Les boucles en Javascript

Les boucles sont un fondement essentiel de n'importe quel langage de programmation. Elles vont vous permettre de parcourir des collections de données. Il existe plein de façon de créer des boucles en JS, dans cette partie nous allons surtout en voir 2.

1. La boucle **FOR**

```
for (let i = 0; i < 10; i++) {
    //instructions
}
```

Disséquons cette boucle :

- **let i = 0** : Correspond à l'expression initiale, cette dernière est généralement utilisée pour initialiser le ou les compteur(s) dont on aura besoin dans la boucle. On peut directement initialiser des variables à l'intérieur.
- **i < 10** : Correspond à la condition. Si elle est remplie, les instructions à l'intérieur de la boucle sont exécutées, sinon on sort directement de la boucle.
- **i++** : Correspond à la valeur d'incrémentation. On incrémente la valeur de i une fois les instructions exécutées.

Concrètement que se passe-t-il?

1. On initialise la valeur du compteur à 0 (i)
2. On vérifie la condition (savoir si i < 10 ici)
3. On exécute les instructions
4. Si elle est présente, on incrémente la valeur d'incrémentation (i = i + 1)
5. Retour à l'étape 2

EX :

```
for (let i = 0; i < 10; i++) {
    console.log(i)
}
```

Ici, on affichera 0, 1, 2, 3, 4, 5, 6, 7, 8, 9.

2. La boucle **WHILE**

Cette boucle permet de répéter une instruction tant qu'une condition donnée est vérifiée. On l'écrit de cette façon :

```
let y = 0;
while (y < 10){
    console.log(y)
    y = y + 1;
}
```

Si la condition d'entrée n'est pas vérifiée, on ne rentre jamais dans les instructions. Contrairement à la boucle for, l'incrémentation se fait ici plus en fin d'instruction (plutôt que dans les parenthèses). Faites **TRÈS** attention à ne pas oublier d'incrémenter votre compteur, sinon vous vous retrouverez dans une boucle infinie.
Dans l'exemple ci-dessus, si on oublie de faire **y = y + 1**, la valeur de y reste à 0, et la condition **y < 10** est toujours vérifiée après chaque fin d'instruction, la boucle ne s'arrêtera donc jamais (Votre navigateur va planter).

Parlons tableaux…

Comme vous venez de le voir, les boucles permettent de parcourir l'ensemble d'une collection. Vous trouverez des collections sous différentes formes au cours de votre vie de développeur, ici nous allons nous attarder sur le cas des tableaux.

1. Déclarer un tableau

Petit rappel, pour déclarer un tableau, il faudra utiliser des [ ] en Javascript.

```
let tableauVide = [] // initialisation d'un tableau vide

const jeuxvideos = ["League of Legend", "Rocket League", "Valorant", "Counter Strike", "Super Smash Bros"];
//Le tableau est initialisé avec des valeurs
```

2. Parcourir un tableau

Il faut maintenant pouvoir parcourir un tableau. On peut accéder à n'importe quel élément d'un tableau en renseignant son indice :

```
const jeuxvideos = ["League of Legend", "Rocket League", "Valorant", "Counter Strike", "Super Smash Bros"];

console.log(jeuxvideos[2]);//affichera Valorant
```

On notera ici que Valorant est la 3ème entrée du tableau, mais s'affiche à l'indice 2, tout simplement parce qu'un tableau commence à l'indice **0** ! Faites attention à cette propriété des tableaux !

Le but de cette mission va être de parcourir les tableaux présents dans le fichier joint en annexe de 2 façons différentes.

1. La première façon va être de parcourir les tableaux avec une boucle for et d'afficher, dans l'ordre croissant des tableaux, une phrase (grâce à la concaténation) :

   Le fruit du {jour} est : {fruit}

   {jour} représente le jour présent dans le tableau semaine et {fruit} le fruit présent dans le tableau fruits.
   On affichera donc ici les jours et les fruits dans l'ordre (lundi, mardi, mercredi…)

2. La deuxième façon va être de parcourir les mêmes tableaux mais avec une boucle while. Cette fois-ci on va les parcourir à l'envers ! Idem, on affichera dans une phrase (grâce à la concaténation) :

   Le fruit du {jour} est : {fruit}
   {jour} représente le jour présent dans le tableau semaine et {fruit} le fruit présent dans le tableau fruits.
   On affichera donc ici les jours et les fruits dans l'ordre inverse (dimanche, samedi, vendredi…)

Drop ton travail dans une archive au format zip sous le nom:

PrenomNom_BoucleJS.zip