Plan

Les bases de JavaScript

I. Introduction

- 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 Netscape Communications Corporation. Il est devenu un standard ECMA (European Computer Manufacturers Association) en 1997 sous le nom officiel ECMAScript ou en core ES.
- Javascript est actuellement à la version 7 (ES7) depuis Juin 2016.
- 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.
- 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.
- 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.

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

II. 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" >
            //Placez ici le code de votre script
</script>
```

L'attribut **type="text/javascript"** n'est pas obligatoire si on est en HTML5.

Le script peut être insérer n'importe où sur la page. Généralement on insère les fonctions JS utiles dans l'élément "**head**" et le code JS principal juste avant la balise de fermeture **</body>** de la page HTML.

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

```
<head>
        <TITLE>Bonjour avec JavaScript</TITLE>
        <script>// ICI on définit une fonction JavaScript
                function bonjour() {   alert("Bonjour tout le monde!!! ") ;}
        </script>
</head>
<body>
        < !--code HTML de la page -->
        <script>
                //code principal. Ici on appelle la fonction bonjour()
                bonjour() ;
        </script>
</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>
        <script src="mesFonctions.js"></script>
</head>
<body>
        < !-- code HTML de la page -->
        <script src="index.js"></script>
</body>
</html>
```

III. Syntaxes

1. Déclaration des variables

En JavaScript les variables ne sont pas typées. La déclaration peut se faire d'une manière explicite grâce aux mots clefs **var** ou **let** 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
let bool ;
bool = true;
```

- Les variables ne doivent pas être des mots-clefs JavaScript : **var**, **form**, **int**, **document**, etc.
- Le mot clé **let** permet de déclarer une variable de portée locale à un bloc (bloc if, boucle, ...)
- L'utilisation du **mode strict** du Javascript exige la déclaration de toute variables avant son utilisation. Pour ecrire en mode strict, il suffit de mettre la ligne suivante au début du script :

"use strict" ;

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";  //déclarée avec  var dans une fonction
}
test();
alert(message); // undefined
```

2.2. Portée globale

```
var message1 ; //variable globale déclarée en dehors des fonctions
function test() {
        message2 = "hi"; //variable globale (déclarée sans var dans la fonction)
}
test();
alert(message2); // "hi"
```

3. Types de variables

Javascript assigne des types aux variables en fonction de leurs contenus. 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

Expression	Résultat
typeof("ali")	string
typeof(3.14)	number
typeof(NaN)	number
typeof(false)	boolean
typeof([1,2,3,4])	object
typeof({name:'ali', age:34})	object
typeof(new Date())	object
typeof(function () {})	function
typeof(N1)	undefined
typeof(null)	object

4. Valeurs prédéfinies

JavaScript présente des valeurs prédéfinies que peut contenir une variable :

- ***undefined :*** si la valeur est indéfinie (variable déclarée mais pas initialisée ou variable non déclarée).
- ***NaN :*** *Not-a-Number* habituellement générée comme résultat d'une opération mathématique incohérente. Exp : a = "ali" / 3 →NaN
- ***+Infinity et –Infinity :*** *valeur représentant un nombre infini (dépassant la capacité des nombres à virgule flottant). Exp : var v=10e308→Infinity*
- ***null :*** *signifie l'absence de donnée dans une variable. Utile pour vider une variable (x=null).*

5. Fonctions prédéfinies

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

- ***isNaN :*** détermine si un paramètre donné n'est pas un nombre
- ***isFinite :*** Détermine si le paramètre est un nombre fini. Renvoie false si ce n'est pas un nombre ou l'infini positif ou infini négatif
- ***Number :*** effectue une conversion
- ***parseInt :*** effectue une conversion en valeur entière
- ***parseFloat :*** effectue une conversion en valeur réelle
- ***eval :*** exécute un code JavaScript à partir d'une chaine de caractères

Exemples

```
alert(isNaN(10));                    // false
alert(isNaN("10"));                  // false - peut être convertie
alert(isNaN("blue"));                // true - ne peut pas 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 base 10 est spécifiée
var num6 = parseFloat("22.5");       // 22.5
alert(isfinite(Infinity));           // false
alert(eval("5*2+3" ));               // 13
```

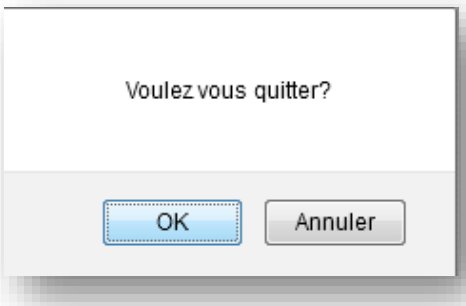
6. Les boites de dialogues

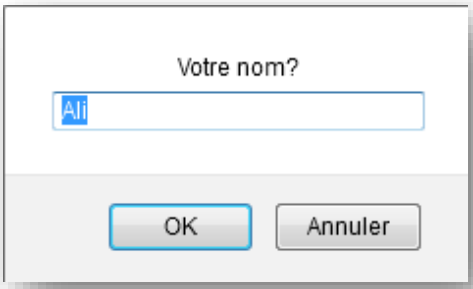
Des boites de dialogues peuvent être ouvertes en utilisant les méthodes ***alert***, ***confirm*** et ***prompt***.

- alert(*"Message")* → *afficher un message dans une boite de dialogue*
- prompt(*"Message")* → *afficher une invite pour saisir une chaine dans une boite de dialogue. Retourne la valeur saisie si le bouton "ok" est appuyé, "null " sinon.*
- confirm(*"Message")* → *afficher une boite de dialogue de confirmation (deux "oui" et "non"). Retourne true si le bouton "oui" est appuyé, false sinon.*

Exemple

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





7. Opérateurs

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

- incrémentation/décrémentation (++, --)
- arithmétiques (+, /, *, -, =, %)
- relationnels (>, <, >=, <=, ==, !=) et (===, !==)
- logique (!, &&, | |)
- affectation (=, +=, -=, *=, /=, %=)
- concaténation des chaines de caractères : +

Exemples

```
var age = 10;
age++;
alert(age);                          // 11
alert( age > 10 && age < 20);        // true
alert(26 % 5);                       // 1
age *= 2;
alert(age);                          // 22
x = 5 ;
alert(x == "5") ;          // true → égalité de valeurs
alert(x === "5") ;                   // false → pas d'égalité de valeurs et de types
alert(x !== "5") ;                   // true → inégalité de valeurs ou de types
```

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 *for-in* pour les objets :

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

IV. Les objets liés au navigateur

1. Window

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

Exemples

```
var vW = window.open("http://www.isetr.rnu.tn/","ISETR");
if (vW == null)
        alert("fenetrebloquee");
else {
        vW.alert("Bienvenue au site de l'ISET");
        vW.resizeTo(600,600);
}
```

Voici quelques propriétés et méthodes de l'objet *window* :

Propriété	Description
closed	Retourne une valeur booléenne indiquant si la fenêtre a été fermée ou non
document	Retourne l'objet Document de la fenêtre

location	Retourne l'emplacement de la fenêtre (adresse)
name	Modifie ou retourne le nom de la fenêtre
navigator	Retourne le navigateur de la fenêtre
opener	La référence de l'objet ayant créé la fenêtre courante
parent	La référence de la fenêtre mère (ayant créé la fenêtre courante)
self	Retourne la fenêtre courante

Methode	Description
alert()	Affiche une boite de dialogue (message + bouton ok)
blur()	Enlève le focus de la fenêtre
close()	Ferme la fenêtre courante
confirm()	Affiche une boite de confirmation (message + bouton ok + bouton annuler)
focus()	Affecte le focus à la fenêtre
getSelection()	Retourne un objet contenant le texte sélectionné par l'utilisateur
matchMedia()	Retourne une liste des MediaQuery représentant les CSS media queries string spécifiées
moveBy()/ moveTo()	Déplace la fenêtre par rapport à sa position actuelle/à une position spécifiée
open()	Ouvre une nouvelle fenêtre
print()	Imprime le contenu de la fenêtre courante
prompt()	Affiche une boite de dialogue pour la saisie
stop()	Arrête le chargement de la fenêtre

2. Document

L'objet *document* représente le document html chargé dans de la page.
Le tableau suivant représente quelques propriétés de l'objet document :

Propriété	Description
document.anchors	Retourne un tableau des éléments<a> dans le document ayant l'attribut name.
document.body	Modifie ou retourne l'élément body
document.characterSet	Retourne l'encodage du document
document.doctype	Retourne le type du document
document.documentElement	Retourne un élément du document
document.forms	Retourne un tableau de tous les éléments <form>du document
document.head	Retourne l'élément <head>du document
document.images	Retourne un tableau des éléments <img>du document
document.links	Retourne un tableau des éléments<a> and <area>du document ayant l'attribut href
document.scripts	Retourne un tableau des éléments<script>du document
document.title	Modifie ou retourne le titre du document

Le tableau suivant représente quelques méthodes de l'objet document :

Method	Description
document.write()	Ecrit du code HTML ou JavaScript dans le document
document.writeln()	Même que write() et ajoute un retour à la ligne
document.addEventListener()	Attache un gestionnaire d'évènement au document
document.createAttribute()	Créé un nœud attribute
document.createComment()	Créé un nœud commentaire avec un texte spécifié
document.createElement()	Créé un nœud élément

document.createTextNode()	Créé un nœud Text
document.getElementById()	Retourne l'élément ayant l'attribut ID attribute avec une valeur spécifiée
document.getElementsByClassName()	Retourne une liste des nœuds contenant les éléments avec une classe CSS spécifiée.
document.getElementsByName()	Retourne une liste des nœuds contenant les éléments avec une valeur spécifiée de l'attribut name.
document.getElementsByTagName()	Retourne une liste des nœuds contenant les éléments avec une valeur de balise spécifiée.
document.querySelector()	Retournele1er élément correspondant au sélecteur CSS en paramètre
document.querySelectorAll()	Retourne tous éléments correspondants au sélecteur CSS en paramètre
document.removeEventListener()	Supprime un gestionnaire d'évènement qui a été ajouté au document avec la méthode addEventListener()
document.renameNode()	Renomme un nœud spécifié

3. console

L'objet ***console*** *est très utile durant la phase d'écriture de code pour voir l'état de certaines variables. Il dispose d'une méthode* ***log()*** *qui permet d'évaluer et d'afficher n'importe quelle expression sur la console du navigateur.*

Exemple

```
for(let i =1 ; i<= 10 ; i++)
{
    console.log(i) ;
}
```