Les fonctions fléchées

Les fonctions fléchées sont des fonctions qui possèdent une syntaxe très compacte, ce qui les rend très rapide à écrire. Les fonctions fléchées utilisent le signe `=>` qui leur a donné leur nom à cause de sa forme de flèche, et souvent utilisées pour la manipulation des tableaux.

Déclaration :

```
JS index.js > ...
1   // Expression de fonction classique :
2   let somme = function(a, b) {
3       return a + b;
4   };
5
6   // Equivalent en fonction fléchée :
7   let somme = (a, b) => a + b;
8   alert(somme(1, 2));
9
10
```

De plus, notez que si notre fonction fléchée n'a besoin que d'un argument pour fonctionner, alors on pourra également omettre le couple de parenthèses

```
11  //Expression de fonction classique :
12  let double = function(n){
13      return n * 2
14  }
15
16  //Equivalent en fonction fléchée :
17  let double = n => n * 2;
18
19  alert(double(3));
20
```

Pour définir une fonction sans paramètre, on met juste des parenthèses vide

```
24  //Expression de fonction classique :
25  let helloWorld = function() {
26  return 'Hello World!'
27  }
28  console.log(helloWorld()) // Hello World!
29
30  //Equivalent en fonction fléchée :
31  let helloWorldES6 = () => 'Hello World!'
32  console.log(helloWorldES6()) // Hello World!
33
```

Particularités :

- **Les fonctions fléchées et le mot clef this**

Le mot clef this est utilisé pour accéder à des informations stockées dans l'objet. Le mot clef this va dans ce cas être substitué par l'objet utilisant la méthode lors de son appel.

```
let pierre = {
    nom: 'Giraud',
    prenom: 'Pierre',
    hobbies: ['Trail', 'Triathlon', 'Cuisine'],

    getFullName(){
        alert(this.prenom + ' ' + this.nom);
    }
};

pierre.getFullName();
```

Pour aller plus loin, vous devez savoir qu'en JavaScript, à la différence de la plupart des langages, **le mot clef this n'est pas lié à un objet en particulier**. En effet, **la valeur de this va être évaluée au moment de l'appel** de la méthode dans laquelle il est présent en JavaScript.

Ainsi, la valeur de this ne va pas dépendre de l'endroit où la méthode a été déclarée mais de l'objet qui l'appelle. Cela permet notamment à une méthode d'être réutilisée par différents objets.

```
let pierre = {name: 'Pierre'};
let mathilde = {name: 'Mathilde'};

function disBonjour(){
  alert('Bonjour ' + this.name);
}

pierre.bonjour = disBonjour;
mathilde.bonjour = disBonjour;

pierre.bonjour();
mathilde.bonjour();
```

Les fonctions fléchées, cependant, sont différentes des autres fonctions au sens où elles ne possèdent pas de valeur propre pour this : si on utilise ce mot clef dans une fonction fléchée, alors la valeur utilisée pour celui-ci sera celle du contexte de la fonction fléchée c'est-à-dire celle de la fonction englobante.

```
let pierre = {
    nom: 'Giraud',
    prenom: 'Pierre',
    hobbies: ['Trail', 'Triathlon', 'Cuisine'],

    disBonjour(){
        let bonjour = () => alert('Bonjour ' + this.prenom);
        bonjour();
    }
};

pierre.disBonjour(); //Bonjour Pierre
```

Remarque ☺ :

N'utilisez pas les fonctions fléchées aussi pour créer des constructeurs, sinon vous aurez de belles erreurs TypeError.