Les boucles en JavaScript vanilla

> Les boucles permettent de répéter un bloc de code plusieurs fois, selon des conditions spécifiques.
> En JavaScript, il existe plusieurs types de boucles, chacune adaptée à des cas particuliers.
> Comprendre leur fonctionnement est essentiel pour écrire un code efficace.

1. La boucle for

> La boucle `for` est utilisée lorsque le nombre d'itérations est connu à l'avance.

1.1. Syntaxe

```javascript
for (initialisation; condition; incrémentation) {
  // Code à exécuter
}
```

Exemple:

```javascript
for (let i = 0; i < 5; i++) {
  console.log(`Itération ${i}`);
}
```

1.2. Cas d'utilisation

- Parcourir une liste ou un tableau lorsque vous savez combien d'éléments il contient.
- Répéter une tâche un certain nombre de fois prédéfini.

2. La boucle while

> La boucle `while` répète un bloc de code tant qu'une condition donnée est vraie.

2.1. Syntaxe

```javascript
while (condition) {
  // Code à exécuter
}
```

Exemple:

```javascript
let compteur = 0;

while (compteur < 3) {
  console.log(`Compteur : ${compteur}`);
  compteur++;
}
```

2.2. Cas d'utilisation

- Exécuter une tâche tant qu'une condition n'est pas remplie.
- Utile lorsque le nombre d'itérations varie selon une logique interne.

3. La boucle do...while

> Cette boucle fonctionne comme une boucle `while`, mais elle exécute toujours au moins une itération avant de vérifier la condition.

3.1. Syntaxe

```javascript
do {
  // Code à exécuter
} while (condition);
```

Exemple:

```javascript
let compteur = 0;

do {
  console.log(`Compteur : ${compteur}`);
  compteur++;
} while (compteur < 3);
```

3.2. Cas d'utilisation

- Exécuter une tâche qui doit avoir lieu au moins une fois avant de vérifier une condition.

4. La boucle for...of

> Parcourt directement les éléments d'un objet itérable comme les tableaux, les chaînes de caractères, ou les ensembles (`Set`).

4.1. Syntaxe

```javascript
for (const element of iterable) {
  // Code à exécuter
}
```

Exemple:

```javascript
const fruits = ["pomme", "banane", "cerise"];

for (const fruit of fruits) {
  console.log(fruit);
}
```

4.2. Cas d'utilisation

- Parcourir les éléments d'un tableau ou d'un autre objet itérable sans gérer les index.

5. La boucle for...in

> Parcourt les clés ou propriétés d'un objet.

5.1. Syntaxe

```javascript
for (const clé in objet) {
  // Code à exécuter
}
```

Exemple:

```javascript
const personne = { nom: "Alice", age: 25 };

for (const propriete in personne) {
  console.log(`${propriete}: ${personne[propriete]}`);
}
```

5.2. Cas d'utilisation

- Accéder aux propriétés d'un objet, notamment lorsque leur nombre ou nom n'est pas connu à l'avance.

6. La boucle forEach

> Bien que ce ne soit pas une boucle au sens traditionnel, `forEach` est une méthode des tableaux qui permet d'exécuter une fonction sur chaque élément d'un tableau.
> Elle est souvent utilisée pour des opérations sur les éléments d'un tableau.
> Elle ne peut pas être utilisée pour des objets.

6.1. Syntaxe

```javascript
array.forEach((element, index) => {
  // Code à exécuter
});
```

Exemple : sans index

```
const fruits = ["pomme", "banane", "cerise"];

fruits.forEach(fruit => { console.log(fruit.toUpperCase());});

// POMME
// BANANE
// CERISE
```

Exemple : avec index

```javascript
const nombres = [1, 2, 3];
nombres.forEach((nombre, index) => {
  console.log(`Index ${index}: ${nombre}`);
});

// Index 0: 1
// Index 1: 2
// Index 2: 3
```

6.2. Cas d'utilisation

- Exécuter une fonction sur chaque élément d'un tableau.
- Idéal pour des opérations simples sur les éléments sans nécessiter de gestion d'index.
- Ne peut pas être utilisé pour des objets.

7. À RETENIR

- for : Pour un nombre d'itérations défini ou parcourir un tableau avec des indices.
- while : Pour des répétitions basées sur une condition non prédéfinie.
- do...while : Similaire à `while`, mais s'assure que le code s'exécute au moins une fois.
- for...of : Pour parcourir directement les éléments d'un tableau ou d'un objet itérable.
- for...in : Spécialement conçu pour parcourir les propriétés d'un objet.

Bien choisir la boucle adaptée à votre situation rendra votre code plus lisible et efficace !