codecademy

Boucles

Pendant que la boucle

La `while` boucle crée une boucle qui est exécutée tant qu'une condition spécifiée est évaluée à `true` . La boucle continuera à s'exécuter jusqu'à ce que la condition soit évaluée à `false` . La condition est spécifiée avant la boucle, et généralement, une variable est incrémentée ou modifiée dans le `while` corps de la boucle pour déterminer quand la boucle doit s'arrêter.

```
while (condition) {
  // code block to be executed
}

let i = 0;

while (i < 5) {
  console.log(i);
  i++;
}
```

Boucle inversée

Une `for` boucle peut itérer "en sens inverse" en initialisant la variable de boucle à la valeur de départ, en testant le moment où la variable atteint la valeur de fin et en décrémentant (en soustrayant) la variable de boucle à chaque itération.

```
const items = ['apricot', 'banana',
'cherry'];

for (let i = items.length - 1; i >= 0; i
-= 1) {
  console.log(`${i}. ${items[i]}`);
}

// Prints: 2. cherry
// Prints: 1. banana
// Prints: 0. apricot
```

Instruction Do...While

Une `do...while` instruction crée une boucle qui exécute un bloc de code une fois, vérifie si une condition est vraie, puis répète la boucle tant que la condition est vraie. Ils sont utilisés lorsque vous souhaitez que le code s'exécute toujours au moins une fois. La boucle se termine lorsque la condition est évaluée comme fausse.

```
x = 0
i = 0

do {
  x = x + i;
  console.log(x)
  i++;
} while (i < 5);

// Prints: 0 1 3 6 10
```

Pour la boucle

Une `for` boucle déclare des instructions de bouclage, avec trois informations importantes séparées par des points-virgules `;` :

L' *initialisation* définit où commencer la boucle en déclarant (ou en référençant) la variable d'itérateur

La *condition d'arrêt* détermine quand arrêter la boucle (lorsque l'expression est évaluée à `false`)

L' *instruction d'itération* met à jour l'itérateur chaque fois que la boucle est terminée

```
for (let i = 0; i < 4; i += 1) {
  console.log(i);
};

// Output: 0, 1, 2, 3
```

Bouclage à travers des tableaux

La longueur d'un tableau peut être évaluée avec la `.length` propriété. Ceci est extrêmement utile pour parcourir des tableaux, car le `.length` du tableau peut être utilisé comme condition d'arrêt dans la boucle.

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

// Output: Every item in the array
```

Casser le mot-clé

Dans une boucle, le `break` mot-clé peut être utilisé pour quitter la boucle immédiatement, en poursuivant l'exécution après le corps de la boucle.
Ici, le `break` mot-clé est utilisé pour sortir de la boucle lorsque `i` est supérieur à 5.

```
for (let i = 0; i < 99; i += 1) {
  if (i > 5) {
     break;
  }
  console.log(i)
}

// Output: 0 1 2 3 4 5
```

Boucle For imbriquée

Une boucle imbriquée `for` est lorsqu'une `for` boucle s'exécute à l'intérieur d'une autre `for` boucle.
La boucle interne exécutera toutes ses itérations pour *chaque* itération de la boucle externe.

```
for (let outer = 0; outer < 2; outer += 1) {
  for (let inner = 0; inner < 3; inner += 1) {
    console.log(`${outer}-${inner}`);
  }
}

/*
Output:
0-0
0-1
0-2
1-0
1-1
1-2
*/
```

Boucles

Une *boucle* est un outil de programmation utilisé pour répéter un ensemble d'instructions. *Itérer* est un terme générique qui signifie « répéter » dans le contexte des *boucles* . Une *boucle* continuera à *itérer* jusqu'à ce qu'une condition spécifiée, communément appelée *condition d'arrêt* , soit remplie.