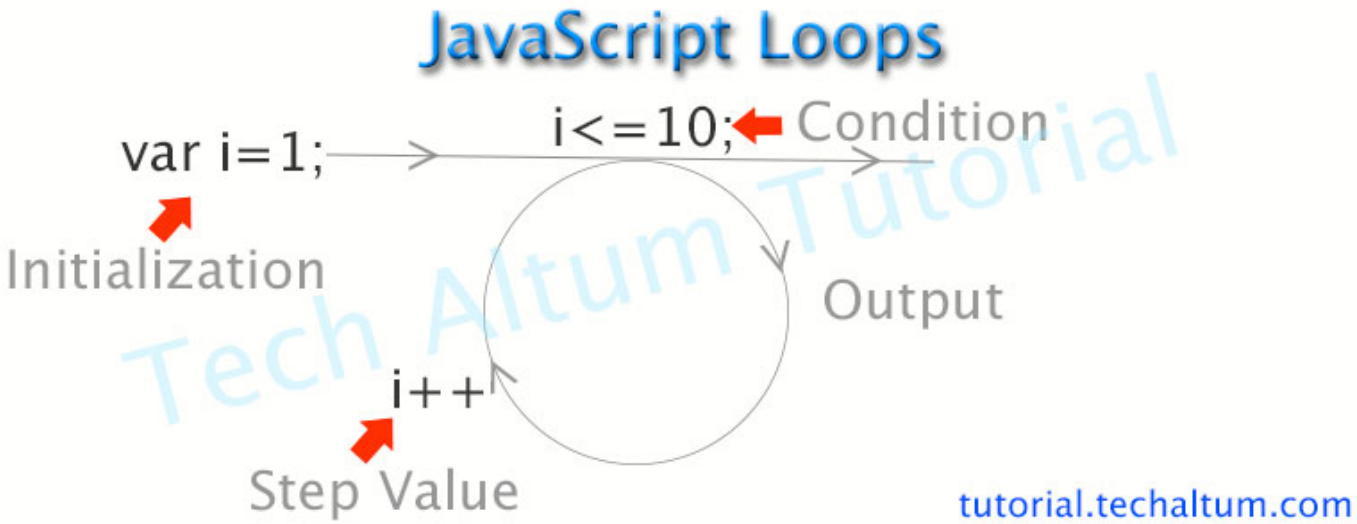
Loops and Iteration

JavaScript Loops or **looping** repeats a piece of code till the particular condition meets. Thus a loop will run a code block again and again until the condition is matched.



JavaScript Loops

The figure above is an example of simple loop where `var i=1` is initialization, `i<=10` is condition, and `i++` is step value. This loop will iterate ten times. After that, it will exit from loop and code will flow.

Type of loops in JavaScript

While Loop

While Loop is Entry Level Loop, which will repeatedly run a code block while a certain condition is true. **While** keyword works as a condition for loop. If condition is matched, then only loop will iterate.

```
 while(condition){
    code block;
}
```

While Loop Example

```
 var i=1;
 while(i<=10){                                          // iterate 10 times
   console.log(i);
   i++;
 }
```

In the example above, we've started with a variable declaration i=1. Any variable used inside a loop must be *initialized* first. If loop is not initialized, this will create syntax error.

The **while loop** starts with **while** keyword and condition in parenthesis. **while** condition says that keep repeating the code block, until i is less than or equal to 5.

i++ is **step value**, which is increment by 1 in this case. This is very important otherwise it will become **Infinite loop**. A Infinite Loop can block code flow and hang your computer. 😇

The above loop will print i five times, starting from 1 to 5.

1. Initialization, condition and increment are compulsory in while loop.
2. If increment is missing, loop will runs Infinitely and system will hang..
3. If i=12, and condition is i<=10, loop will never run.

Do While Loop

Do While is exit level loop. This will works even if first condition is false, as condition was checked later. This could be helpful where we wants our loop to run *at least once* even if the condition is not true.

```
do{
    code block;
}
while(condition)
```

Do while loop starts with **do**. While condition will be checked later. This will run code block at least once.

Do While Loop Example

```
var i=12;
do{
  console.log(i);
  i++;
   }
while(i<=10)
```

The above **do while loop** starts from 12, while condition is i<=10.

do will work first, and print value of i, i.e. 12.

i++ will increase i value from 12 to 13.

But while condition will terminate loop as i<=10 doesn't match our condition.

1. Initialization, condition and increment are compulsory in do while loop.
2. If increment is missing, loop will runs Infinitely .
3. If i=12, and condition is i<=10, loop will once and print 12.

For Loop

For loop is the most commonly used **loop in javascript**. Like **While loop**, **for loop** also check condition first and then execute, but in a neat and cleaner way.

```
for( initialization; condition; step value ){
    code block;
}
```

for loop starts with **for** keyword followed by parenthesis. Initialization, condition and step values are inserted inside. Initialization is declared inside for loop. If the condition is ok, only then the loop will continue.

For Loop Example

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

The above **for loop** starts from 2 and the condition is i<=20.

i=i+2 will increase the value of i by 2.

value of i is printed only if condition is matched.

The above loop will return Table of 2.

1. Initialization, condition and increment are compulsory in for loop.
2. If increment is missing, loop will runs Infinitely .
3. If i=12, and condition is i<=10, loop will not work.

break in loop

To break or terminate a loop or switch in between, we can use **break keyword**. This will stop further iterations of loop.

loop without break

```
for( var i=1; i<=10; i++){
  if( i%5==0){ console.log(i);}
  // print 5 and 10
}
```

loop with break

```
for( var i=1; i<=10; i++){
  if( i%5==0){ console.log(i); break}
  // print 5 only
}
```

Nested For Loop

Nested Loop means a *loop inside another loop*. We can add one or more loop inside a **for loop**. The inner loop will execute with multiple iteration on each outer loop iteration. Will See example of **Javascript for in loop** in next article.

```
for( initialization; condition; step value ){
    for( initialization; condition; step value ){
        code block;
    }
}
```

Nested For Loop is used to calculate table of numbers from 1 to 10. Here is an example.

1, 2, 3, 4, 5, 6, 7, 8, 9, 10,

2, 4, 6, 8, 10, 12, 14, 16, 18, 20,

3, 6, 9, 12, 15, 18, 21, 24, 27, 30,

4, 8, 12, 16, 20, 24, 28, 32, 36, 40,

5, 10, 15, 20, 25, 30, 35, 40, 45, 50,

6, 12, 18, 24, 30, 36, 42, 48, 54, 60,

7, 14, 21, 28, 35, 42, 49, 56, 63, 70,

8, 16, 24, 32, 40, 48, 56, 64, 72, 80,

9, 18, 27, 36, 45, 54, 63, 72, 81, 90,

10, 20, 30, 40, 50, 60, 70, 80, 90, 100,

```
for( var i=1; i<=10; i++){                  //10 Times
    for( var j=1; j<=10; j++){              //10 Times
        document.write((i*j) +", ");
    }
    document.write("<hr>");
}
```

In first iteration, when i=1, and j=1, i*j will be 1.

In Second iteration, when i=1, and j=2, i*j will be 2.

In Third iteration, when i=1, and j=3, i*j will be 3.

In 11th iteration, when i=2, and j=1, i*j will be 2.

In 21st iteration, when i=3, and j=1, i*j will be 3.

In last iteration, when i=10, and j=10, i*j will be 100.

For in Loop

for in loop is used in Arrays and Objects. In Array, a **for in loop** will return index of array. But in Object, **for in loop** will return key of Object.