

On insère normalement toutes les fonctions et initialisations dans l'élément "head" de la page HTML (cela assure que les procédures soient connues, avant d'être appelées).

Exemple

programme *sayhello()* qui permettra d'afficher une petite fenêtre avec le texte: "Bonjour Tout le monde!!!"

```
<HEAD>
        <TITLE>Hello World avec JavaScript</TITLE>
        <script language="JavaScript" type="text/javascript">
                // ICI on definit une fonction JavaScript
                function sayhello() {
                        alert("Bonjour tout le monde!!! ")    }
        </script>
</HEAD>
<BODY>
```

2. Script Externe

Il est possible de mettre les portions de code dans un fichier externe. L'appel se fait via l'attribut **SRC**, placé dans la balise **<script>.**

Exemple

```
<html>
  <head>
    <title>   </title>
    <script type="text/javascript" src="MonFichier.js"></script>
  </head>
<body>
 </body>
</html>
```

IV. Syntaxes

1. Déclaration des variables

En JavaScript les variables ne sont pas typées. La déclaration peut se faire d'une manière explicite grâce au mot clef **var** ou d'une manière implicite.

Exemples

```
var i;   //Déclaration de variable d'une façon explicite
i = 2;
chaine = "bonjour";   //Déclaration de variable d'une façon implicite
bool = true;
```

Les variables ne doivent pas être des mots-clefs JavaScript : **var**, **form**, **int**, **document**, etc.

2. Définition des fonctions

Une fonction est introduite par ***function***.

```
function nom (arg0, arg1, ..., argN) {  //votre script.  }
   //ou
var nom = function (arg0, arg1, ..., argN) {   //votre script.  }
```

Exemples

```
function somme(n1, n2) {        return n1+n2;  }
alert(somme(5,10));     //15
          // ou bien
var somme = function (n1, n2) { return n1+n2; };
alert(somme(5,10));     //15
```

2.1. Portée locale d'une variable

```
function test() {
          var message = "hi";
}
test();
alert(message); // undefined
```

2.2. Portée globale

```
function test() {
          message = "hi";
}
test();
alert(message); // "hi"
```

3. Types

Il y a principalement trois types classiques de données *Boolean*, *Number*, *String* et un type complexe *Object* (liste de couples nom-valeurs). Il est possible de déterminer le type (courant) d'une variable avec l'opérateur *typeof* qui retourne l'une des valeurs suivantes :

- *undefined* si la valeur est indéfinie (variable déclarée mais pas initialisée ou variable non déclarée)
- *boolean*
- *number*
- *string*
- *object* si la valeur est un objet ou null
- *function* si la valeur est une fonction

4. Conversions numériques

Relatifs aux valeurs et conversions des nombres, on trouve 4 fonctions :

- *isNaN* détermine si un paramètre donné n'est pas un nombre
- *Number* effectue une conversion
- *parseInt* effectue une conversion en valeur entière
- *parseFloat* effectue une conversion en valeur réelle

Exemples

```
alert(isNaN(10));                    // false
alert(isNaN("10"));                  // false - peut être convertie
alert(isNaN("blue"));                // true - ne peut être convertie
var num1 = Number("hello world");    // NaN
var num2 = Number("00001");          // 1
var num3 = Number(true);             // 1
var num3 = parseInt("");             // NaN
var num4 = parseInt(22.5);           // 22
var num5 = parseInt("70",10);        // 70 - la bsase 10 est spécifiée
var num6 = parseFloat("22.5");       // 22.5
```

5. String

On utilise les quotes simples (*apostrophes*) ou doubles (*guillemets*) pour définir des valeurs chaînes de caractères. L'attribut *length* permet de déterminer la longueur d'une chaîne.

Exemples

```
var nom = "Ali";
var prenom = 'Mohamed';
alert(prenom.length);    // 7
var message = "toto a dit \"je suis malade\".";
```

Il est possible de transtyper une valeur en chaîne avec ***String().***

Exemples

```
var v1 = 10; alert(String(v1));          // "10"
var v2 = true; alert(String(v2));        // "true"
var v3 = null; alert(String(v3));        // "null"
var v4; alert(String(v4));               // "undefined"
```

Il existe de nombreuses fonctions sur les chaînes de caractères.

Exemples

```
var s = "hello world";
alert(s.length);                 // 11
alert(s.charAt(1));              // "e"
alert(s.charCodeAt(1));          // 101
alert(s.slice(3));               // "lo world"
alert(s.slice(-3));              // "rld"
alert(s.substring(3,7));         // "lo w"
alert(s.indexOf("o"));           // 4
alert(s.lastIndexOf("o"));       // 7
alert(s.toUpperCase());          // HELLO WORLD
alert(s + " !");                 // hello world !
```

6. Math

Il s'agit d'un objet définissant de nombreuses constantes et fonctions mathématiques.

Exemples

```
alert(Math.E);                           // la valeur de e
alert(Math.PI);                          // la valeur de pi
alert(Math.min(5,12));                   // 5
alert(Math.max(23,5,7,130,12));          // 130
alert(Math.ceil(25.3));                  // 26
alert(Math.floor(25.8));                 //25
alert(Math.random());                    // valeur aléatoire entre 0 et 1
var n = Math.floor((Math.random()*nb) + min);
alert(n);                                // valeur aléatoire entre min et nb (exclus)
```

D'autres fonctions :

- Math.abs(x) Math.exp(x) Math.log(x)
- Math.pow(x,y) Math.sqrt(x)
- Math.sin(x) Math.cos(x) Math.tan(x)

7. Opérateurs

Typiquement, ceux de C, C++ et java:

- incrémentation/décrémentation (++, --)
- arithmétiques (+, *, -, =, %)
- relationnels (>, <, >=, <=, ==, s! =) et (===, ! ==)
- logique (!, &&, ||)
- affectation (=, +=, -=, *=, /=, %=)

Exemples

```
var age = 10;
age++;
alert(age);                    // 11
alert(age > 10 && age < 20);   // true
alert(26 % 5);                 // 1
age*=2;
alert(age);                    // 22
```

8. Structures de contrôle

Elles sont très proches de celles de langages tels que C, C++ et Java. Pour rappel, les structures de contrôles sont de trois types :

- Séquence : exécution séquentielle d'une suite d'instructions séparées par un point-virgule
- Alternative : structure permettant un choix entre divers blocs d'instructions suivant le résultat d'un test logique
- Boucle : structure itérative permettant de répéter plusieurs fois le même bloc d'instructions tant qu'une condition de sortie n'est pas avérée

8.1. Alternatives (ou conditionnelles)

- L'instruction ***if*** sans partie ***else*** :

```
if (condition) instruction;
if (condition) { instruction; }
if (condition) {  instruction1;   instruction2;   ... }
```

Exemples

```
if (x >= 0) alert("valeur positive ou nulle");
...
if (note > 12 && note <= 14) {
        alert("bravo");
        mention="bien";
}
```

♠ L'instruction *if...else* :

```
if (condition) instruction1;
else instruction2;
```

```
if (condition) {
    instructions1;
} else {
    instructions2;
}
```

```
if (condition1)
{
    instructions1;
} else if (condition2)
{
    instructions2;
} else
{
    instructions3;
}
```

Exemple

```
if (rank == 1)
        medaille="or";
else if (rank == 2)
        medaille="argent";
else if (rank == 3)
        medaille="bronze";
```

♠ L'opérateur ternaire *?* :

Permet de remplacer une instruction **if...else** simple. Sa syntaxe (lorsqu'utilisée pour donner une valeur à une variable) est :

```
variable = condition ? expressionIf : expressionElse;
```

Elle est équivalente à:

```
if (condition) variable=expressionIf;
else variable=expressionElse;
```

Exemple

```
var civilite = (sexe == "F") ? "Madame" : "Monsieur";
```

✍ Cet opérateur est utile pour les expressions courtes.

♠ L'instruction *switch* :

```
switch (expression) {
        case valeur1 : instructions1; break;
        case valeur2 : instructions2; break;
        ...
        case valeurN : instructionsN; break;
        default: instructionsDefault;
}
```

✍ Le branchement par défaut n'est pas obligatoire.

8.2. Itératives (les boucles)

- L'instruction *while* :

```
while (condition) instruction;
while (condition) {  instruction1;   instruction2;   ...   }
```

Exemple

```
var num = 1;
while (num <= 5) {    alert(num);      num++;    }
```

- L'instruction *for* :

```
for (instructionInit; condition; instructionIter) instruction;
for (instructionInit; condition; instructionIter) { instruction1;   instruction2;  ...   }
```

Exemple

```
for (var num = 1; num <= 5; num++)    alert(num);
```

- L'instruction *do...while* :

```
do {
        instruction1;   instruction2;
        ...
} while (condition);
```

- L'instruction *forEach* pour les tableaux :

```
NomTableau.forEach (function(ValeurActuelle){
        instruction1;   instruction2; ……
});
```

Exemple

```
var numbers = new Array (1, 2, 3, 4, 5, 6);
numbers.forEach(function (number) {
        document.write(number + ' ');
});
```

- L'instruction *for-in* pour les objets :

```
for (var prop in window)
        document.writeln(prop);
```

- Certaines instructions permettent un contrôle supplémentaire sur les boucles :

break permet de quitter la boucle courante
continue permet de terminer l'itération en cours de la boucle courante

Exemples

```
var text = "";
var i;
for (i = 0; i < 10; i++) {
        if (i === 3) { break; }
        text += "Le nombre est " + i + "<br>";
}
document.writeln(text);
```

```
for (i = 0; i < 10; i++) {
if (i === 3) { continue; }
        text += "Le nombre est " + i + "<br>";
}
document.writeln(text);
```

Exercice : Ecrire le code Javascript qui détermine si un nombre entier x est parfait.

✍ Un nombre est parfait ssi il est égal à la somme de ses diviseurs stricts.

6 est parfait car 6 = 1 + 2 + 3.

9. Object

Utilisé pour stocker des données. Création d'une variable (de type Object) avec ***new Object()*** ou de manière littérale (énumération entre accolades).

Exemples

```
var personne = new Object();
personne.nom = "Wissam";
personne.age = 23;
//ou bien
var personne = { nom : "Wissam",  age : 23  }
```

✍ Il est possible d'utiliser les crochets pour accéder à un champ.

```
alert(personne.nom);
alert(personne["nom"]);
var attribut="nom"; alert(personne[attribut]);          //23
for (var pers in personne)
                document.writeln(pers + " : " + personne[pers] );
//nom : Wissam age : 23
```

10. Array

Les tableaux peuvent contenir des données de nature différente.

Exemples

```
var colors = new Array();                       // tableau vide
var colors2 = new Array(20);                    // tableau avec 20 cases
var colors3 = new Array("red","blue","green");  // 3 cases
var colors4 = ["red","blue","green"];           // notation littérale
```

✍ Le champ ***length*** indique la taille d'un tableau.

```
var colors = ["red","blue","green"];
alert(colors.length);                   // 3
colors[colors.length]="black";          // nouvelle couleur
colors[99]="pink";
alert(colors.length);                   // 100
```

```
alert(colors[50]):                    // undefined
colors.length=10;                     // plus que 10 cases
```

✍ De nombreuses méthodes existent sur les tableaux.

Exemples

```
var colors = ["red","blue","green"];
alert(colors);                  // red,blue,green
alert(colors.join(";"));        // red;blue;green
colors.push("black");           //Ajouter un élément à la fin du tableau
alert(colors);                  // red,blue,green,black
var item = colors.pop();        // Enlever le dernier élément du tableau
alert(item + " " + colors);     // black red,blue,green
var item2= colors.shift();      // Enlever le premier élément du tableau
alert(item2 + " " + colors);    // red blue,green
colors.unshift("yellow");       // Ajouter un élément au début du tableau
alert(colors);    // yellow,blue,green
```

✍ Il existe d'autres méthodes telles que *delete*, *concat*, *slice* et *splice*.

10.1. Réordonner les tableaux

On peut utiliser *reverse* et *sort*.

Exemples

```
var values = [0, 1, 2, 3, 4];
alert(values.reverse()); // 4,3,2,1,0
values = [0, 1, 15, 10, 5];
alert(values.sort()); // 0, 1, 15, 10, 5
alert(values.sort(function(a, b){return a-b})); // 0,1,5,10,15
alert(values.sort(function(a, b){return b-a})); // 15,10,5,1,0
```

11. Window

L'objet *window* représente la fenêtre du navigateur.

Exemples

```
var vW = window.open(
                    "http://www.isetb.rnu.tn/",
                    "ISETB",
                    "resizable,scrollbars,status,width=400, height=300");
if (vW == null)
        alert("fenetre bloquee");
else {
        vW.alert("Bienvenue au site de l'ISET");
        vW.resizeTo(600,600);
}
```

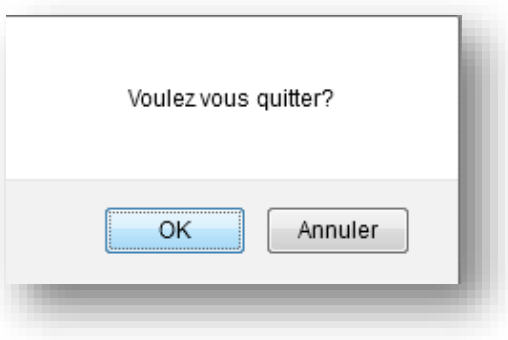
✍ Le navigateur peut être configuré de manière à empêcher de modifier son emplacement, de modifier sa taille ou encore d'afficher une fenêtre pop-up.

12. Boites de dialogues

Des boites de dialogues peuvent être ouvertes en utilisant les méthodes *alert*, *confirm* et *prompt*.

Exemple

```
if (confirm("Voulez vous quitter?"))
        alert("Au revoir");
else
            alert(" Suite… ");
var nom = prompt("Votre nom?","Ali");
if (nom != null)
            alert("Bienvenue " + nom);
```



13. Les évènements

Les évènements sont des actions de l'utilisateur, qui vont pouvoir donner lieu à une interactivité.

Ce sont les gestionnaires d'événements qui permettent d'associer une action à un événement.

La syntaxe d'un gestionnaire d'événement est la suivante :

```
onEvenement="Action_Javascript_ou_Fonction();"
```

13.1. Associer un évènement à un élément

Pour ce faire il suffit de mettre dans la balise le nom de l'évènement et de lui associer une fonction

Exemple :

```
<img src='une_image.jpg ' onclick='nom_de_lafonction(parametre1,parametre2)' />
```

On peut exécuter plusieurs fonctions sur le même évènement pour ce faire il suffit de rajouter un point-virgule.

```
<img src='une_image.jpg' onclick='fonction1() ; fonction2() ; fonction3()' />
```

13.2. Liste non exhaustive des évènements

Evénement	Description
onClick	Lorsque l'utilisateur clique sur un bouton, un lien ou tout autre élément.
onDbClick	Lorsque l'utilisateur double clique sur un bouton, un lien ou tout autre élément.
onLoad	Lorsque la page est chargée par le browser ou le navigateur.
onUnload	Lorsque l'utilisateur quitte la page.
onMouseOver	Lorsque l'utilisateur place le pointeur de la souris sur un lien ou tout autre élément.
ondrag	Lorsque l'utilisateur effectue un glisser sur la fenêtre du navigateur.
ondrop	Lorsque l'utilisateur effectue un déposer sur la fenêtre du navigateur.
onMouseOut	Lorsque le pointeur de la souris quitte un lien ou tout autre élément.
onMouseUp	Lorsque l'utilisateur relâche le bouton de la souris
onMouseDown	Lorsque l'utilisateur appuie sur le bouton de la souris sur un élément
onFocus	Lorsqu' un élément de formulaire a le focus c-à-d devient la zone d'entrée active.
onBlur	Lorsqu'un élément de formulaire perd le focus c-à-d que l'utilisateur clique hors du champ et que la zone d'entrée n'est plus active.
onChange	Lorsque la valeur d'un champ de formulaire est modifiée.
onSelect	Lorsque l'utilisateur sélectionne un champ dans un élément de formulaire.
onSubmit	Lorsque l'utilisateur clique sur le bouton Submit pour envoyer un formulaire
onError	Lorsqu'une erreur apparaît durant le chargement de la page.
onKeyDown	Lorsque l'utilisateur appuie sur une touche de son clavier.
onKeyPress	Lorsque l'utilisateur maintient une touche de son clavier enfoncée.
onKeyUp	Lorsque l'utilisateur relâche une touche de son clavier préalablement enfoncée.
onReset	Lorsque l'utilisateur efface les données d'un formulaire à l'aide du bouton Reset.
onResize	Lorsque l'utilisateur redimensionne la fenêtre du navigateur
onAbort	Lorsque le chargement d'une ressource a été annulé
…	

Il faut faire attention, les événements de fonctionnent pas sur tous les éléments.