Javascript
1° PARTIE : LES BASES

1. Définition
2. Motivations
3. La balise <script>
4. Insertion de code javascript
5. Emplacement du code javascript
6. Exemple
7. Les variables
8. Expressions et opérateurs
9. Structures de contrôles
10. Les fonctions
11. Fonctions prédéfinies

1 . Définition

JavaScript® (qui est souvent abrégé en "JS") est un langage de **script léger, orienté objet**, principalement connu comme le langage de script des pages web.

Il peut être incorporé aux balises HTML, il permet alors d'améliorer la présentation et l'interactivité des pages Web côté client.

Javascript a été initialement développé par Netscape (LiveScript),

puis adopté en 1995, par la société SUN (qui développa aussi Java), pour prendre le nom de Javascript.

2 . Motivations

Ajout du contrôle au niveau du client

• Exemple : contrôle des entrées d'un formulaire avant son envoi.

¬ Scripts embarqués dans des documents HTML

• Contrôle des ressources du client (documents, formulaires, …) DONC moins d'intervention du Serveur www

Que contient JavaScript ?

- JavaScript contient une bibliothèque standard d'objets tels que

  Array, Date, et Math,

- ainsi qu'un ensemble d'éléments de langage tels que :

  les opérateurs,

  les structures de contrôles

  les instructions.

Documentation de référence

- https://developer.mozilla.org/fr/docs/Web/JavaScript

3 . La balise <script>

- La balise <script> permet d'indiquer au navigateur le début d'un script. La fin du script sera indiqué par la balise </script>.
- Exemple d'insertion d'un code js dans une page HTML :

```
<script language="JavaScript">
        … Instruction; …
</script>
```

4 . Insertion du code js

- **Interne** au document (on peut insérer du code javascript à plusieurs endroits dans un document HTML).
- **Externe** au document : le code javascript se trouve dans un fichier séparé portant le plus souvent l'extension **.js**. On devra préciser dans la balise <script> le chemin du fichier :

```
<script language="JavaScript" SCR="script.js">
    //inclut le fichier script.js puis l'exécute
</script>
```

¬ Cas particuliers …

4 . Insertion du code js (CAS PARTICULIERS)

Dans certains cas, l'usage de la balise <script> n'est pas obligatoire :

- Cas des événements où il faut simplement insérer le code à l'intérieur de la balise HTML comme un attribut de celle-ci :

  <body onLoad = "message()">

- Cas des liens HREF : on peut insérer du code javascript dans un lien **href** en précisant le mot clé **javascript:** :

  une fonction js

5 . Emplacement du code js

Il suffit de respecter les deux principes spécifiques aux scripts :

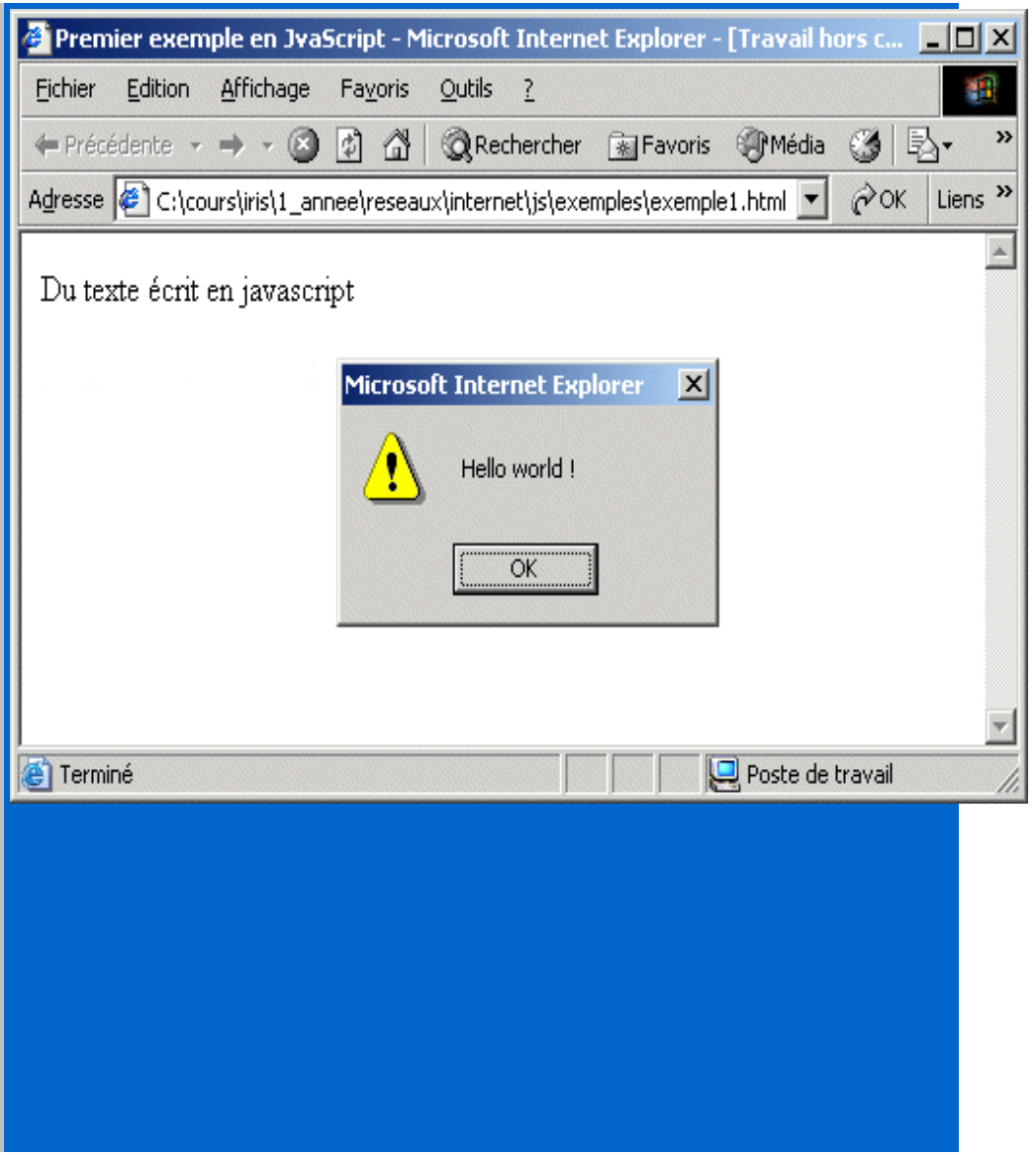
- N'importe où
- Mais là où il faut

Les instructions ne pourront s'exécuter que, si le navigateur possède à ce moment précis, tous les éléments nécessaires à son exécution.

Le code javascript est généralement inséré entre les balises <head> et </head>.

6 . Exemple

```
<html>
  <head>
    <title>Premier exemple en
    JavaScript</title>
    <script>
document.write("<p>Du texte
écrit en javascript</p>");
alert("Hello world !");
    </script>
  </head>
  <body>
    <p>Du texte écrit en
        HTML</p>
  </body>
</html>
```



7 . Les variables

La déclaration préalable des variables est facultative mais fortement conseillée.

```
var nbr = 10;  //un entier

var pi = 3.141;   //un réel

var str1 = "L'étoile";   //une chaîne de caractère

var str2 = 'brille';   //une autre chaîne de caractère
```

Il existe aussi le type booléen (true ou false)

```
var existe = true;
```

La portée des variables

- JavaScript a deux types de portées : globale et locale.
- Si vous déclarez une variable en dehors d'une définition de fonction, il s'agit d'une **variable globale** et sa valeur est accessible et modifiable dans tout le programme.
- En revanche, si vous déclarez une variable au sein d'une définition de fonction, il s'agit d'une **variable locale**. Elle est créée et détruite chaque fois que la fonction est exécutée ; en dehors de cette fonction, aucun code ne peut y accéder.
- Une variable locale peut posséder le même nom qu'une variable globale, mais elle est entièrement dissociée de celle-ci.

8 . Expressions et Opérateurs

- Expressions

Arithmétique : (3+4) * (56.7 / 89)

Chaîne : "Une étoile" + " " + "filante" (**concaténation**)

Logique : h2o = (temp<100) **?** "eau" **:** "vapeur";

- Opérateurs

Affectations : +=, -=, *=, /=, %=, &=, |=, <<=, >>=

Comparaisons : ==, !=, <, <=, >, >=

Arithmétiques : %, ++, --

Logiques : &&, ||, !

Bit : &, |, ^ (XOR), ~ (NON), <<, >>

9 . Structures de contrôle

La syntaxe est identique à celle du C :

- Bloc : { }
- **if else,**

  switch case,

  for, for in, while, do while,

  break, continue,

9-1 instructions conditionnelles

- Dans les instructions if, une condition est testée, et si la condition passe le test, le code JavaScript approprié est exécuté. Dans l'instruction if...else, un code différent est exécuté si la condition échoue au test.

```
// The test succeeds if either condition is true.

if ((dayOfWeek == "Saturday") || (dayOfWeek == "Sunday")) {
      return "I'm off to the beach!";
}   else {
     return "I'm going to work.";
}
```

9-1 instructions conditionnelles

- Dans les instructions if, une condition est testée, et si la condition passe le test, le code JavaScript approprié est exécuté. Dans l'instruction if...else, un code différent est exécuté si la condition échoue au test.

```
// The test succeeds if either condition is true.

if ((dayOfWeek == "Saturday") || (dayOfWeek == "Sunday")) {
      return "I'm off to the beach!";
}   else {
     return "I'm going to work.";
}
```

9-1 For

- L'instruction for spécifie une variable compteur, une condition de test et une action qui met à jour le compteur.

```
// Set a limit of 10 on the loop.
var howFar = 10;
// Create an array called sum with 10 members, 0 through 9.
var sum = new Array(howFar);

sum[0] = 0;

// Iterate from 0 through 9.

var theSum = 0;

for(var icount = 0; icount < howFar; icount++){

   theSum += icount;

   sum[icount] = theSum;
}
```

9-3 while

- Immédiatement avant chaque itération de la boucle, la condition est testée. Si le test est positif, le code à l'intérieur de la boucle est exécuté. Dans le cas contraire, ce code n'est pas exécuté et le programme continue à partir de la première ligne de code qui suit immédiatement la boucle.

```
var x = 0;
while ((x != 5) && (x != null)) {
    x = window.prompt("What is my favorite number?", x);
}
if (x == null)
    window.alert("You gave up!");
else
    window.alert("Correct answer!");
```

10 . Les fonctions

Définition et Appel :

```
function nomfonction( param1, ..., paramN)
        {  // code JavaScript …

            return variable_ou_valeur ;

        }

var res = nomfonction(var1, val2, varN) ;
```

◊ Remarque : passage des paramètres par valeur

- Exemple :

```
function isHumanAge(age) {

        if ((age < 0) || (age > 129)) { return false ; }
        else { return true ; }
}

if (!isHumanAge(age)) {

        alert("Vous ne pouvez pas avoir " + age + " ans !");

}
```

10 . Les fonctions avec un nombre d'arguments variables

```
function somme()
{
        var argv = somme.arguments;
        var argc = somme.arguments.length;
        var result=0;
          for(var i=0 ; i < argc ; i++)
                { result += argv[i]; }
         return result;
}
```

somme(1,2,3) retournera 6

somme(2) retournera 2

11 . Fonctions globales

Les fonctions globales, appelées globalement (et non par rapport à un objet)

parseInt : retourne la représentation entière de l'argument
parseFloat : retourne la représentatin réelle de l'argument

parseInt("12", 10)), parseInt("1100", 2)), parseInt("0xC", 16)) // retournent 12

parseFloat("1.23") // 1.23

parseFloat("1,23") // 1

parseFloat("1abc23") // 1

floatValue= parseFloat("abc789"); // NaN

if (isNaN(floatValue)) { alert("notFloat ! "); }

else { alert(floatValue); }

11 bis les Fonctions globales

- La méthode **eval()** permet d'évaluer du code JavaScript représenté sous forme d'une chaîne de caractères.

```
eval("2 + 2");          // renvoie 4
```

- **escape(), unescape() :** utilisées pour le codage des **URL** ou des **Cookies**

```
escape("äöü");      // "%E4%F6%FC"

unescape("%23") // retourne #
```

- La méthode **encodeURI()** encode un Uniform Resource Identifier (URI) en remplaçant chaque exemplaire de certains caractères par une, deux, trois ou quatre séquences d'échappement

12 opérateur typeof

- L'opérateur typeof renvoie une chaîne qui indique le type de son opérande.

type = typeof variable;

Javascript connaît 6 types différents :

undefined, boolean, number, string, function et object.

typeof true boolean

typeof 37 number

typeof 3.14 number

typeof "blabla" string

typeof new Date() object

typeof Math.sin function