Cours : Les Fonctions en JavaScript

1. Qu'est-ce qu'une fonction ?

Une **fonction** est un bloc de code que l'on écrit une fois et qu'on peut **réutiliser** plusieurs fois dans un programme.

Elle sert à **exécuter une tâche précise** : par exemple, afficher un message, faire un calcul, ou modifier une page web.

2. Pourquoi utiliser des fonctions ?

- Éviter de répéter le même code plusieurs fois.
- Structurer le programme en parties claires et réutilisables.
- Faciliter la lecture et la maintenance du code.
- Pouvoir effectuer des calculs ou actions personnalisées à partir de valeurs données.

3. Les fonctions nommées

3.1. Définition

Une **fonction nommée** est une fonction que l'on déclare avec le mot-clé `function`, suivie d'un **nom** et d'un bloc d'instructions.

3.2. Syntaxe

```
function nomDeLaFonction() {
  // instructions à exécuter
}
```

3.3. Exemple simple

```
function direBonjour() {
  console.log("Bonjour !");
}

// Appel de la fonction
direBonjour();
```

Explication :

- `function` indique qu'on crée une fonction.
- `direBonjour` est le nom de la fonction.
- Les instructions entre `{}` sont exécutées quand on l'appelle avec `direBonjour()`.

3.4. Fonctions avec paramètres

Une fonction peut recevoir des **valeurs d'entrée** appelées **paramètres**.

```
function saluer(prenom) {
  console.log("Bonjour " + prenom);
}

saluer("Awa");
saluer("Ali");
```

Résultat :

```
Bonjour Awa
Bonjour Ali
```

Explication :

- `prenom` est un paramètre (une sorte de variable temporaire).
- Lors de l'appel, `"Awa"` est l'argument transmis à la fonction.

3.5. Fonction avec plusieurs paramètres

```
function addition(a, b) {
  console.log(a + b);
}

addition(4, 6); // Affiche 10
addition(10, 25); // Affiche 35
```

3.6. Fonction avec retour de valeur (`return`)

Une fonction peut **renvoyer une valeur** à l'aide du mot-clé `return`.

```
function carre(nombre) {
  return nombre * nombre;
}

let resultat = carre(5);
console.log(resultat); // 25
```

Explication :

- `return` envoie le résultat au programme.
- La variable `resultat` récupère cette valeur.

---

4. Les fonctions anonymes

4.1. Définition

Une **fonction anonyme** est une fonction **sans nom**, souvent utilisée dans des variables ou des événements.

4.2. Exemple simple

```
let direSalut = function() {
  console.log("Salut !");
};

direSalut();
```

Explication :

- La fonction est stockée dans la variable `direSalut`.
- On l'appelle ensuite comme une fonction normale : `direSalut()`.

4.3. Exemple avec paramètres

```
let multiplier = function(a, b) {
  return a * b;
};

console.log(multiplier(3, 4)); // 12
```

---

5. Les fonctions fléchées (`()=>{}`)

5.1. Définition

Les **fonctions fléchées** (arrow functions) sont une **forme simplifiée** de fonctions anonymes introduite avec **ES6**.

Elles sont plus courtes à écrire, surtout pour des fonctions simples.

5.2. Syntaxe générale

```
const nomFonction = (paramètres) => {
  // instructions
};
```

5.3. Exemple simple

```
const direBonjour = () => {
  console.log("Bonjour !");
};

direBonjour();
```

5.4. Exemple avec paramètres

```
const addition = (a, b) => {
  return a + b;
};

console.log(addition(5, 3)); // 8
```

5.5. Version encore plus courte

Si la fonction contient **une seule instruction return**,

on peut **supprimer les accolades** `{}` et le mot-clé `return`.

```
const carre = x => x * x;
console.log(carre(4)); // 16
```

Remarques :

- Si un seul paramètre, les parenthèses `()` sont facultatives.
- Si plusieurs paramètres, elles sont obligatoires : `(a, b) => a + b`.

6. Différences entre les trois formes

Type de fonction	Syntaxe	Peut avoir un nom ?	Mot-clé `this`	Utilisation typique
Nommée	`function nom(){}`	Oui	Dynamique	Fonctions globales, réutilisables
Anonyme	`let f = function(){}`	Non	Dynamique	Fonctions stockées dans des variables
Fléchée	`const f = ()=>{}`	Non	Hérité (fixe)	Fonctions courtes, callbacks

7. Bonnes pratiques

- Donner un **nom clair** à chaque fonction pour décrire son action.
- Utiliser les **paramètres** au lieu de valeurs fixes dans la fonction.

- Utiliser `return` quand une valeur doit être réutilisée.
- Préférer les **fonctions fléchées** pour des fonctions courtes et anonymes.

---

8. Exemples d’application

Exemple 1 : afficher un message

```
function afficherMessage() {
  alert("Bienvenue dans le cours de JavaScript !");
}

afficherMessage();
```

Exemple 2 : calculer le carré d’un nombre

```
function carre(nombre) {
  return nombre * nombre;
}

console.log(carre(5)); // 25
```

Exemple 3 : fonction anonyme dans une variable

```
let afficherNom = function() {
  console.log("Je m'appelle Aminata");
};

afficherNom();
```

Exemple 4 : fonction fléchée

```
const direBonjour = (nom) => {
  console.log("Bonjour " + nom);
};
```

```
direBonjour("Fatou");
```

Exemple 5 : fonction fléchée très courte

```
const addition = (a, b) => a + b;
console.log(addition(10, 5)); // 15
```

9. Exercices d’application

1. Créer une fonction `bonjour()` qui affiche "Bonjour à tous !" dans la console.
2. Créer une fonction `multiplier(a, b)` qui retourne le produit de deux nombres.
3. Créer une fonction anonyme stockée dans une variable `direNom` qui affiche un prénom.
4. Créer une fonction fléchée `carre` qui retourne le carré d’un nombre.
5. Créer une fonction fléchée `soustraction(a, b)` qui retourne la différence entre deux nombres.