



JavaScript :

Les fonctions en JavaScript

Introduction

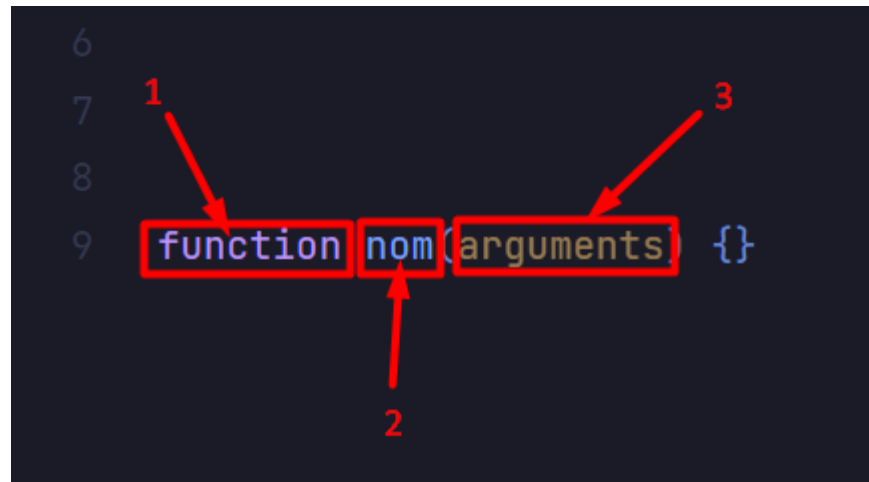
Dans ce chapitre, nous allons explorer les fonctions en JavaScript.

- Les fonctions sont des morceaux de code qui peuvent être réutilisés plusieurs fois pour effectuer une tâche spécifique.
- Cela permet d'organiser le code et de faciliter la maintenance et l'évolution d'un programme.

Déclaration de fonctions

En JavaScript, une fonction peut être déclarée à l'aide :

1. du mot-clé **function**
2. suivi d'un **nom** de fonction
3. et d'une **liste d'arguments**.



```
6  
7  
8  
9 function nom(arguments) {}
```

The diagram shows a JavaScript function declaration on a dark background. The code is `function nom(arguments) {}`. Three red arrows with numbers point to specific parts: arrow 1 points to the keyword `function`, arrow 2 points to the function name `nom`, and arrow 3 points to the opening curly brace of the argument list `(arguments)`. Each of these three components (`function`, `nom`, and `(arguments)`) is enclosed in a red rectangular box. The closing curly brace `}` is not boxed.

Les instructions à exécuter lorsque la fonction est appelée sont définies dans un bloc de code entre accolades :

```
7  
8  
9  function nom(arguments) {  
10     // bloc de code  
11 }
```

Les fonctions en JavaScript sont des blocs de code qui peuvent être réutilisés plusieurs fois dans un programme.

Elles permettent de regrouper des instructions en un seul endroit, ce qui facilite la lecture et la maintenance du code.

Les fonctions peuvent également prendre des paramètres en entrée et renvoyer une valeur en sortie.

Pour déclarer une fonction en JavaScript, vous utilisez le mot-clé "function", suivi du nom de la fonction, puis d'une paire de parenthèses qui peuvent contenir des paramètres. Les instructions de la fonction sont ensuite encadrées par des accolades.

Voici un exemple de déclaration de fonction :

```
function direBonjour(nom) {  
    console.log("Bonjour, " + nom);  
}
```

JavaScript Copy

Pour utiliser une fonction, vous devez l'appeler en utilisant son nom suivi d'une paire de parenthèses.

Les valeurs passées entre les parenthèses sont appelées des arguments, et ils sont utilisés pour fournir des informations à la fonction.

```
direBonjour("John");
```

JavaScript Copy

Dans cet exemple, la fonction "direBonjour" est appelée avec l'argument "John" et affiche "Bonjour John" dans la console.

Les fonctions avec valeur de retour

Il est possible de définir des fonctions qui retournent une valeur en utilisant l'instruction **"return"** suivie de la valeur à retourner.

```
function addition(a, b){  
    return a + b;  
}  
let result = addition(2, 3);  
console.log(result); // affiche 5
```

JavaScript

Copy

Fonctions anonymes

Il est également possible de créer des fonctions anonymes, qui ne sont pas associées à un nom.

Ces fonctions peuvent être assignées à une variable ou passées en tant qu'argument à une autre fonction.

```
let direBonjour = function(nom) {  
  console.log("Bonjour, " + nom);  
};  
direBonjour("Paul"); //
```

JavaScript Copy

Fonctions fléchées

Depuis ECMAScript 6 (ES2015), il est possible de créer des fonctions fléchées, qui sont une syntaxe plus courte pour les fonctions anonymes.

- ✓ Les fonctions fléchées (également appelées "arrow functions") sont une syntaxe alternative pour déclarer des fonctions en JavaScript.
- ✓ Elles permettent de créer des fonctions plus concises et lisibles.

La syntaxe d'une fonction fléchée est composée d'une liste de paramètres entre parenthèses, suivie d'une flèche "=>" et d'une expression représentant le corps de la fonction.

Voici un exemple de déclaration de fonction fléchée :

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

JavaScript

Copy

Si la fonction n'a qu'une seule instruction, les accolades peuvent être omises et l'instruction elle-même peut être écrite après la flèche:

```
let direBonjour = nom => console.log("Bonjour " + nom);
```

JavaScript

Copy

Si une fonction n'a qu'un seul paramètre, les parenthèses peuvent être omises:

```
let direBonjour = nom => console.log("Bonjour " + nom);
```

JavaScript

Copy

Si une fonction n'a pas de paramètre :

```
let direBonjour = () => console.log("Bonjour !");
```

On peut convertir une fonction normale en fonction fléchée en remplaçant le mot-clé "function" par une déclaration de fonction fléchée.

Exemple de transformation de fonction normale en fonction fléchée :

```
// fonction normale
function addition(a, b){
  return a + b;
}
// transformation en fonction fléchée
let addition = (a, b) => a + b;
```

JavaScript Copy

Il est important de noter que les fonctions fléchées n'ont pas de mot-clé "this" lié, elles utilisent celui de leur contexte d'appel.

Conclusion

En résumé:

En résumé, les fonctions sont un moyen de réutiliser du code et de faciliter la maintenance et l'évolution d'un programme. JavaScript prend en charge plusieurs manières de déclarer des fonctions, telles que les fonctions classiques, les fonctions anonymes et les fonctions fléchées. Il est important de comprendre comment utiliser les fonctions pour organiser le code et améliorer l'efficacité d'un programme.

Exercices

Exercice 1 :

1. Créez une fonction normale qui prend un nombre en entrée et renvoie son carré
2. Transformez cette fonction en une fonction fléchée

Exercice 2 :

1. Créez une fonction normale qui prend deux nombres en entrée et renvoie leur somme
2. Transformez cette fonction en une fonction fléchée

Exercice 3 :

1. Créez une fonction normale qui prend un tableau en entrée et renvoie la somme de tous ses éléments
2. Transformez cette fonction en une fonction fléchée

Exercice 4 :

1. Créez une fonction normale qui prend un objet en entrée et renvoie une chaîne de caractères contenant ses propriétés et leurs valeurs
2. Transformez cette fonction en une fonction fléchée

Exercice 5 :

1. Créez une fonction normale qui prend un paramètre callback et l'appelle avec un argument de votre choix
2. Transformez cette fonction en une fonction fléchée

Corrections

Exercice 1 : correction :

```
// Fonction normale
function carre(nombre){
    return nombre * nombre;
}
console.log(carre(5)); // 25

// Fonction fléchée
let carre = nombre => nombre * nombre;
console.log(carre(5)); // 25
```

JavaScript

Copy

Exercice 2 : correction :

```
// Fonction normale
function somme(a, b){
    return a + b;
}
console.log(somme(2,3)); // 5

// Fonction fléchée
let somme = (a, b) => a + b;
console.log(somme(2,3)); // 5
```

JavaScript Copy

Exercice 3 : correction :

```
// Fonction normale
function sommeTableau(tableau){
  let result = 0;
  for(let i = 0; i < tableau.length; i++){
    result += tableau[i];
  }
  return result;
}
console.log(sommeTableau([1, 2, 3, 4])); // 10

// Fonction fléchée
let sommeTableau = tableau => tableau.reduce((a, b) => a + b);
console.log(sommeTableau([1, 2, 3, 4])); // 10
```

JavaScript Copy

Exercice 4 : correction :

```
// Fonction normale
function afficherObjet(objet){
  let result = "";
  for(let propriete in objet){
    result += propriete + ": " + objet[propriete] + "\n";
  }
  return result;
}
console.log(afficherObjet({nom: "John", age: 25}));
// nom: John
// age: 25

// Fonction fléchée
let afficherObjet = objet => Object.entries(objet).map(([cle, valeur]) => `${cle}: ${valeur}`).join("\n");
console.log(afficherObjet({nom: "John", age: 25}));
// nom: John
// age: 25
```

JavaScript

Copy

Exercice 5 : correction :

```
// Fonction normale
function appelCallback(callback){
    callback("argument");
}
appelCallback(arg => console.log(arg)); // "argument"

// Fonction fléchée
let appelCallback = callback => callback("argument");
appelCallback(arg => console.log(arg)); // "argument"
```

JavaScript

Copy

Références

Mozilla Developer Network, JavaScript Guide, <https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Functions>

W3Schools, JavaScript Functions, https://www.w3schools.com/js/js_functions.asp

ECMAScript 6, Arrow function, https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Functions/Arrow_functions

Questions ??

Prochain chapitre: Les objets en JavaScript.