Loops

In programming, loops are used to repeat a block of code.

For example, if you want to show a message 100 times, So instead of writing console.log 100 times you can use a loop that will just repeat a single console.log 100 times. It's just a simple example; you can achieve much more with loops.

Loops can be very useful where you want to repeat some set of statements over and over again.

Types of Loops:

1. **For loop** : Used when the number of times loop has to run is known
2. **While loop** : Used when u don't know how many times loop will run. While loop run on an condition.
3. **Do While loop**: Same as While but it will execute body of loop then check condition
4. **For of loop** : Used to Iterate over the array
5. **For in loop**: Used to iterate over the Objects

1. For Loop

The syntax of the for loop is:

```
for (initialExpression; condition; updateExpression) {
    // for loop body
}
```

Here,

1. The **initialExpression** initialises or declares variables and executes only once.
2. The condition is evaluated.
    - If the condition is false, the for loop is terminated.
    - If the condition is true, the block of code inside of the for loop is executed.

3. The **updateExpression** updates the value of initialExpression when the condition is true.
4. The condition is evaluated again. This process continues until the condition is false.

Example 1: Display a Text Five Times

```
// program to display text 5 times
const n = 5;

// looping from i = 1 to 5
for (let i = 1; i <= n; i++) {
    console.log(`I love JavaScript.`);
}
```

```
I love JavaScript.
I love JavaScript.
I love JavaScript.
I love JavaScript.
I love JavaScript.
```

Example 2: Display Numbers from 1 to 5

```
// program to display numbers from 1 to 5
const n = 5;

// looping from i = 1 to 5
// in each iteration, i is increased by 1
for (let i = 1; i <= n; i++) {
    console.log(i);     // printing the value of i
}
```

```
1
2
3
4
5
```

Example 3: Display Sum of n Natural Numbers

```
// program to display the sum of natural numbers
let sum = 0;
const n = 100

// looping from i = 1 to n
// in each iteration, i is increased by 1
for (let i = 1; i <= n; i++) {
    sum += i;  // sum = sum + i
}

console.log('sum:', sum);
```

Example 4: Print multiples of 20 till 100 in reverse order

```
for(var i = 100 ; i >= 0 ; i-=20)
{
    console.log(i)
}
```

```
100
80
60
40
20
0
```

JavaScript Infinite loop

If the test condition in any loop is always true, it runs forever (until memory is full). For example

```
// infinite for loop
for(let i = 1; i > 0; i++) {
    // block of code
}
```

In the above program, the condition is always true which will then run the code for infinite times

2. While Loop

The syntax of the while loop is:

```
while (condition) {
    // body of loop
}
```

Here,

1. A while loop evaluates the condition inside the parenthesis ().
2. If the condition evaluates to true, the code inside the while loop is executed.
3. The condition is evaluated again.
4. This process continues until the condition is false.
5. When the condition evaluates to false, the loop stops.

Example 1: Display Numbers from 1 to 5

```
// program to display numbers from 1 to 5
// initialize the variable
let i = 1, n = 5;

// while loop from i = 1 to 5
while (i <= n) {
    console.log(i);
    i += 1;
}
```

Output

```
1
2
3
4
5
```

Example 2: Print Multiplication table of 5

```
var i = 0;
while(i <= 10)
{
    console.log(5, "X", i, "=", 5*i)
    i+=1
}
```

```
5 X 0 = 0
5 X 1 = 5
5 X 2 = 10
5 X 3 = 15
5 X 4 = 20
5 X 5 = 25
5 X 6 = 30
5 X 7 = 35
5 X 8 = 40
5 X 9 = 45
5 X 10 = 50
```

3. Do while Loop

The syntax of do...while loop is:

```
do {
    // body of loop
} while(condition)
```

Here,

1. The body of the loop is executed at first. Then the condition is evaluated.
2. If the condition evaluates to true, the body of the loop inside the do statement is executed again.
3. The condition is evaluated once again.
4. If the condition evaluates to true, the body of the loop inside the do statement is executed again.
5. This process continues until the condition evaluates to false. Then the loop stops.

Example 1: Display Numbers from 1 to 5

```
// program to display numbers
let i = 1;
const n = 5;

// do...while loop from 1 to 5
do {
    console.log(i);
    i++;
} while(i <= n);
```

Output

```
1
2
3
4
5
```

for Vs while Loop

A for loop is usually used when the number of iterations is known. For example,

```
// this loop is iterated 5 times
for (let i = 1; i <=5; ++i) {
    // body of loop
}
```

And while and do...while loops are usually used when the number of iterations are unknown. For example,

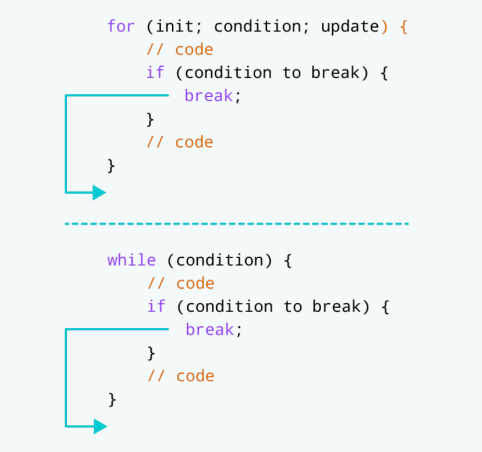
```
while (condition) {
    // body of loop
}
```

Break Statement

The break statement is used to **terminate** the loop **immediately** when it is encountered. Whenever we run a loop sometimes on some special conditions and situations we want to stop the loop in between, so the break keyword is used to stop the loop at any given point of time.

When the break keyword is executed the loop will terminate immediately and all the lines of code of loop below the break statement will be just ignored because loop is terminated.

Working of Break Keyword:



Working of JavaScript break Statement

Example 1:

```
// program to print the value of i
for (let i = 1; i <= 5; i++) {
    // break condition
    if (i == 3) {
        break;
    }
    console.log(i);
}
```

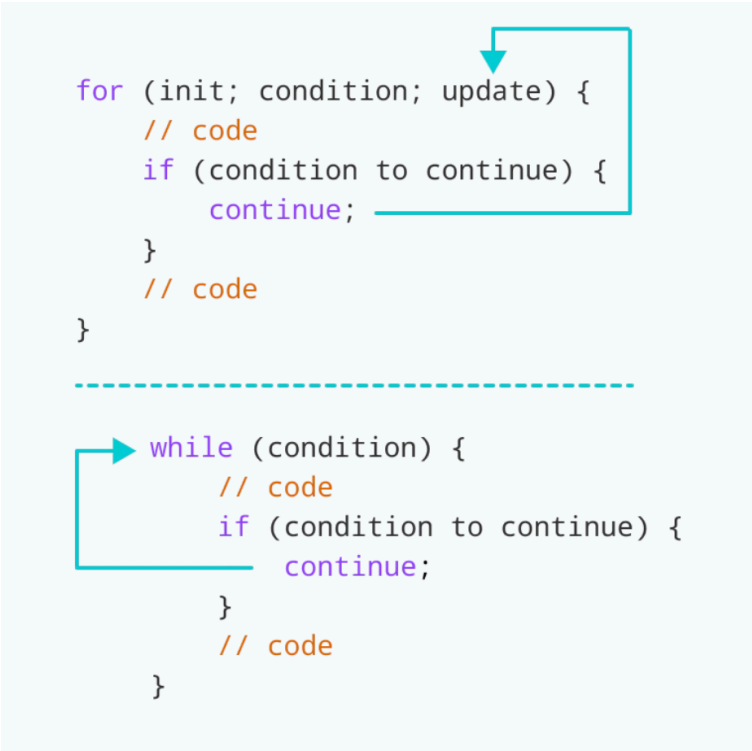
Explanation: In the above program, the for loop is used to print the value of i in each iteration.when i is equal to 3, the break statement terminates the loop. Hence, the output doesn't include values greater than or equal to 3.

Continue Statement

The continue statement is used to skip the current iteration of the loop and the control flow of the program goes to the next iteration.

When Continue keyword is hit, unlike break the whole loop will not stop just the current iteration of loop will be skipped and the loop will go back to the checking condition phase.

Working of JavaScript continue Statement



Example 1: Print the Value of i

```
// program to print the value of i
for (let i = 1; i <= 5; i++) {

    // condition to continue
    if (i == 3) {
        continue;
    }

    console.log(i);
}
```

Output

```
1
2
4
5
```

In the above program, for loop is used to print the value of i in each iteration.

Notice the continue statement inside the loop.

This means

- When i is equal to 3, the continue statement skips the third iteration.
- There for the 3 will not printed on output because 3rd iteration is skipped and it will update value of I to 4.
- Hence, 4 and 5 are printed in the next two iterations.

Note: Infinite Loop with Continue

```
// infinite loop
var i = 1;
while(i <= 5)
{

    if(i==3)
    {
        continue;
    }
    console.log(i)
    i+=1
}
```

```
var i = 1;
while(i <= 5)
{

    if(i==3)
    {
        i+=1
        continue;
    }
    console.log(i)
    i+=1
}
```

In the first example, the condition for continue is I==3. When I will become equals to 3 then the current iteration will be skipped and the loop will again go to check condition I <= 5. This will be an infinite loop because whenever the value of I becomes 3 it will just skip that iteration and I+=1 will never be executed due to skipping. So I will always remains 3 which cause and infinite loop.

The solution to this is second code. Before skipping the iteration we will update the value of I ourself.

3. For Of Loops

In Js for of loops can be used to traverse (going to every element of an array one by one) arrays or strings or any other ordered datatype.

for...of with Arrays

```
// array
const students = ['John', 'Sara', 'Jack'];

// using for...of
for ( let element of students ) {

    // display the values
    console.log(element);
}
```

```
Output

John
Sara
Jack
```

In the above program, the for...of loop is used to iterate over the students array object and display all its values.

for...of with Strings

```
// string
const string = 'code';

// using for...of loop
for (let i of string) {
    console.log(i);
}
```

```
Output

c
o
d
e
```

4. For In Loop

For in loop are used to iterate over all the values of an object using their keys.

The for..in loop in JavaScript allows you to iterate over all property keys of an object.

The syntax of the for...in loop is:

```
for (key in object) {
    // body of for...in
}
```

Example

```
const student = {
    name: 'Monica',
    class: 7,
    age: 12
}

// using for...in
for ( let key in student ) {

    // display the properties
    console.log(key, student[key]);
}
```

Output

```
node /tmp/wIi8qg3c3c.js
name Monica
class 7
age 12
```

Extra Resources to learn

1. **For Loop:** *https://www.programiz.com/javascript/for-loop*
2. **While Loop:** *https://www.programiz.com/javascript/while-loop*
3. **Do While Loop:** *https://www.programiz.com/javascript/while-loop*
4. **For in:** *https://www.programiz.com/javascript/for-in*
5. **For of:** *https://www.programiz.com/javascript/for-of*
6. **Break**: *https://www.programiz.com/javascript/break-statement*
7. **Continue**: *https://www.programiz.com/javascript/continue-statement*