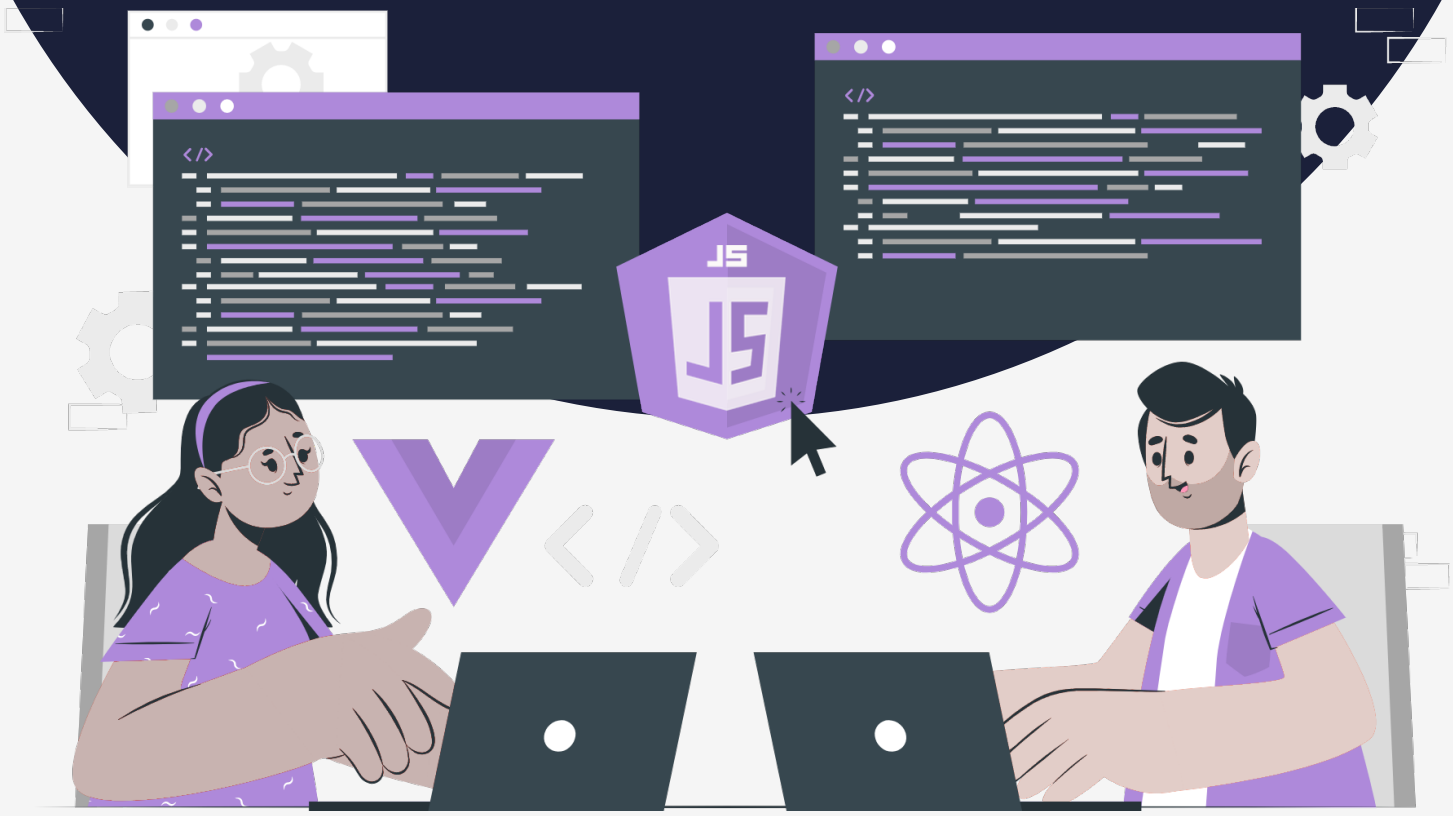
Lesson:

For loop, Break and Continue



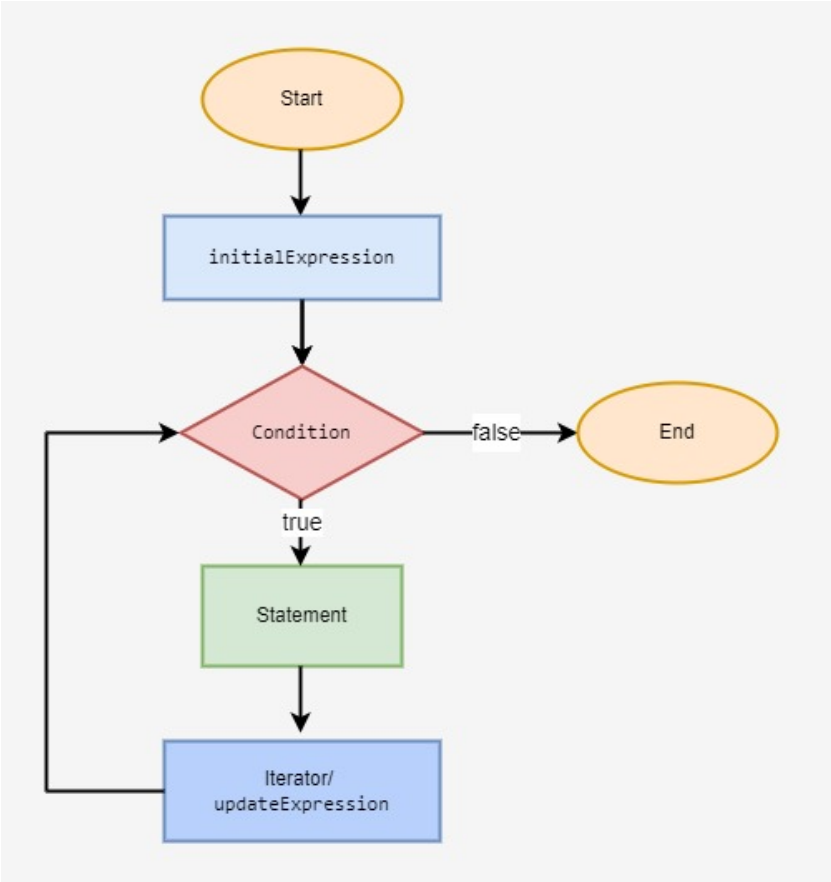
PW SKILLS

List of content:

For Loop

Syntax :

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



- InitialExpression only ever executes once while initializing and declaring variables.
- The condition is assessed.
- The for loop is ended if the condition is false.
- The for loop's code block is performed if the condition is satisfied.
- When the condition is true, the update expression changes the initial expression's value.
- The condition is once more assessed. Up till the condition is false, this process keeps going.

Example 1: Using For loop print "PW Skills" 3 times.

```
for (let i = 0; i < 3; i++)
{
    let name = "PW Skills";
    console.log(name);
  }
```

Output:

```
[Running] node "c:\Users\ACER\Desktop\First Pr
PW Skills
PW Skills
PW Skills

[Done] exited with code=0 in 0.106 seconds
```

Example 2: Display a sequence of even numbers till 20

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

```
[Running] node "c:\Users\ACER\Desktop\First Pr
2
4
6
8
10
12
14
16
18
20

[Done] exited with code=0 in 0.094 seconds
```

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:

```
// 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:

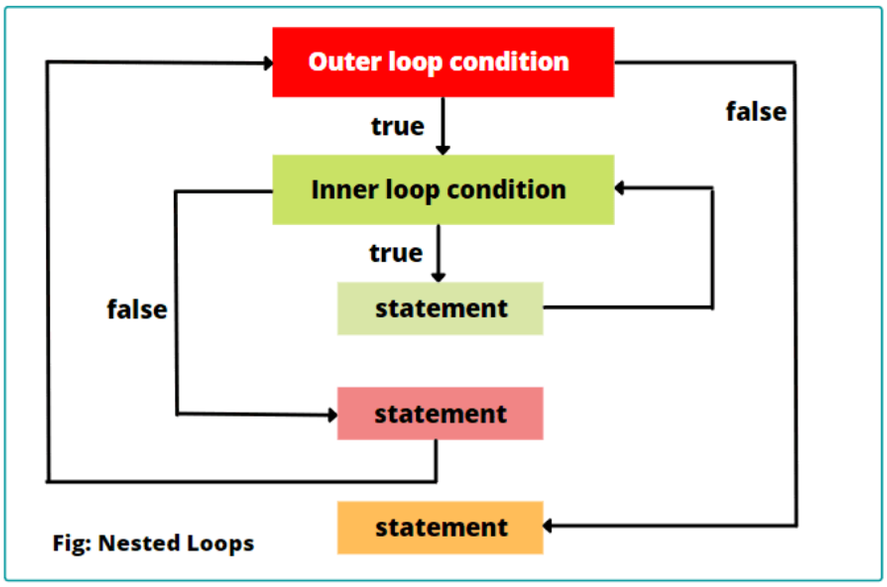


Fig: Nested Loops

Example 3 : Write a program to show the inner for loop values for each outer iteration in along with the outer "for" loop.

```
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)
    }
}
```

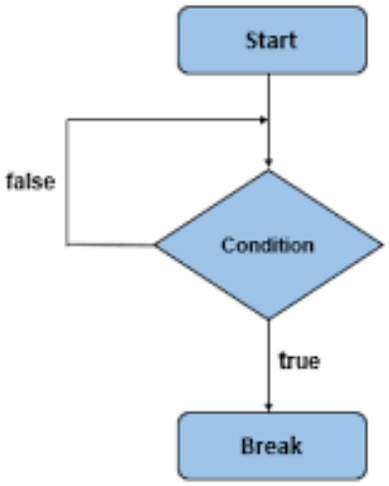
Output:

```
[Running] node "c:\Users\ACER\Desktop\Fi
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

[Done] exited with code=0 in 0.099 secor
```

Break Statement:
A loop such as a for, do...while, or while loop, a switch, or a label statement are all prematurely terminated by the break statement. The break statement's syntax is as follows:

break [label];



Example of using break with for loop:
Can you guess the output of the following code ?

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

Output:

```
[Running] node "c:\Users\ACER\Desktop\First
0
1
2

[Done] exited with code=0 in 0.087 seconds
```

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.

Example of using break with while loop

We will take the same example and see if it works in while loop

```
let i = 0;

while (i < 4) {
  console.log(i);
  i++;
  if (i == 3) {
    break;
  }
}
```

Output:

```
[Running] node "c:\Users\ACER\Desktop\First Pr
0
1
2

[Done] exited with code=0 in 0.092 seconds
```

Here again, we have used an if statement to check the condition for break, the moment i becomes 3, if statement is executed leading to loop termination.

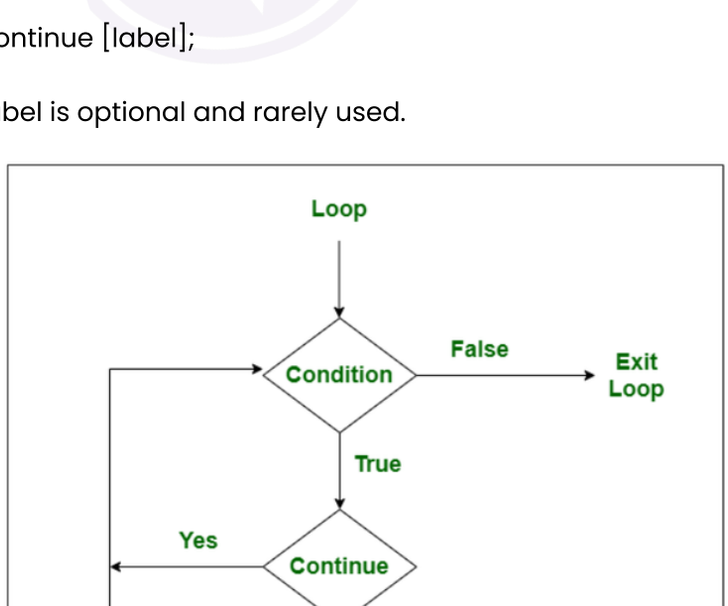
Notice and observe the difference between for and while loops for getting the same result.

Continue Statement:
The current iteration of the loop is skipped when using the continue statement, and the program moves on to the subsequent iteration.

The syntax of the continue statement is:

continue [label];

label is optional and rarely used.

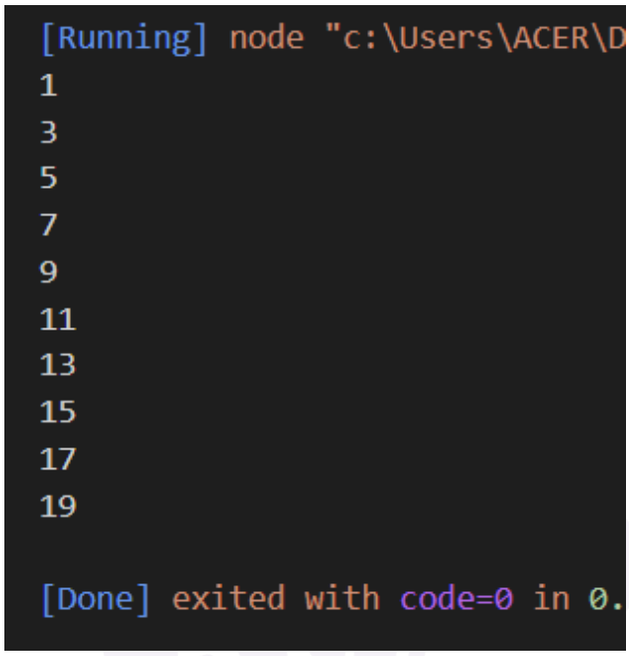


Example of using continue in for loop

Write a program to display only odd numbers till 20:

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

Output:



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.

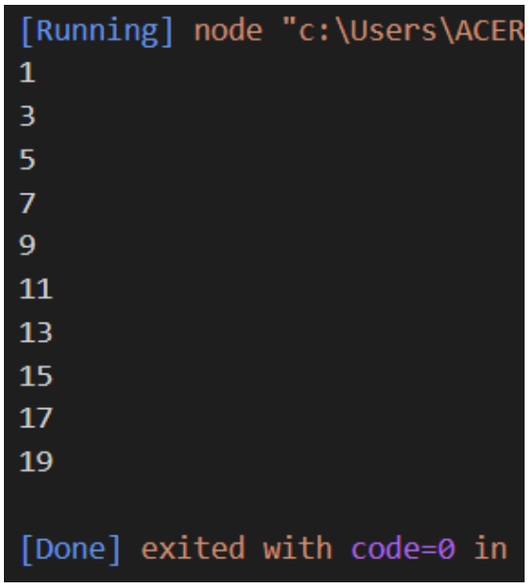
Example of using continue in while loop

We will use the same example here and implement it with while loop

Write a program to display only odd numbers till 20

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

Output:



Here, the while loop iterates through the values in this example 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.

We can also use a label here, to print the same result. If you are wondering how, here it is :

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

Here, the continue statement skips the execution and takes it to labelex.
You can use a label to identify a loop and the continue statement, here, to tell a program whether to skip the loop or keep running it. Note that JavaScript has no goto statement(as in C/C++); you can only use labels with break and continue.
Labeled loops are easier to track and understand with respect to program flow in case of continue.

Output: