JavaScript

420-JJA-JQ
Programmation mobile

Types et variables

- Javascript comporte 10 types
  - Object
  - String
  - Number
  - boolean
  - Array
  - Date
  - RegExp
  - Function
  - Undefined
  - Null

Conversion et test de type

- **parseInt()** et **parseFloat()** permettent de convertir une chaîne de caractères en nombre
- **Number()** transforme un objet en nombre à la manière de **parseFloat()** pour les chaînes de caractères.
- **string()** transforme un objet en chaîne de caractères.
- **isFinite()** permet de tester si la variable est bien un nombre fini
- **isNaN()** teste si le paramètre *n'est pas* un nombre (NaN = Not a Number)

Les boucles

- JavaScript possède quatre types de boucle, soit **for**, **for…in**, **while** et **do…while**.
- **for:** boucle utilisée lorsque l'on connais le nombre d'iterations que l'on veut faire. S'écrit sous la forme:
  - for(**variable numeric**; **condition d'arrêt**; **bond**)

    {

    code à exécuter

    }

```
1  for(var cptFor = 0; cptFor < 10; cptFor ++)
2  {
3      //affiche les nombres de 0 à 9
4      alert(cptFor);
5  }
```

Les boucles

- **for…in**: boucle permettant de parcourir chacun des éléments d'un tableau ou chacune des clés d'un objet. S'écrit sous la forme:
  - for (**variable** in **objet ou tableau**)
    {
    code à exécuter
    }

```
for(var elem in tableau){
  console.log(elem);
}
for(var key in objet){
  console.log(key + ": " + objet[key]);
}
```

ATTENTION, LORSQU'UTILISÉ AVEC UN TABLEAU, ÇA VA RETOURNER LES INDICES NON-NULL ET NON LES VALEURS!

Les boucles

- **while:** boucle utilisée lorsque l'on ne connais pas le nombre d'iterations que l'on veut faire. S'écrit sous la forme:
  - while(**condition d'arrêt**)
    {
        code à exécuter
    }
- **do…while:** boucle utilisée lorsque l'on ne connais pas le nombre d'iterations que l'on veut faire, mais qu'on sait qu'elle doit se faire au moins une fois. S'écrit sous la forme:
  - do
    {
        code à executer
    }while(**condition d'arrêt**)

Exemple While et do…while

```
var chaine;

do {
        chaine=prompt("Entrez une chaîne de caractères contenant le
        caractère \"");
}while (chaine.indexOf("\"")==-1);

alert("La chaîne entrée est \""+chaine+"\"");

var chaine2;

while ( chaine2.indexOf(("\"")==-1)
{
        chaine2=prompt("Entrez une chaîne de caractères contenant le
        caractère \"");
}

alert("La chaîne entrée est \""+chaine2+"\"");
```

Fonctions

- En JavaScript, les fonctions sont un type de données. Elles peuvent être passées en argument à d'autres fonctions et être mises dans des variables.
- Il existe deux catégories de fonctions:
  - Fonction nommée;
  - Fonction anonyme;

Fonction nommées

- S'écrit sous la forme:
  function ma_fonction(arg1, arg2){ … }

```
function surfaceCercle()
{
    var entree=prompt("Entrez le rayon du cercle :  ");
    var rayon = parseFloat(entree);

    return 3.14*rayon*rayon;
}
```

Fonctions anonymes

- Fonction anonyme
  function(arg1, arg2) { … }

```
var uneFonction = function(x, y)
                  {
                    return x + y;
                  };
```

```
function(x, y)
{
    return x + y;
};
```

CAUSE UNE ERREUR!

Exemple de fonctions

```
var a=3;
var b=-2;

function multiplie(x)
{
        return 3*x;
}

function affiche()
{
        alert(multiplie(a));
        alert(multiplie(b));
}
```

Exemples de fonctions

```
function times(n, action){
  for(var i = 0; i < n; i++)
  {
    action(i);
  }
 }

times(10, function(iteration)
          {
              console.log("Hello world!"  + iteration);
          });
```

```
var array = ['a','b','c'];

array.forEach(function(element)
              {
                  alert(element);
              });
```

Portée d'une variable

- La portée d'une variable désigne l'ensemble du code dans lequel elle peut être utilisée.
- Si une variable est déclarée sans le mot-clef **var**, elle peut être utilisée n'importe où dans le script. On l'appelle alors **variable globale**.
- Si une variable est déclarée avec le mot-clef **var**, elle ne peut être utilisée que dans le bloc où elle se trouve. On l'appelle alors **variable locale**.

Exemple de portée

```
var a = 8 ;

function testFonction()
{
   var pi = 3.14 ;
   (...) ;
}

function testFonction2()
{
   (...) ;
}
```

- La variable **a** peut être utilisée dans les fonctions **testFonction1** et **testFonction2**, mais la variable **pi** ne peut être utilisée que dans la fonction **testFonction1**.