Les fonctions

Définition

- Une fonction est une procédure ou un sous-programme qui contient un ensemble d'instructions. Ces instructions ne seront exécutées que suite à l'appel de la fonction.

  Les avantages d'une fonction, c'est de réduire le code, réutilisation de cette fonction autant de fois qu'on en aura besoin.
- Il existe quatre syntaxes différentes pour créer une fonction :

  - **Déclaration de fonction**
  - **Expression de fonction**
  - **Fonction fléchée**

Déclaration d'une fonction

- Une fonction correspond à un bloc de code dont le but est d'effectuer une tâche précise.
- Syntaxe de création de fonction:

```
function     NomDeFonction (liste des paramètres){
        //liste d'instructions
    }
```

Si la fonction retourne une valeur il faudra écrire:

```
function     NomDeFonction (liste des paramètres){
        //liste d'instructions
        return valeur ;
    }
```

Exemple

- Exemple de fonction qui calcule la somme de deux nombre et affiche le résultat:

```
function Somme (x ,  y)  {
            alert( x+y) ;
}
```

- Exemple de fonction qui calcule la somme de deux nombre et retourne un résultat:

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

Appel d'une fonction

- Une fonction définie ne peut être exécutée que s'elle est appelée (utilisée)
- Syntaxe: si la fonction ne retourne pas de valeur:

```
NomDeFonction (liste d'arguments) ;

Exemple :
Somme (12,20) ;
```

- Syntaxe: si la fonction retourne une valeur:

```
Var R=NomDeFonction (liste d'arguments) ;

Exemple :

Var S=Somme(12,20); alert(S) ;
```

Les fonctions (exemple 1)

```
<!DOCTYPE html>
<html>
    <head>
        <title>Les fonctions en JavaScript</title>
        <meta charset="utf-8">
    </head>
    <body>
        <h1>Les fonctions</h1>
        <script>
            //On crée notre fonction avec "function"
            function multiplication(x, y){
                alert(x*y);
            }

            multiplication(5, 10);
            multiplication(-4, 60);
            multiplication(3.14, 3.14);
        </script>
    </body>
</html>
```

Les fonctions (exemple 2)

```
<!DOCTYPE html>
<html>
    <head>
        <title>Les fonctions en JavaScript</title>
        <meta charset="utf-8">
    </head>
    <body>
        <h1>Les fonctions</h1>
        <script>
            function multiplication(x, y){
                return  x * y ;
            }

            var resultat = multiplication(5, 10);
            resultat += 2
            alert(resultat);
        </script>
    </body>
</html>
```

Les variables globales

- Une variable globale est une variable déclarée en dehors de la fonction et accessible à l'intérieur de la fonction et peut être modifiée.

```
<script type="text/javascript">

var x=20;  // x est une variable globale
function fct(){

   x=x+5;   //x est accessible à l'intérieur de la fonction
   alert(x);
}


fct();

</script>
```

Les variables locales

- Une variable locale est une variable déclarée à l'intérieur d'une fonction, et est inaccessible en dehors de la fonction:

```
<script type="text/javascript">

function fct(){

  var x=20; //variable locale
  alert(x);

}

fct();//à l'exécution la valeur 20 est affichée
alert(x);  // rien n 'est affiché car x est inaccessible en dehors de la focntion

</script>
```

Le mot clé **var** et **let** pour la déclaration des variables

- Ces deux mots clés sont utilisés pour déclarer les variables.
- 1 ère différence : avec var on peut déclarer deux variables de même nom. (le premier sera écrasé)

```
var x="Salim";
var x="Karim";
alert(x);   //affiche Karim
```

- Avec let, on a pas le droit de déclarer deux variables portants le même nom.

```
let y="Salim";
let y="Karim" ;
alert(y);  //erreur : Previously declared (deja déclaré)
```

Le mot clé **var** et **let** pour la déclaration des variables

- L'utilisation du mot let peut se faire juste à l'intérieur du bloc où a été déclaré.

```
function fct1(){
    let x=12;
    if(true){
        let x=5;
        alert(x);     //affcihe 5
    }
    alert(x);    //affiche 12
}

fct1();
```

```
function fct2(){
    var x=12;
    if(true){
        var x=5;
        alert(x);  //affcihe 5
    }
    alert(x);    //affiche 5
}

fct2();
```

Variable sans déclaration

- Une variable qui n'est pas déclarée (sans le mot var ou let) sont **toujours globales.**

```
function exemple(){
            d=12;
}

exemple();
alert(d);   //affiche bien la valeur 12
```

Expression de fonctions

- Pour créer une expression de fonction, on va **assigner notre fonction à une variable dont on choisira le nom.**

```
let MaFonct = function() {
    alert('Bonjour !');
};
```

Vous remarquez qu'on a utilisé une **fonction anonyme** qu'on assigne ensuite à une **variable**. Pour appeler cette fonction créée comme cela, on va pouvoir utiliser la variable comme une fonction,

```
MaFonct() ;
```

Les fonctions fléchées(ou fonctions Lambda)

- Les fonction fléchées sont des fonctions qui possèdent une syntaxe très compacte, ce qui les rend très rapides à écrire. Ces fonctions utilisent le signe => d'où vient la nomination
- ***Syntaxe :***

```
let nomVariable = (param1 , pram2,...) => Valeur à retourner
```

- Remarquez qu'il n'y a pas **d'accolades** ni le mot **return**
- ***Exemple*** :

```
let somme=(x,y)=>x+y

let res=somme(10,15)
```

Les fonctions fléchées (ou fonctions Lambda)

- ***Exemple2***: Si on a pas de paramètre on laisse les parenthèses vides.

```
let x=()=>alert("hello")
x()
```

- ***Exemple 3:*** Lorsque on a un seul paramètre, on peut enlever les ()

```
let cube=x=>x**3
alert(cube(2))
```

- ***Exemple4***: on peut utiliser le mot return sans problème,

```
let cube=(x)=>{return x**3}

alert(cube(5))
```

setTimeOut

- Le fonction **setTimeout** définit une action à exécuter et un délai avant son exécution. Elle retourne un identificateur pour le processus.

```
var x = setTimeout(fonction, délai en millisecondes)
```

- Exemple :

```
setTimeout(function(){alert("hello")},  5000);
```

- La fonction sera exécutée après **5 secondes** (5 000 millisecondes).

ClearTimeout

- Cette méthode interrompe l'exécution du code associé à ce délai. Le processus à supprimer est reconnu par l'identificateur retourné par *setTimeout*.

```
clearTimeout (identificateur);
```

- Exemple :

```
var  x = setTimeout(mafonction , 5000);
…
 clearTimeout(x);
```

SetInterval

- Similaire à ***setTimeout***, elle déclenche répétitivement la même action à intervalles réguliers.

```
setInterval(fonction, délai)
```

- ***Exemple***:

```
setInterval(function(){alert("hello")}, 10000);
```

- Ce code permet d'afficher un message **toutes les dix secondes.**

clearInterval

- Stoppe le processus déclenché par setInterval.

```
var x = setInterval(mafonction, 10000);
 ...
clearInterval(x);
```