codecademy

Itérateurs

Fonctions affectées aux variables

En JavaScript, les fonctions sont un type de données tout comme les chaînes, les nombres et les tableaux sont des types de données. Par conséquent, les fonctions peuvent être affectées en tant que valeurs à des variables, mais sont différentes de tous les autres types de données car elles peuvent être appelées.

```
let plusFive = (number) => {
  return number + 5;
};
// f is assigned the value of plusFive
let f = plusFive;

plusFive(3); // 8
// Since f has a function value, it can
be invoked.
f(9); // 14
```

Fonctions de rappel

En JavaScript, une fonction de rappel est une fonction qui est transmise à une autre fonction en tant qu'argument. Cette fonction peut alors être invoquée lors de l'exécution de cette fonction d'ordre supérieur (dont elle est un argument).
Puisque, en JavaScript, les fonctions sont des objets, les fonctions peuvent être passées en arguments.

```
const isEven = (n) => {
  return n % 2 == 0;
}

let printMsg = (evenFunc, num) => {
  const isNumEven = evenFunc(num);
  console.log(`The number ${num} is an
even number: ${isNumEven}.`)
}

// Pass in isEven as the callback
function
printMsg(isEven, 4);
// Prints: The number 4 is an even
number: True.
```

Fonctions d'ordre supérieur

En Javascript, les fonctions peuvent être affectées à des variables de la même manière que les chaînes ou les tableaux. Ils peuvent être transmis à d'autres fonctions en tant que paramètres ou renvoyés par celles-ci également.
Une « fonction d'ordre supérieur » est une fonction qui accepte des fonctions comme paramètres et/ou renvoie une fonction.

Fonctions JavaScript : objets de première classe

Les fonctions JavaScript sont des objets de première classe. Donc:

- Ils ont des propriétés et des méthodes intégrées, telles que la `name` propriété et la `.toString()` méthode.
- Des propriétés et des méthodes peuvent leur être ajoutées.
- Ils peuvent être passés comme arguments et renvoyés par d'autres fonctions.
- Ils peuvent être affectés à des variables, des éléments de tableau et d'autres objets.

```
//Assign a function to a variable originalFunc
const originalFunc = (num) => { return num + 2 };

//Re-assign the function to a new variable newFunc
const newFunc = originalFunc;

//Access the function's name property
newFunc.name; //'originalFunc'

//Return the function's body as a string
newFunc.toString(); //'(num) => { return num + 2 }'

//Add our own isMathFunction property to the function
newFunc.isMathFunction = true;

//Pass the function as an argument
const functionNameLength = (func) => { return func.name.length };
functionNameLength(originalFunc); //12

//Return the function
const returnFunc = () => { return newFunc };
returnFunc(); //[Function: originalFunc]
```

Méthode de tableau `.reduce()`

La `.reduce()` méthode parcourt un tableau et renvoie une seule valeur.
Il prend une fonction de rappel avec deux paramètres `(accumulator, currentValue)` comme arguments. A chaque itération, `accumulator` est la valeur renvoyée par la dernière itération, et `currentValue` est l'élément courant. Facultativement, un deuxième argument peut être passé qui agit comme la valeur initiale de l'accumulateur.
Ici, la `.reduce()` méthode additionnera tous les éléments du tableau.

```
const arrayOfNumbers = [1, 2, 3, 4];

const sum
= arrayOfNumbers.reduce((accumulator, currentValue) => {
  return accumulator + currentValue;
});

console.log(sum); // 10
```

Méthode de tableau `.forEach()`

La `.forEach()` méthode exécute une fonction de rappel sur chacun des éléments d'un tableau dans l'ordre.
Ici, la fonction de rappel contenant une `console.log()` méthode sera exécutée plusieurs `5` fois, une fois pour chaque élément.

```
const numbers = [28, 77, 45, 99, 27];

numbers.forEach(number => {
  console.log(number);
});
```

Méthode de tableau `.filter()`

La `.filter()` méthode exécute une fonction de rappel sur chaque élément d'un tableau. La fonction de rappel pour chacun des éléments doit renvoyer soit `true` ou `false` . Le tableau renvoyé est un nouveau tableau contenant tous les éléments pour lesquels la fonction de rappel renvoie `true` .
Ici, le tableau `filteredArray` contiendra tous les éléments de `randomNumbers` but `4` .

```
const randomNumbers = [4, 11, 42, 14,
39];
const filteredArray
= randomNumbers.filter(n => {
  return n > 5;
});
```

Méthode de tableau `.map()`

La `.map()` méthode exécute une fonction de rappel sur chaque élément d'un tableau. Elle renvoie un nouveau tableau composé des valeurs de retour de la fonction de rappel.
Le tableau d'origine n'est pas modifié et le tableau renvoyé peut contenir des éléments différents du tableau d'origine.

```
const finalParticipants = ['Taylor',
'Donald', 'Don', 'Natasha', 'Bobby'];

const announcements
= finalParticipants.map(member => {
  return member + ' joined the contest.';
})

console.log(announcements);
```