

# Lesson:

## For Loop, Break and Continue



# Topics Covered

1. For loop
2. Nested For loop
3. Break Statement
4. Continue Statement

## Loops are used in programming to repeatedly run a block of code.

### Several loops in JS.

- for loop
- while loop
- do while loop

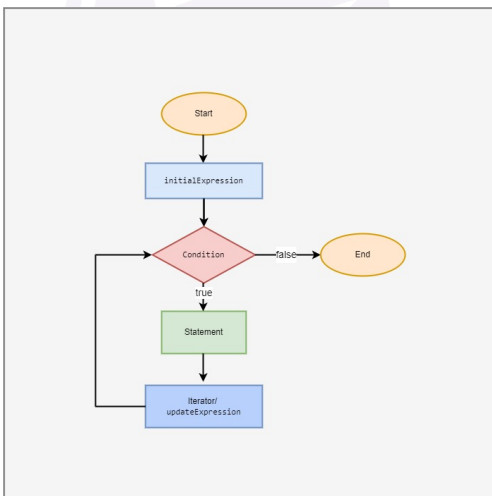
In this lesson, we will study about the **for loop**.

### For Loop

```
JavaScript
//Syntax

for (initial expression; condition; update expression) {
  // for loop body
}
```

## Understanding flow



1. **InitialExpression** only ever executes once while initializing and declaring variables.
2. The **condition** is assessed.
  - a. If the condition is **false**, the for loop will end.
  - b. If the condition is true,
    - i. The for loop's code block will be executed
    - ii. **Update** expression changes the **initial expression's** value.
  - c. The condition is once more assessed and the loop continues until the condition becomes false.

**Example 1:** Using For loop to print “PW Skills” 3 times.

```
JavaScript
for (let i = 0; i < 3; i++)
```

```
{
    let name = "PW Skills";
    console.log(name);
}
```

```
// Output
/*
PW Skills
PW Skills
PW Skills
*/
```

**Example 2:** Display a sequence of even numbers till 20

```
JavaScript
for (let i = 2; i <= 20; i+=2) {
    console.log(i);
}
```

```
// Output
/*
2
4
6
8
10
12
14
16
18
20
*/
```

## Nested for loop

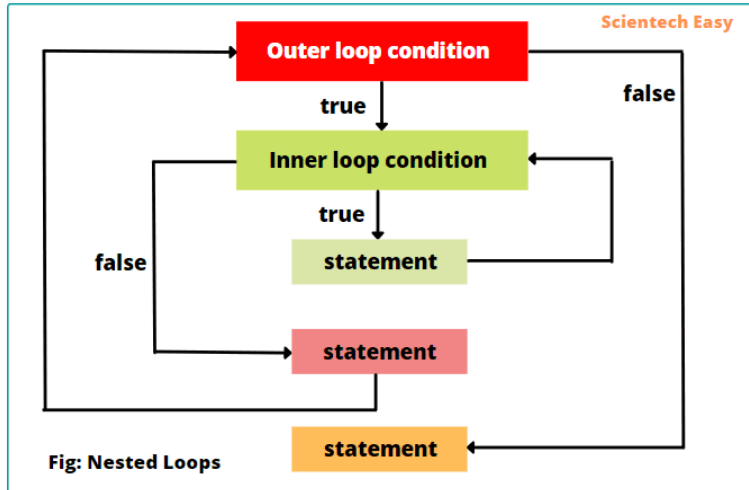
An outer for loop and one or more inside for loops constitute a nested for loop. Control re-enters the inner for loop and initiates a fresh execution every time the outer for loop repeats.

In other words, every time the outer for loop repeats, the control will enter the inner for loop.

Nested for loop often looks like this:

```
JavaScript
// Outer for loop.
for ( initialization; test-condition; increment/decrement )
{
    // Inner for loop.
    for ( initialization; test-condition; increment/decrement )
    {
        // statement of inner loop
    }
    // statement of outer loop
}
```

The execution flow is something like this:



**Example 1:** Write a program to show the inner for loop values for each outer iteration along with the inner for loop.

```
JavaScript
for(let i=1;i<=3;i++){ //outer loop
  console.log("for i= " + i + " the innerloop values are")
  for(let j=1;j<3;j++){ //inner loop
    console.log("j= "+j)
  }
}
```

// Output

```
/*
for i=1 the innerloop values are
j=1
j=2
for i=2 the innerloop values are
j=1
j=2
for i=3 the innerloop values are
```

```
j=1
j=2
*/
```

**Example 2:** Write a program to draw following pattern,

```
Unset
|*
* *
* * *
* * * *
* * * * *
* * * *
* * *
* *
*
*
```

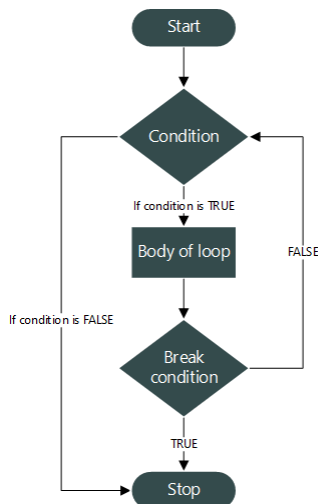
JavaScript

```
for (let i = 1; i <= 5; i++) {
  let row = "";
  for (let j = 1; j <= i; j++) {
    row += "* ";
  }
  console.log(row);
}

for (let i = 4; i >= 1; i--) {
  let row = "";
  for (let j = 1; j <= i; j++) {
    row += "* ";
  }
  console.log(row);
}
```

## Break Statement

We can terminate the loop explicitly using the break statement.



### Example of using break with for loop:

Can you guess the output of the following code ?

JavaScript

```
for (let i = 0; i < 4; i++) {
  console.log(i);
  if (i == 2) {
    break;
  }
}
```

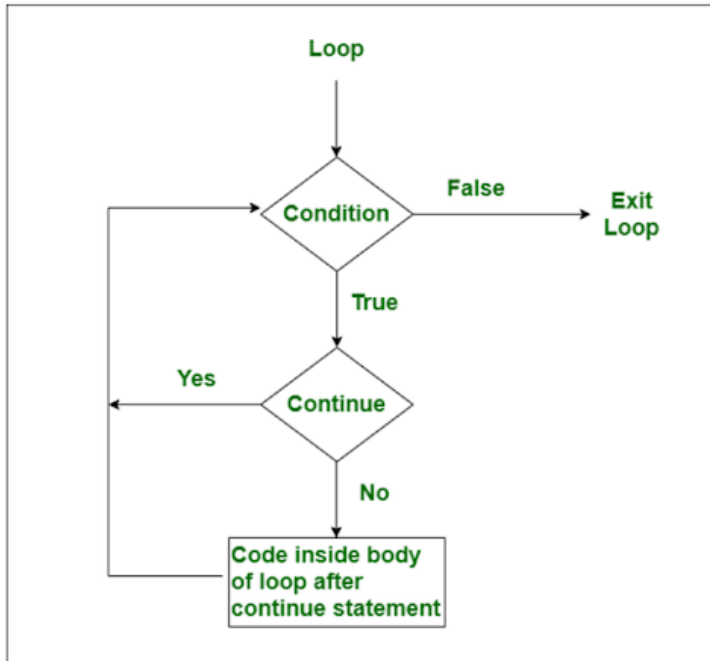
```
/* Output
0
1
2
*/
```

Here, we use an **if statement** inside the loop. If the current value of *i* becomes 2, the if statement will execute and the **break statement** will terminate the loop.

That is why we only see numbers till 2 in the output.

## Continue Statement

Continue statement skips the current iteration of loop, and moves on to the subsequent iteration.



### Example of using continue in for loop

Write a program to display only odd numbers till 20:

```

JavaScript
for (let i = 0; i < 20; i++) {
  if (i % 2 === 0) {
    continue;
  }
  console.log(i);
}
  
```

```

/* Output
1
3
5
7
9
11
13
15
17
19
*/
  
```

Here, the for loop iterates through the values from 0 to 20.

The remainder of the division of the current value of *i* by 2 is returned by the *i%2* expression.

The continue statement, which skips the current iteration of the loop and moves to the iterator expression *i++*, is executed if the remainder is zero. If not, the value of *i* is output to the console.