Javascript


JAVASCRIPT
!= JAVA

Objectifs

1. Créer des **boucles**
2. **Parcourir** des tableaux

C'est quoi une boucle ?

Une boucle est un moyen d'exécuter le même code plusieurs fois jusqu'à ce qu'une certaine condition soit remplie.

C'est quoi une boucle ?

Une boucle est un moyen d'exécuter le même code plusieurs fois jusqu'à ce qu'une certaine condition soit remplie.

Par exemple, imaginons une voiture sur une piste de course : la voiture doit faire **x fois le tour** de la piste avant d'atteindre la fin du parcours.

La boucle “for”

for nécessite 3 paramètres pour fonctionner :

La boucle “for”

for nécessite 3 paramètres pour fonctionner :

- Le premier est une variable appelée **itérateur**, que nous devons créer et fixer à une valeur (ici nous voulons partir de zéro). Cette variable sera notre "compteur".

La boucle “for”

for nécessite 3 paramètres pour fonctionner :

- Le premier est une variable appelée **itérateur**, que nous devons créer et fixer à une valeur (ici nous voulons partir de zéro). Cette variable sera notre "compteur".
- La seconde est la **condition que nous voulons vérifier** avant chaque tour de boucle (itération).

La boucle “for”

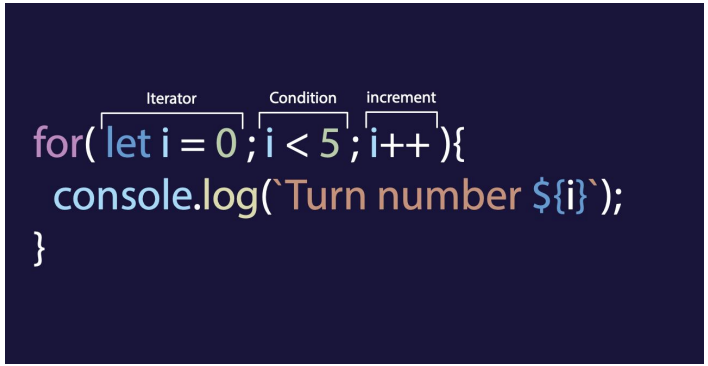
for nécessite 3 paramètres pour fonctionner :

- Le premier est une variable appelée **itérateur**, que nous devons créer et fixer à une valeur (ici nous voulons partir de zéro). Cette variable sera notre "compteur".
- La seconde est la **condition que nous voulons vérifier** avant chaque tour de boucle (itération).
- Et la troisième est **l'incrément**. L'incrément sera exécuté à la fin de chaque boucle et, généralement, nous ajoutons +1 à l'itérateur.

La boucle “for”

for nécessite 3 paramètres pour fonctionner :

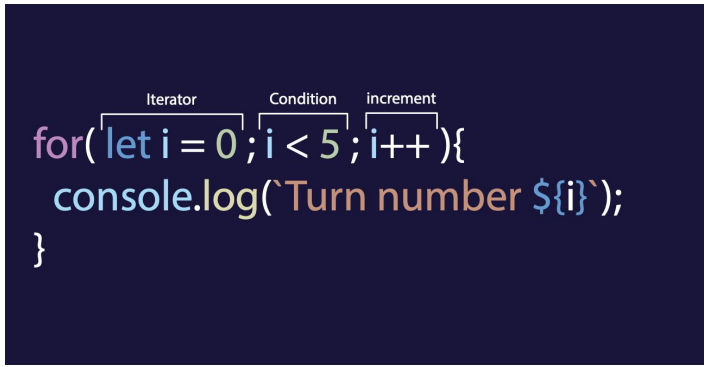
- Le premier est une variable appelée **itérateur**, que nous devons créer et fixer à une valeur (ici nous voulons partir de zéro). Cette variable sera notre "compteur".
- La seconde est la **condition que nous voulons vérifier** avant chaque tour de boucle (itération).
- Et la troisième est **l'incrément**. L'incrément sera exécuté à la fin de chaque boucle et, généralement, nous ajoutons +1 à l'itérateur.



La boucle “for”

for nécessite 3 paramètres pour fonctionner :

- Le premier est une variable appelée **itérateur**, que nous devons créer et fixer à une valeur (ici nous voulons partir de zéro). Cette variable sera notre "compteur".
- La seconde est la **condition que nous voulons vérifier** avant chaque tour de boucle (itération).
- Et la troisième est **l'incrément**. L'incrément sera exécuté à la fin de chaque boucle et, généralement, nous ajoutons +1 à l'itérateur.




```
for(let i = 0; i < 5; i++){
  console.log("Turn number " + i)
}

/*
Turn number 0
Turn number 1
Turn number 2
Turn number 3
Turn number 4
*/
```

JS

La boucle “for”

```
for(let i = 0; i < 5; i++){
  console.log("Turn number " + i)
}

/*
Turn number 0
Turn number 1
Turn number 2
Turn number 3
Turn number 4
*/
```

La boucle “for”

```
for(let i = 0; i < 5; i++){
  console.log("Turn number " + i)
}

/*
Turn number 0
Turn number 1
Turn number 2
Turn number 3
Turn number 4
*/
```

Nous créons d'abord la variable **i** qui est fixée à la valeur 0. Par convention, nous appelons souvent cette variable "i" (signifie itérateur ou index)

JS

La boucle “for”

```
for(let i = 0; i < 5; i++){
  console.log("Turn number " + i)
}

/*
Turn number 0
Turn number 1
Turn number 2
Turn number 3
Turn number 4
*/
```

Nous créons d'abord la variable **i** qui est fixée à la valeur 0. Par convention, nous appelons souvent cette variable "i" (signifie itérateur ou index)

Ensuite, nous voulons que les instructions dans la boucle soient répétées **tant qu'une certaine condition est vérifiée**. Dans ce cas, nous voulons faire une boucle jusqu'à ce que la valeur de "i" atteigne 5 (donc cinq fois).

JS

La boucle “for”

```
for(let i = 0; i < 5; i++){
  console.log("Turn number " + i)
}

/*
Turn number 0
Turn number 1
Turn number 2
Turn number 3
Turn number 4
*/
```

Nous créons d'abord la variable **i** qui est fixée à la valeur 0. Par convention, nous appelons souvent cette variable "i" (signifie itérateur ou index)

Ensuite, nous voulons que les instructions dans la boucle soient répétées **tant qu'une certaine condition est vérifiée**. Dans ce cas, nous voulons faire une boucle jusqu'à ce que la valeur de "i" atteigne 5 (donc cinq fois).

Pour chaque tour de boucle, **nous augmentons la valeur de "i" de un** (afin que la boucle s'arrête à un moment donné).

```
for(let i = 0; i < 5; i++){
  console.log("Turn number " + i)
}

/*
Turn number 0
Turn number 1
Turn number 2
Turn number 3
Turn number 4
*/
```

De cette façon, la boucle partira de 0, et vérifiera à chaque tour si "i" est inférieur à 5.

JS

```
for(let i = 0; i < 5; i++){
  console.log("Turn number " + i)
}

/*
Turn number 0
Turn number 1
Turn number 2
Turn number 3
Turn number 4
*/
```

De cette façon, la boucle partira de 0, et vérifiera à chaque tour si "i" est inférieur à 5.

Si **Oui** ⇒ Elle exécutera le code et augmentera la valeur de un. Et ensuite, on recommence.

```
for(let i = 0; i < 5; i++){
  console.log("Turn number " + i)
}

/*
Turn number 0
Turn number 1
Turn number 2
Turn number 3
Turn number 4
*/
```

De cette façon, la boucle partira de 0, et vérifiera à chaque tour si "i" est inférieur à 5.

Si **Oui** ⇒ Elle exécutera le code et augmentera la valeur de un. Et ensuite, on recommence.

Si **Non** ⇒ la boucle s'arrête.

JS

Parcourir un tableau

Il est possible d'utiliser une boucle pour **parcourir un tableau.**

```
const fruits = ["Apple", "Peach", "Banana"];

for(let i = 0; i < fruits.length; i ++){
  console.log(fruits[i]);
}
```

JS

Parcourir un tableau

Il est possible d'utiliser une boucle pour **parcourir un tableau.**

```
const fruits = ["Apple", "Peach", "Banana"];

for(let i = 0; i < fruits.length; i ++){
  console.log(fruits[i]);
}
```

Ici, on peut voir que nous avons un tableau qui contient 3 éléments.

JS

Parcourir un tableau

Il est possible d'utiliser une boucle pour **parcourir un tableau.**

```
const fruits = ["Apple", "Peach", "Banana"];

for(let i = 0; i < fruits.length; i ++){
  console.log(fruits[i]);
}
```

Ici, on peut voir que nous avons un tableau qui contient 3 éléments.

Nous créons une boucle **for**, dont l'itérateur va de 0 à **fruits.length** (soit 3).

JS

Parcourir un tableau

Il est possible d'utiliser une boucle pour **parcourir un tableau.**

```
const fruits = ["Apple", "Peach", "Banana"];

for(let i = 0; i < fruits.length; i ++){
  console.log(fruits[i]);
}
```

Ici, on peut voir que nous avons un tableau qui contient 3 éléments.

Nous créons une boucle for, dont l'itérateur va de 0 à fruits.length (soit 3).

Au premier tour, elle affichera fruits[0], puis fruits[1] et enfin fruits[2].

JS

Parcourir un tableau

Il est possible d'utiliser une boucle pour **parcourir un tableau.**

```
const fruits = ["Apple", "Peach", "Banana"];

for(let i = 0; i < fruits.length; i ++){
  console.log(fruits[i]);
}
```

```
const fruits = ["Apple", "Peach", "Banana"];

for(let i = 0; i < fruits.length; i ++){
  console.log(fruits[i]);
}

// will print :
// Apple
// Peach
// Banana
```

Ici, on peut voir que nous avons un tableau qui contient 3 éléments.

Nous créons une boucle for, dont l'itérateur va de 0 à fruits.length (soit 3).

Au premier tour, elle affichera fruits[0], puis fruits[1] et enfin fruits[2].

Portée - contexte

Même fonctionnement que pour les fonctions : on ne peut pas utiliser une variable déclarée à l'intérieur d'une boucle en dehors de celle-ci

```
let sum = 0;

for (let i = 0; i < 10; i++) {
  const name = "Pierre";
  console.log(name + " saw " + sum + " StarWars movies.");
  sum++;
  // fonctionne correctement dans le contexte de la boucle
}

console.log(sum);
// on verra la valeur de sum

console.log(name);
// une erreur s'affiche : 'reference error: name is not defined'
```

Pratiquons !