Introduction à la programmation fonctionnelle

Qu'est-ce que la programmation fonctionnelle ?

La programmation fonctionnelle est un paradigme de programmation qui consiste à concevoir ses programmes comme un ensemble de fonctions mathématiques que l'on compose entre elles.

Les deux éléments essentiels dans la programmation fonctionnelle sont **l'utilisation de fonctions de première classe** et **l'utilisation de fonctions pures**.

Les fonctions sont des objets de première classe en `JavaScript`, **elles peuvent donc être passées en argument et donc être composées**. C'est notamment le principe des fonctions de rappel.

Les fonctions pures sont des fonctions dont le résultat ne dépend que des arguments passés et qui n'ont pas d'effets extérieurs (appelés effets de bord).

L'une des principales conséquence de privilégier la programmation fonctionnelle est que **les fonctions sont indépendantes de leur contexte d'exécution**. C'est-à-dire que pour les mêmes arguments, elles retourneront toujours le même résultat, et ce où qu'elles soient appelées. Cela permet une grande maintenabilité.

Nous avons déjà utilisé de la programmation fonctionnelle et allons voir dans cette leçon des fonctions pures extrêmement puissantes permettant une composition par chaînage.

La fonction `map()`

La méthode `map()` crée et retourne un nouveau tableau avec les résultats de l'appel d'une fonction de rappel utilisée sur chaque élément du tableau.

`map` ne modifie pas le tableau sur lequel elle est utilisée, il s'agit d'une fonction pure.

```
const tableau = [1, 2, 3];
const tableau2 = tableau.map(el => el ** el);
console.log(tableau2); // [1, 4, 27]
```

Nous passons une fonction de rappel qui va être exécutée pour tous les éléments du tableau.

Ici nous élevons chaque nombre à la puissance du nombre.

Le premier argument reçu par la fonction de rappel est l'élément en cours d'itération.

Le second argument reçu par la fonction de rappel est l'`index` de l'élément en cours d'itération. Il est facultatif.

Le troisième argument reçu par la fonction de rappel est le tableau sur lequel est utilisé la méthode. Il est également facultatif.

Il s'agit d'une fonction pure car elle n'a pas d'effet de bord, elle peut en outre être chaînée avec d'autres fonctions pures ce qui favorise grandement la **composition fonctionnelle**.

Vous pouvez utiliser également un raccourci syntaxique si la fonction de rappel peut utiliser directement les arguments. Par exemple :

```
const tableau = [1, 2, 3];
const tableau2 = tableau.map(Math.pow);
console.log(tableau2); // [1, 2, 9]
```

Cela équivaut exactement à :

```
const tableau = [1, 2, 3];
const tableau2 = tableau.map((el, i) => Math.pow(el,i));
console.log(tableau2); // [1, 2, 9]
```

Autre exemple :

```
const tableau = ["1", "2", "3"];
const tableau2 = tableau.map(Number);
console.log(tableau2); // [1, 2, 3]
```

Cela équivaut exactement à :

```
const tableau = ["1", "2", "3"];
const tableau2 = tableau.map(el => Number(el));
console.log(tableau2); // [1, 2, 3]
```

Cette fonction est extrêmement importante et est utilisée énormément en `JavaScript`, n'hésitez pas à passer du temps dessus.

La fonction `filter()`

La méthode `filter()` crée et retourne un nouveau tableau contenant les éléments qui passent le test de la fonction de rappel.

La fonction de rappel reçoit les mêmes arguments que pour celle de `map()`.

```
const tableau = [
  { prix: 42 },
```

```
  { prix: 2 },
  { prix: 12 },
  { prix: 50 },
  { },
  { prix: undefined },
  { prix: NaN },
  { prix: null }
];
const tableau2 = tableau.filter(el => el.prix > 10);
console.log(tableau2); // [{prix: 42}, {prix: 12}, {prix: 50}]
```

Ce qui est extrêmement courant et puissant est la combinaison de fonctions pures :

```
const tableau = ["12", "111", "4", "56", "42"];
const tableau2 = tableau.map(Number).filter(el => el > 20);
console.log(tableau2); // [111, 56, 42]
```