codecademy

Les fonctions

Fonctions fléchées (ES6)

Les expressions de fonction de flèche ont été introduites dans ES6. Ces expressions sont claires et concises. La syntaxe d'une expression de fonction fléchée ne nécessite pas le `function` mot-clé et utilise une grosse flèche `=>` pour séparer le(s) paramètre(s) du corps.
Il existe plusieurs variantes de fonctions fléchées :

- Les fonctions fléchées avec un seul paramètre ne nécessitent pas de `()` autour de la liste des paramètres.
- Les fonctions fléchées avec une seule expression peuvent utiliser le corps de fonction concis qui renvoie le résultat de l'expression sans le mot-`return` clé.

```
// Arrow function with two arguments
const sum = (firstParam, secondParam) =>
{
  return firstParam + secondParam;
};
console.log(sum(2,5)); // Prints: 7

// Arrow function with no arguments
const printHello = () => {
  console.log('hello');
};
printHello(); // Prints: hello

// Arrow functions with a single argument
const checkWeight = weight => {
  console.log(`Baggage weight : ${weight}
kilograms.`);
};
checkWeight(25); // Prints: Baggage
weight : 25 kilograms.


// Concise arrow functions
const multiply = (a, b) => a * b;
console.log(multiply(2, 30)); // Prints:
60
```

Les fonctions

Les fonctions sont l'un des éléments fondamentaux de JavaScript. Une *fonction* est un ensemble réutilisable d'instructions pour effectuer une tâche ou calculer une valeur. Les fonctions peuvent recevoir une ou plusieurs valeurs et peuvent renvoyer une valeur à la fin de leur exécution. Pour utiliser une fonction, vous devez la définir quelque part dans la portée où vous souhaitez l'appeler.
L'exemple de code fourni contient une fonction qui prend 2 valeurs et renvoie la somme de ces nombres.

```
// Defining the function:
function sum(num1, num2) {
  return num1 + num2;
}

// Calling the function:
sum(3, 6); // 9
```

Fonctions anonymes

Les fonctions anonymes en JavaScript n'ont pas de propriété de nom. Ils peuvent être définis à l'aide du mot- `function` clé ou d'une fonction fléchée. Voir l'exemple de code pour la différence entre une fonction nommée et une fonction anonyme.

```
// Named function
function rocketToMars() {
  return 'BOOM!';
}

// Anonymous function
const rocketToMars = function() {
  return 'BOOM!';
}
```

Expressions de fonction

Les expressions de fonction créent des fonctions à l'intérieur d'une expression plutôt qu'en tant que déclaration de fonction. Ils peuvent être anonymes et/ou affectés à une variable.

```
const dog = function() {
  return 'Woof!';
}
```

Paramètres de fonction

Les entrées des fonctions sont appelées *paramètres* lorsqu'une fonction est déclarée ou définie. Les paramètres sont utilisés comme variables dans le corps de la fonction. Lorsque la fonction est appelée, ces paramètres auront la valeur de tout ce qui est *passé* en tant qu'arguments. Il est possible de définir une fonction sans paramètre.

```
// The parameter is name
function sayHello(name) {
  return `Hello, ${name}!`;
}
```

`return`Mot-clé

Les fonctions renvoient (renvoient) des valeurs à l'aide du mot- `return` clé. `return` termine l'exécution de la fonction et renvoie la valeur spécifiée à l'emplacement où elle a été appelée. Une erreur courante consiste à oublier le mot- `return` clé, auquel cas la fonction reviendra `undefined` par défaut.

```
// With return
function sum(num1, num2) {
  return num1 + num2;
}

// Without return, so the function
doesn't output the sum
function sum(num1, num2) {
  num1 + num2;
}
```

Déclaration de fonction

Les déclarations de fonction sont utilisées pour créer des fonctions nommées. Ces fonctions peuvent être appelées en utilisant leur nom déclaré. Les déclarations de fonction sont construites à partir de :

- Le `function` mot clé.
- Le nom de la fonction.
- Une liste facultative de paramètres séparés par des virgules entre parenthèses `()` .
- Un corps de fonction entouré d'accolades `{}` .

```
function add(num1, num2) {
  return num1 + num2;
}
```

Fonctions d'appel

Les fonctions peuvent être *appelées* ou exécutées ailleurs dans le code en utilisant des parenthèses après le nom de la fonction. Lorsqu'une fonction est appelée, le code à l'intérieur de son corps de fonction s'exécute. *Les arguments* sont des valeurs transmises à une fonction lorsqu'elle est appelée.

```
// Defining the function
function sum(num1, num2) {
  return num1 + num2;
}

// Calling the function
sum(2, 4); // 6
```