Les fonctions

Description Sommaire

À propos de ce tutoriel

Ecrire le code "comme il arrive" est suffisant pour un algorithme simple mais on va être très rapidement amené à répéter une même logique plusieurs fois. Pour remédier à ce problème, il est possible d'écrire des **fonctions**.

Syntaxe

Les fonctions permettent de stocker en mémoire une certaine logique que l'on pourra utiliser à plusieurs reprises dans la suite de notre code. Elles prennent en général des paramètres et retournent un résultat particulier.

```
function saluer (nom) {
    return "Salut " + nom
}
// On appelle ensuite notre fonction avec
saluer('Marc') // Salut Marc
```

Les fonctions en JavaScript sont un type de variable particulier, il est donc aussi possible de ne pas leur donner de nom, mais de les stocker dans une variable de manière classique.

```
const saluer = function (nom) {
    return "Salut " + nom
}
```

- Si la fonction ne **return** rien à la fin de son exécution alors elle renverra automatiquement la valeur **undefined**.

- Si on appelle la fonction en omettant certains paramètres, ils prendront la valeur **undefined**.

```
const demo = function (nom) {
    return "Salut " + nom
}
demo() // nom aura la valeur undefined dans la fonction
```

On peut préciser une valeur par défaut à un paramètre en utilisant une assignation

```
const demo = function (nom = 'John') {
    return "Salut " + nom
}
demo() // nom aura la valeur "John" par défaut
```

Portée des fonctions

Une fonction déclarée dans une variable aura la même portée que les variables (limité au bloc courant). Par contre une fonction déclarée directement avec le mot clef `function` aura une portée plus globale :

```
if (true) {
    function a() { console.log('a') }
}
a() // 'a'
```

Aussi, l'hoisting fera qu'une fonction peut être appelée avant d'être déclarée

```
a() // 'a'
function a() { console.log('a') }
```

La portée est limitée si la fonction est déclarée dans une autre fonction

```
function a () {
    console.log('a')
    function b () { console.log('b') }
}

a()
b() // b is not defined
```

this

L'opérateur `this` est un opérateur particulier que l'on peut utiliser dans une fonction. La valeur de cet opérateur dépendra de la façon dont une fonction est appelée.

```
function maFonction () {
    console.log(this)
}

maFonction.call(3) // this aura la valeur 3
maFonction.call(4) // thiis aura la valeur 4
```

Si une fonction est déclarée sur un objet alors this aura la valeur de l'objet.

```
const a = {
  prop: 42,
  func: function() {
    return this.prop;
  },
};

console.log(a.func()) // '42'
```

Mais on peut tout de même changer si on appelle la fonction avec `call()` ou `apply()`.

Les fonctions fléchées

Les fonctions fléchées sont une syntaxe alternative (plus courte) pour les fonctions.

```
const hello = (name) => {
    console.log(`Bonjour ${name}`)
}
```

Ces fonctions ont comme particularité de ne pas posséder de valeur `this`

```
const hello = () => {
    console.log(this)
}
hello() // this sera l'objet global (window dans le cas du navigateur)
hello.call(3) // this sera toujours l'objet global
```

Aussi, si il n'y a qu'une instruction de retour, on pourra simplifier l'appel en retirant les accolades. On pourra aussi retirer les parenthèses si il n'y a qu'un paramètre.

```
const double = (n) => {
    return 2 * n
}
// On peut simplifier en retournant directement en retirant les accolades
const double = (n) => 2 * n
// Et on peut retirer les parenthèses
const double = n => 2 * n
```

Exercices

Deviner le nombre

24:20 - On crée un nombre aléatoire entre 0 et 10, ensuite on demandera à l'utilisateur de deviner ce nombre avec 3 essais. Pour créer la logique on utilisera des fonctions spécifiques

- Une fonction `isRight(n)` qui renverra un booléen si l'utilisateur à la bonne réponse ou non
- Une fonction `guess()` qui permet de faire un essai, cette fonction renverra true ou false en fonction de la réponse donnée

Voici le début du code

```
const solution = Math.floor(Math.random() * 11)
```

Voir la réponse

Les nombres premiers

31:17 - Créer une fonction `isPremier()` qui permet d'indiquer si un nombre est premier

```
console.log('0', isPremier(0)) // false
console.log('1', isPremier(1)) // false
console.log('2', isPremier(2)) // true
console.log('3', isPremier(3)) // true
console.log('11', isPremier(11)) // true
console.log('12', isPremier(12)) // false
```

Voir la réponse