Par Robert DIASSÉ

Les Fonctions en JavaScript

Les fonctions sont des blocs de code réutilisables qui permettent d'organiser et de structurer votre code. Elles prennent des entrées (appelées paramètres), effectuent des opérations et retournent une valeur.

Déclaration de Fonction

Fonction Nommée

Une fonction nommée est définie avec le mot-clé `function` suivi d'un nom, de paramètres entre parenthèses et d'un bloc de code entre accolades.

```
function saluer(nom) {
    return "Bonjour " + nom + "!";
}

console.log(saluer("Alice")); // Affiche "Bonjour Alice!"
```

Fonction Anonyme

Une fonction anonyme est une fonction sans nom, souvent utilisée comme valeur de retour ou argument pour une autre fonction.

```
let afficherMessage = function(message) {
    console.log(message);
};

afficherMessage("Salut!"); // Affiche "Salut!"
```

Fonction Fléchée (Arrow Function)

Les fonctions fléchées offrent une syntaxe plus concise et ne lient pas leur propre `this`.

```
let addition = (a, b) => a + b;

console.log(addition(2, 3)); // Affiche 5
```

Fonctions Imbriquées

Les fonctions peuvent être définies à l'intérieur d'autres fonctions.

```
function exterieur(a) {
    function interieur(b) {
        return a + b;
    }
    return interieur;
}

let additionDeCinq = exterieur(5);
console.log(additionDeCinq(3)); // Affiche 8
```

Fonctions Prédéfinies (Built-in Functions)

JavaScript inclut de nombreuses fonctions prédéfinies qui peuvent être utilisées immédiatement.

`parseInt()`

Convertit une chaîne en entier.

```
let entier = "123";
console.log(typeof entier);// Affiche string
entier = parseInt("123");
console.log(entier); // Affiche 123
console.log(typeof entier); // Affiche number
```

`parseFloat()`

Convertit une chaîne en nombre à virgule flottante.

```
let flottant = parseFloat("123.45");
console.log(flottant); // Affiche 123.45
```

`isNaN()`

Vérifie si une valeur est NaN (Not-a-Number).

```
let resultat = isNaN("123");
console.log(resultat); // Affiche false
```

setTimeout()

Exécute une fonction après un certain délai.

```
setTimeout(function() {
    console.log("Exécuté après 2 secondes");
}, 2000);
```

setInterval()

Exécute une fonction à intervalles réguliers.

```
let compteur = 0;
let intervalId = setInterval(function() {
    compteur++;
    console.log("Compteur : " + compteur);
    if (compteur === 5) {
        clearInterval(intervalId);
    }
}, 1000);
```

Fonctions Avancées

Fonctions de Rappel (Callback Functions)

Les fonctions de rappel sont des fonctions passées en argument à une autre fonction, qui les appelle après avoir terminé son travail.

```
function demanderNom(callback) {
    let nom = prompt("Quel est votre nom?");
    callback(nom);
}

demanderNom(function(nom) {
    console.log("Bonjour " + nom + "!");
});
```

Fonctions Immédiatement Invocables (IIFE)

Les fonctions immédiatement invoquées sont définies et exécutées immédiatement.

```
(function() {
    console.log("Cette fonction est immédiatement invoquée.");
})();
```

Fonctions Asynchrones

Les fonctions asynchrones utilisent `async` et `await` pour gérer les opérations asynchrones de manière plus lisible.

```
async function fetchData() {
    let response = await fetch('https://api.example.com/data');
    let data = await response.json();
    console.log(data);
}

fetchData();
```

Portée des Variables

Les variables en JavaScript peuvent avoir une portée globale ou locale.

- **Globale** : Déclarée en dehors de toute fonction.
- **Locale** : Déclarée à l'intérieur d'une fonction.

```
let globalVar = "Je suis globale";

function afficherGlobale() {
    console.log(globalVar);
}

afficherGlobale(); // Affiche "Je suis globale"
```

Récursivité

Une fonction récursive est une fonction qui s'appelle elle-même.

```
function factorielle(n) {
    if (n <= 1) {
        return 1;
    } else {
        return n * factorielle(n - 1);
    }
}

console.log(factorielle(5)); // Affiche 120
```

Exercices

Exercice 1 : Création de Fonction

1. Créez une fonction `multiplier` qui prend deux nombres en paramètres et retourne leur produit.

Exercice 2 : Fonction de Rappel

1. Créez une fonction `effectuerOperation` qui prend deux nombres et une fonction de rappel pour effectuer une opération (addition, multiplication, etc.).

Exercice 3 : Fonction Fléchée

1. Convertissez la fonction `multiplier` en fonction fléchée.

Exercice 4 : Récursivité

1. Créez une fonction récursive `compter` qui affiche les nombres de 1 à 5.

Spread Operator

Le spread operator (`...`) permet d'étaler les éléments d'un tableau ou les propriétés d'un objet. Il est particulièrement utile dans les fonctions pour accepter un nombre variable d'arguments ou pour étendre des objets.

Utilisation avec les Tableaux

```
let nombres = [1, 2, 3];
let plusDeNombres = [...nombres, 4, 5, 6];

console.log(plusDeNombres); // Affiche [1, 2, 3, 4, 5, 6]
```

Utilisation avec les Objets

```
let personne = { nom: "Alice", age: 25 };
let personneAvecVille = { ...personne, ville: "Paris" };

console.log(personneAvecVille); // Affiche { nom: "Alice", age: 25, ville: "Paris"
}
```

Utilisation dans les Fonctions

Le spread operator peut être utilisé pour passer un nombre variable d'arguments à une fonction.

```
function addition(...nombres) {
    return nombres.reduce((acc, curr) => acc + curr, 0);
}

console.log(addition(1, 2, 3, 4)); // Affiche 10
```

Exercice 5 : Utilisation du Spread Operator

1. Créez une fonction `concatenerTableaux` qui utilise le spread operator pour fusionner deux tableaux.
2. Créez une fonction `fusionnerObjets` qui utilise le spread operator pour fusionner deux objets.

Ce cours couvre les bases des fonctions en JavaScript, ainsi que des concepts plus avancés, incluant la documentation pour les types de retour, le spread operator, et des exemples concrets pour une meilleure compréhension. Les exercices fournis permettent de mettre en pratique les concepts appris et de consolider les connaissances.