# Javascript

## partie 4

---

## Fonctions - suite

**Révision de la synthase :**

### Fonction

```
function maFonction(monObjet)
 {
   monObjet.marque = "Toyota";
 }

maFonction(monArgument);
```

Référence

Appel de fonction

- Instruction pour déclarer la fonction
- Nom de la fonction
- Paramètres de la fonction
- Arguments dans l'appel (entre les parenthèses)

### Objet

```
let maVariable
 {
   monObjet.marque = "Toyota";
 }
```

Propriété

- Instruction pour déclarer la variable
- Nom de la variable
- Clé
- Valeur

## 1) Créer une fonction

Il existe quatre syntaxes différentes nous permettant de créer une fonction en JavaScript. On va ainsi pouvoir créer une fonction en utilisant :

une déclaration de fonction
une expression de fonction
une fonction fléchée
la syntaxe new Function.

Nous allons nous approfondir sur ces 4 façons. Voici un récapitulatif rapide

| déclaration de fonction | expression de fonction |
|---|---|
| `function somme(a, b){`<br>`  return a + b ;`<br>`}`<br>`somme(1,2);` | `let somme= function(a, b){`<br>`  return a + b ;`<br>`};`<br>`somme(1,2);` |
| **fonction fléchée** | **new Function** |
| `let somme = (a, b) => a + b;`<br>`somme(1,2);` | `let somme= new Function(a, b){`<br>`  return a + b ;`<br>`};`<br>`somme(1,2);` |

## Déclaration de fonction

Nous connaissons bien cette syntaxe : ici, on déclare une fonction avec le mot clef **function**, on le donne un nom et on fait suivre ce nom par un couple de parenthèses (pour les paramètres) et un couple d'accolades (pour les instructions).

```
function somme(a, b){
     return a + b ;
}
somme(1,2);
```

**Ensuite, on l'appelle**

## Expression de fonction

On va directement assigner notre fonction **à une variable dont on choisira le nom**.

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

Généralement, lorsqu'on crée une fonction de cette manière, on utilise une **fonction anonyme** **qu'on assigne ensuite à une variable.** Pour appeler une fonction créée comme cela, on va pouvoir utiliser la variable comme une fonction avec un couple de parenthèses après son nom.

Cependant, rien ne nous empêche de donner un nom à notre fonction. Dans ce cas-là, on parlera de « NFE » pour « **Named Function Expression** » ou « **expression de fonction nommée** » en français.

**Exemple**

```
var disBonjour= function bonjour(nom){
    if (nom){
     console.log('Bonjour '+ nom);
    }else{
        bonjour('inconnu');
    }
};


disBonjour('Pierre');
disBonjour();
```

```
» ▸ var disBonjour= function bonjour(nom){
        if (nom){
            console.log('Bonjour ' + nom);
        }else{
            bonjour('inconnu');…
Bonjour Pierre
Bonjour inconnu
← undefined
```

Le nom de la fonction sera alors local au corps de la fonction (portée). Dans notre exemple, **nous ne pouvons pas utiliser « bonjour » pour appeler la fonction** (mais toujours le nom de la variable qui la stocke suivie de parenthèses)

```
var disBonjour= function
bonjour(nom){
    if (nom){
        console.log('Bonjour '
        + nom);
    }else{
        bonjour('inconnu');
    }
};


bonjour('Pierre');
```

```
» ▸ var disBonjour= function bonjour(nom){
        if (nom){
            console.log('Bonjour ' + nom);
        }else{
            bonjour('inconnu');…
● ▸ Uncaught ReferenceError: bonjour is not
  defined
      <anonymous> debugger eval code:10
    [En savoir plus]
```

## Expression de fonction et IIFE (Immediately Invoked Function Expression)

On peut utiliser une expression de fonction pour créer une « IIFE », c'est-à-dire une expression de fonction qu'on appelle dès sa définition.

```
let a = "coucou";
let b = "monde";

(function(x, y) {
  console.log(x + " " + y);
})(a, b);
```

1) On déclare la/les variable(s) et on assigne une valeur
2) On crée la fonction et on déclare ses paramètres
3) on définie l'instruction
4) on déclare les arguments tout du suite entre parenthèses

```
» ▸ let a = "coucou";
    let b = "monde";

    // IIFE
    (function(x, y) {…
  coucou monde
← undefined
```

Si la fonction est assignée à une variable elle devient une **expression de fonction qui est immédiatement exécutée** (sans avoir besoin d'être appelée par le nom de la variable suivie par les ( );). Si elle est anonyme, on l'appelle f**onction anonyme auto-exécutable.**

**Syntaxe d'une IIFE :**

```
(function () {
            // instruction
             } ) ();
```

**Exemple d'IIFE anonyme** connue sous le nom de f**onction anonyme auto-exécutable**

```
(function () {
    let create = "Coucou";
    console.log(create);
             } ) ();
```

```
» (function () {
            let create = "Coucou";
            console.log(create);
             } ) ();
Coucou
← undefined
```

**ou assignée à une variable:**

```
var toto = (function (a, b) {
    let somme = a + b;
    console.log(somme);
             } ) (1,2);
```

```
» var toto = (function (a, b) {
            let somme = a + b;
            console.log(somme);
             } ) (1,2);
3
```

## Fonction fléchée

Les fonctions fléchées sont des fonctions qui possèdent une syntaxe très compacte, ce qui les rend très rapide à écrire. Les fonctions fléchées utilisent le signe `=>` qui leur a donné leur nom à cause de sa forme de flèche.

**Expression de fonction classique versus fonction fléchée :**

Expression de fonction classique :

```
let addition = function(a, b) {
    return a + b;
};
```

Equivalent en fonction fléchée :

```
let addition = (a, b) => a + b;
console.log(addition(1, 2));
```

```
» let addition = (a, b) => a + b;
  console.log(somme(1, 2));
3
← undefined
```

⚠️ Les fonctions fléchées n'ont pas forcement besoin du couple d'accolades pour fonctionner ou d'une expression return puisque celles-ci vont automatiquement évaluer l'instruction à droite du signe => et retourner son résultat.

**Il existe plusieurs synthases pour une fonction fléchée (en mode déclaration ou expression de fonction, avec ou sans IIFE) :**

| Synthase | Exemples |
|---|---|
| Fonction (expression de fonction, IIFE..., fonction anonyme...) avec plusieurs instructions :<br><br>`let maVariable = (param1, param2, paramX) => {`<br>`  instruction ;`<br>`  return instruction;`<br>`}`<br><br>`maVariable(argument1, argument2, argumentX) ;` | `let somme = (a, b, c) => {`<br>`  let maVariable = a + b + c;`<br>`  return maVariable`<br>`}`<br><br>`somme(1, 2, 3);`<br><br>`» let somme = (a, b, c) => {`<br>`  let maVariable = a + b + c;`<br>`  return maVariable`<br>`}`<br>`…`<br>`← 6` |
| Dans une fonction (expression de fonction, IIFE..., fonction anonyme...) avec une seule instruction (on peut s'en passer des accolades):<br><br>//Ex : expression de fonction :<br>`let maVariable = (param1, param2, …, paramX) => instruction;`<br><br>`maVariable(argument1, argument2, argumentX) ;`<br><br>ou<br>//Ex : IIFE - fonction anonyme auto-exécutable :<br>`((param1, param2,paramX) => instruction)(argument1, argument2, argumentX) ;` | `let maths = (x, y) => x * 2 + y;`<br>`maths(3,2) ;`<br><br>`» let maths = (x, y) => x * 2 + y;`<br><br>`maths(3,2) ;`<br>`← 8`<br><br>ou<br><br>`((a, b, c) => {`<br>`  return a + b + c;`<br>`})(1,2,3)`<br><br>`» ((a, b, c) => {`<br>`  return a + b + c;`<br>`})(1,2,3)`<br>`← 6` |

Dans une fonction (expression de fonction, IIFE..., fonction anonyme...) avec un seul paramètre, on peut s'en passer des parenthèses

//Ex : expression de fonction :

```
let maVariable = param =>
instruction;
maVariable(argument) ;
```

ou

//Ex : IIFE anonyme :

```
(param => {instruction;})
(argument) ;
```

```
let disHello = monParam =>
console.log(monParam);
disHello("Hello") ;
```

```
» let disHello = monParam => console.log(monParam);
  disHello("Hello") ;
  Hello
← undefined
```

ou

```
(a=>{console.log(a) ;})("Hello
world")
```

```
» (a=>{console.log(a) ;})("Hello world")
  Hello world
← undefined
```

Une fonction (expression de fonction, IIFE..., fonction anonyme...) sans paramètre peut s'écrire avec un couple de parenthèses vide

//Ex : expression de fonction :

```
let maVariable = () => {
  instructions
}
maVariable() ;
```

ou

// expression de fonction IIFE:

```
let maVariable = (() => {
  instructions
})() ;
```

ou

//Ex : IIFE anonyme :

```
(() => {
  instructions
})() ;
```

```
let x = 9 ;
var fonction = () => x * x;
fonction() ;
```

```
» let x = 9 ;
  var fonction = () => x * x;
  fonction()
← 81
```

ou

```
let coucou = (() =>
{console.log("Hello")})();
```

```
» let coucou = (() => {console.log("Hello")})();
  Hello
← undefined
```

ou

```
(() => {console.log("Hello")})();
```

```
» (() => {console.log("Hello")})();
  Hello
← undefined
```