**JavaScript Tutorial: for, while, forEach, Do While Loops | Web Development Tutorials #55**

In this tutorial, we are going to learn about different loops used in JavaScript and how to use them to iterate through the arrays. Make a new file as *tut55.html*and add the boilerplate to the basic HTML template. Loops can execute a block of code a number of times. Loops are handy, if you want to run the same code over and over again, each time with a different value.

Let us start by understanding **for loops.**It loops through a block of code a number of times. If we write the code as follows-

let i = 0;

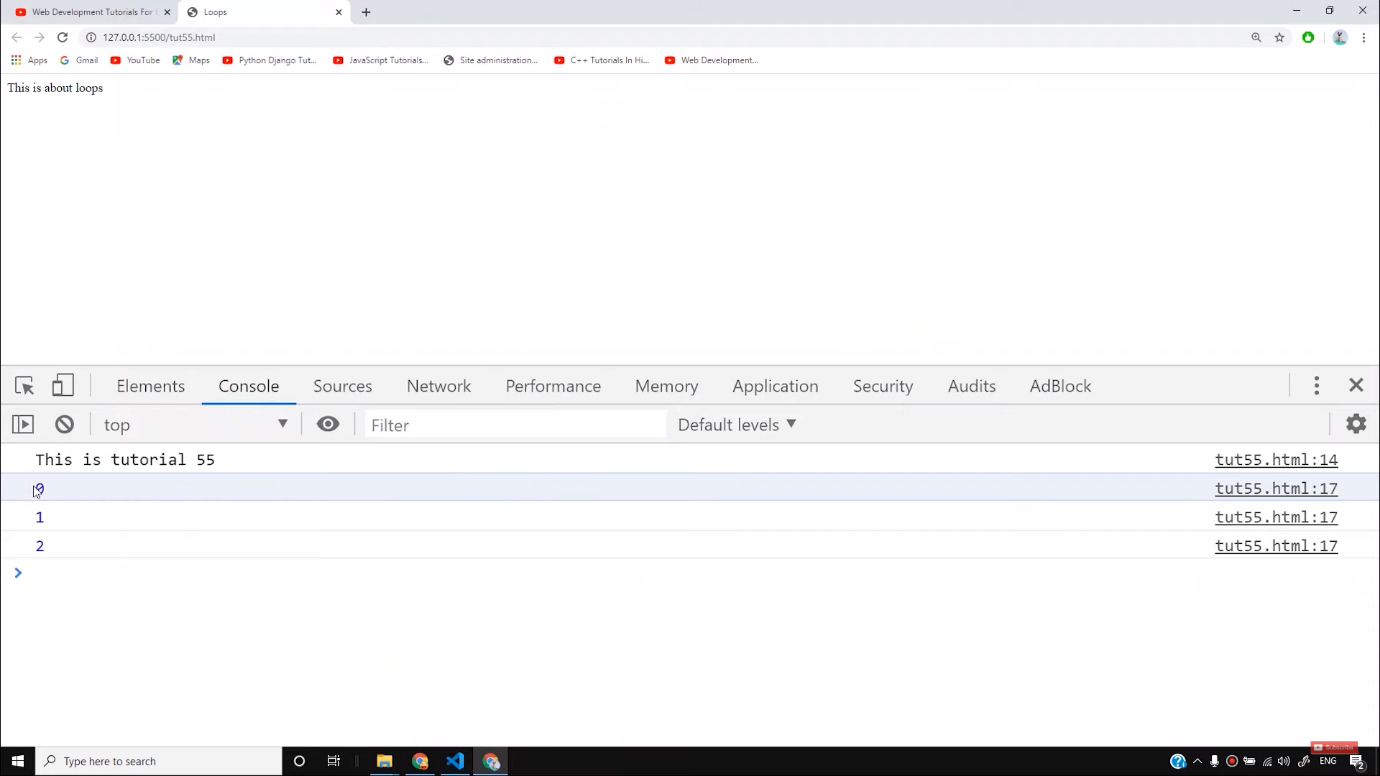
for(i=0; i<3;i++){

console.log(i);

}

Copy

In the above example, the for loop will print the value of **i**until it is less than 3. The output of the above code is as follows-



Let us now see how to use for loop to iterate an array. If we write the code as follows-

let friends = ["Rohan", "Sanjeev", "Deepti", "Pooja", "SkillF"];

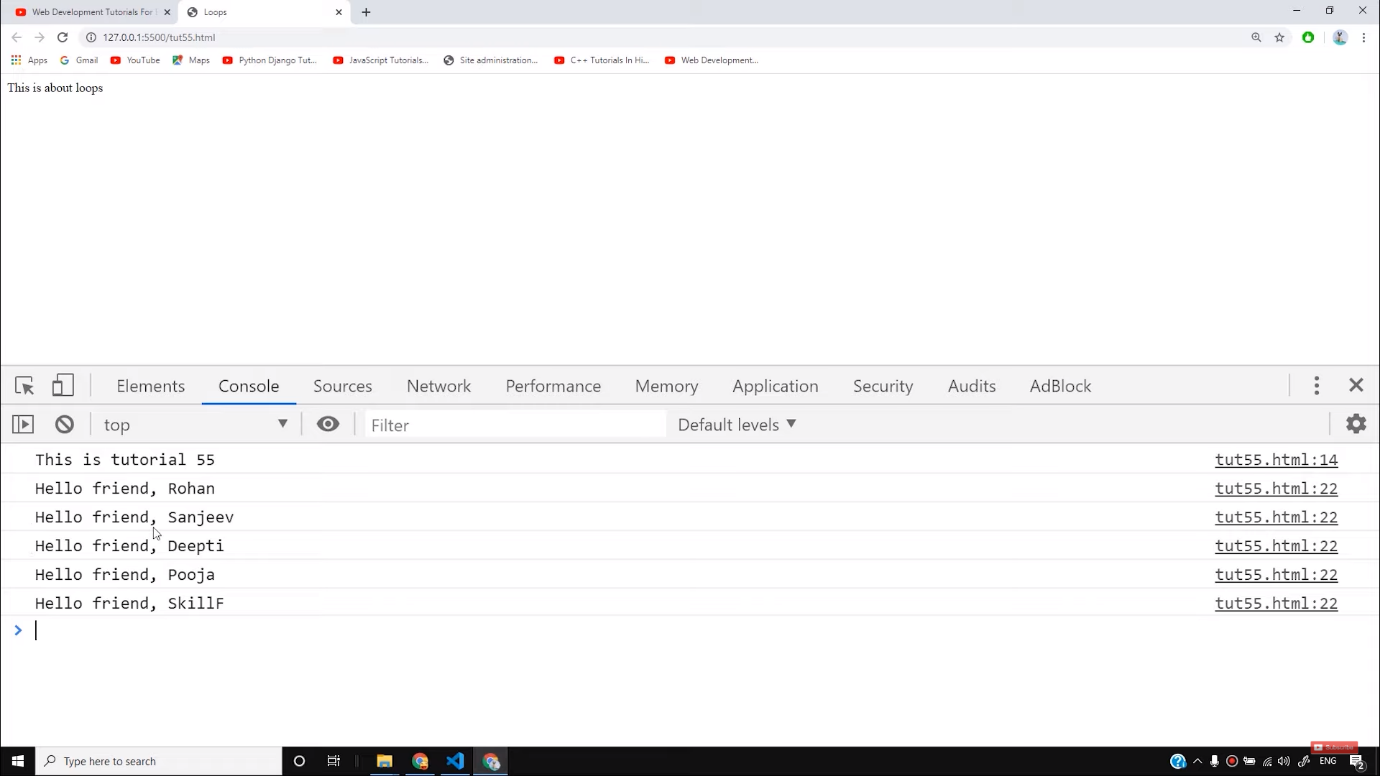
for (let index = 0; index < friends.length; index++) {

console.log("Hello friend, " + friends[index]);

}

Copy

In this example, as the value of **index**increases, the name of all the friends will be printed until it becomes equal to the length of an array. The output will be as follows-



However, there is another simpler way to write the same loop with the help of **forEach**.It is written as follows-

friends.forEach(function f(element){

console.log("Hello Friend, " + element + " to modern JavaScript");

});

Copy

There is one more way of writing it using **for/of loops.**It loops through the values of an iterable object. It is written as follows-

for (element of friends){

console.log("Hello Friend, " + element + " to modern JavaScript again");

}

Copy

To iterate through an object, we can use **for/in**loops. Let us understand with the example below-

let employee = {

name: "Harry",

salary: 2,

channel: "CWH"

}

// Use this loop to iterate over objects in JavaScript

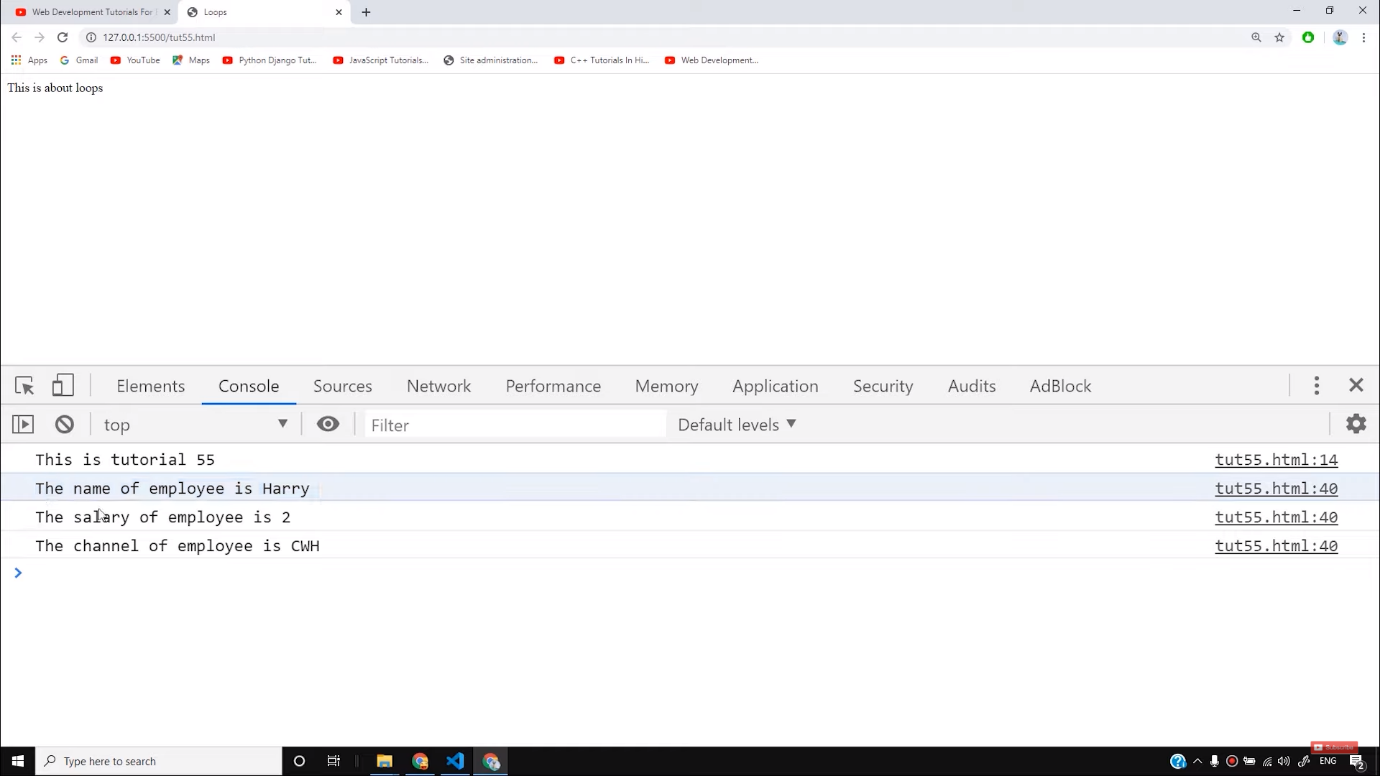
for(key in employee){

console.log(`The ${key} of employee is ${employee[key]}`);

}

Copy

This loop will iterate through all the elements of an object **employee**and the result will be as follows-



Now we will see how to use **while loops** in JavaScript. The while loop loops through a block of code as long as a specified condition is true. The syntax is as follows-

let i =0;

while(i<4){

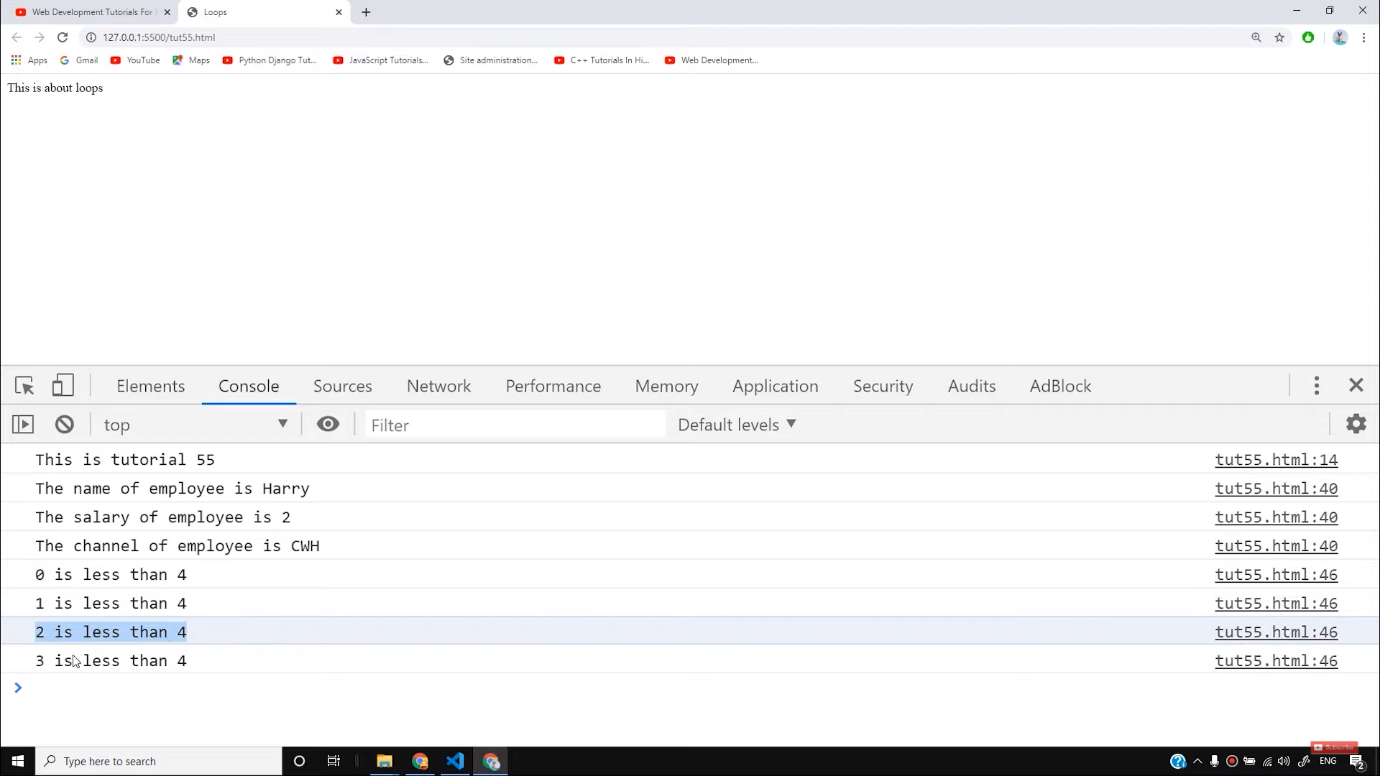
console.log(`${i} is less than 4`);

i++;

}

Copy

This loop will execute until the value of i is less than 4. The output will be as follows-



There is another loop known as **do-while loops.**It is a variant of the while loop. It will execute the code block once, before checking the condition, then it will repeat the loops as long as the condition is true. The example is as follows-

let j =34;

do{

console.log(`${j} is less than 4 and we are inside do while loop`);

j++;

}while(j<4);

Copy

So I hope you must have understood about different loops used in JavaScript. From the next tutorials, we will do some practical implementation from these loops and understand them better.

**Code as described/written in the video**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<meta http-equiv="X-UA-Compatible" content="ie=edge">

<title>Loops</title>

</head>

<body>

<div class="container">

This is about loops

</div>

<script>

console.log("This is tutorial 55");

// let i = 0;

// for(i=0; i<3;i++){

// console.log(i);

// }

let friends = ["Rohan", "Sanjeev", "Deepti", "Pooja", "SkillF"];

// for (let index = 0; index < friends.length; index++) {

// console.log("Hello friend, " + friends[index]);

// }

// friends.forEach(function f(element){

// console.log("Hello Friend, " + element + " to modern JavaScript");

// });

// for (element of friends){

// console.log("Hello Friend, " + element + " to modern JavaScript again");

// }

let employee = {

name: "Harry",

salary: 2,

channel: "CWH"

}

// Use this loop to iterate over objects in JavaScript

for(key in employee){

console.log(`The ${key} of employee is ${employee[key]}`);

}

// while loop in js

let i =0;

while(i<4){

console.log(`${i} is less than 4`);

i++;

}

// do while loop in js

let j =34;

do{

console.log(`${j} is less than 4 and we are inside do while loop`);

j++;

}while(j<4);

</script>

</body>

</html>

# avaScript For Loop

[❮ Previous](https://www.w3schools.com/js/js_switch.asp)[Next ❯](https://www.w3schools.com/js/js_loop_forin.asp)

Loops can execute a block of code a number of times.

## **JavaScript Loops**

Loops are handy, if you want to run the same code over and over again, each time with a different value.

Often this is the case when working with arrays:

### **Instead of writing:**

text += cars[0] + "<br>";  
text += cars[1] + "<br>";  
text += cars[2] + "<br>";  
text += cars[3] + "<br>";  
text += cars[4] + "<br>";  
text += cars[5] + "<br>";

### **You can write:**

for (let i = 0; i < cars.length; i++) {  
  text += cars[i] + "<br>";  
}

[Try it Yourself »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_loop_for)

## **Different Kinds of Loops**

JavaScript supports different kinds of loops:

* for - loops through a block of code a number of times
* for/in - loops through the properties of an object
* for/of - loops through the values of an iterable object
* while - loops through a block of code while a specified condition is true
* do/while - also loops through a block of code while a specified condition is true

## **The For Loop**

The for loop has the following syntax:

for (*statement 1*;*statement 2*;*statement 3*) {  
  // *code block to be executed*  
}

**Statement 1** is executed (one time) before the execution of the code block.

**Statement 2** defines the condition for executing the code block.

**Statement 3** is executed (every time) after the code block has been executed.

### **Example**

for (let i = 0; i < 5; i++) {  
  text += "The number is " + i + "<br>";  
}

[Try it Yourself »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_loop_for_ex)

From the example above, you can read:

Statement 1 sets a variable before the loop starts (let i = 0).

Statement 2 defines the condition for the loop to run (i must be less than 5).

Statement 3 increases a value (i++) each time the code block in the loop has been executed.

ADVERTISEMENT

## **Statement 1**

Normally you will use statement 1 to initialize the variable used in the loop (let i = 0).

This is not always the case, JavaScript doesn't care. Statement 1 is optional.

You can initiate many values in statement 1 (separated by comma):

### **Example**

for (let i = 0, len = cars.length, text = ""; i < len; i++) {  
  text += cars[i] + "<br>";  
}

[Try it Yourself »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_loop_for_om1)

And you can omit statement 1 (like when your values are set before the loop starts):

### **Example**

let i = 2;  
let len = cars.length;  
let text = "";  
for (; i < len; i++) {  
  text += cars[i] + "<br>";  
}

[Try it Yourself »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_loop_for_om2)

## **Statement 2**

Often statement 2 is used to evaluate the condition of the initial variable.

This is not always the case, JavaScript doesn't care. Statement 2 is also optional.

If statement 2 returns true, the loop will start over again, if it returns false, the loop will end.

If you omit statement 2, you must provide a **break** inside the loop. Otherwise the loop will never end. This will crash your browser. Read about breaks in a later chapter of this tutorial.

## **Statement 3**

Often statement 3 increments the value of the initial variable.

This is not always the case, JavaScript doesn't care, and statement 3 is optional.

Statement 3 can do anything like negative increment (i--), positive increment (i = i + 15), or anything else.

Statement 3 can also be omitted (like when you increment your values inside the loop):

### **Example**

let i = 0;  
let len = cars.length;  
let text = "";  
for (; i < len; ) {  
  text += cars[i] + "<br>";  
  i++;  
}

[Try it Yourself »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_loop_for_om3)

## **Loop Scope**

Using var in a loop:

### **Example**

var i = 5;  
  
for (var i = 0; i < 10; i++) {  
  // some code  
}  
  
// Here i is 10

[Try it Yourself »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_let_for_loop1)

Using let in a loop:

### **Example**

let i = 5;  
  
for (let i = 0; i < 10; i++) {  
  // some code  
}  
  
// Here i is 5

[Try it Yourself »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_let_for_loop2)

In the first example, using var, the variable declared in the loop redeclares the variable outside the loop.

In the second example, using let, the variable declared in the loop does not redeclare the variable outside the loop.

When let is used to declare the i variable in a loop, the i variable will only be visible within the loop.

# JavaScript For In

[❮ Previous](https://www.w3schools.com/js/js_loop_for.asp)[Next ❯](https://www.w3schools.com/js/js_loop_forof.asp)

## **The For In Loop**

The JavaScript for in statement loops through the properties of an Object:

### **Syntax**

for (key in object) {  
  // *code block to be executed*  
}

### **Example**

const person = {fname:"John", lname:"Doe", age:25};  
  
let text = "";  
for (let x in person) {  
  text += person[x];  
}

[Try it Yourself »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_object_for_in)

## **Example Explained**

* The **for in** loop iterates over a **person** object
* Each iteration returns a **key** (x)
* The key is used to access the **value** of the key
* The value of the key is **person[x]**

## **For In Over Arrays**

The JavaScript for in statement can also loop over the properties of an Array:

### **Syntax**

for (variable in array) {  
  code  
}

### **Example**

const numbers = [45, 4, 9, 16, 25];  
  
let txt = "";  
for (let x in numbers) {  
  txt += numbers[x];  
}

[Try it Yourself »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_array_for_in)

Do not use **for in** over an Array if the index **order** is important.

The index order is implementation-dependent, and array values may not be accessed in the order you expect.

It is better to use a **for** loop, a **for of** loop, or **Array.forEach()** when the order is important.

ADVERTISEMENT

## **Array.forEach()**

The forEach() method calls a function (a callback function) once for each array element.

### **Example**

const numbers = [45, 4, 9, 16, 25];  
  
let txt = "";  
numbers.forEach(myFunction);  
  
function myFunction(value, index, array) {  
  txt += value;  
}

[Try it Yourself »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_array_foreach)

Note that the function takes 3 arguments:

* The item value
* The item index
* The array itself

The example above uses only the value parameter. It can be rewritten to:

### **Example**

const numbers = [45, 4, 9, 16, 25];  
  
let txt = "";  
numbers.forEach(myFunction);  
  
function myFunction(value) {  
  txt += value;  
}

# JavaScript For Of

[❮ Previous](https://www.w3schools.com/js/js_loop_forin.asp)[Next ❯](https://www.w3schools.com/js/js_loop_while.asp)

## **The For Of Loop**

The JavaScript for of statement loops through the values of an iterable object.

It lets you loop over iterable data structures such as Arrays, Strings, Maps, NodeLists, and more:

### **Syntax**

for (variable of iterable) {  
  // *code block to be executed*  
}

**variable** - For every iteration the value of the next property is assigned to the variable. Variable can be declared with const, let, or var.

**iterable** - An object that has iterable properties.

## **Browser Support**

**For/of** was added to JavaScript in 2015 ([ES6](https://www.w3schools.com/js/js_es6.asp))

Safari 7 was the first browser to support for of:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
| Chrome 38 | Edge 12 | Firefox 51 | Safari 7 | Opera 25 |
| Oct 2014 | Jul 2015 | Oct 2016 | Oct 2013 | Oct 2014 |

**For/of** is not supported in Internet Explorer.

## **Looping over an Array**

### **Example**

const cars = ["BMW", "Volvo", "Mini"];  
  
let text = "";  
for (let x of cars) {  
  text += x;  
}

[Try it Yourself »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_object_for_of)

## **Looping over a String**

### **Example**

let language = "JavaScript";  
  
let text = "";  
for (let x of language) {  
text += x;  
}

# JavaScript While Loop

[❮ Previous](https://www.w3schools.com/js/js_loop_forof.asp)[Next ❯](https://www.w3schools.com/js/js_break.asp)

Loops can execute a block of code as long as a specified condition is true.

## **The While Loop**

The while loop loops through a block of code as long as a specified condition is true.

### **Syntax**

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

### **Example**

In the following example, the code in the loop will run, over and over again, as long as a variable (i) is less than 10:

### **Example**

while (i < 10) {  
  text += "The number is " + i;  
  i++;  
}

[Try it Yourself »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_while)

If you forget to increase the variable used in the condition, the loop will never end. This will crash your browser.

## **The Do While Loop**

The do while loop is a variant of the while loop. This loop will execute the code block once, before checking if the condition is true, then it will repeat the loop as long as the condition is true.

### **Syntax**

do {  
*// code block to be executed*}  
while (condition);

### **Example**

The example below uses a do while loop. The loop will always be executed at least once, even if the condition is false, because the code block is executed before the condition is tested:

### **Example**

do {  
  text += "The number is " + i;  
  i++;  
}  
while (i < 10);

[Try it Yourself »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_dowhile)

Do not forget to increase the variable used in the condition, otherwise the loop will never end!

ADVERTISEMENT

## **Comparing For and While**

If you have read the previous chapter, about the for loop, you will discover that a while loop is much the same as a for loop, with statement 1 and statement 3 omitted.

The loop in this example uses a for loop to collect the car names from the cars array:

### **Example**

const cars = ["BMW", "Volvo", "Saab", "Ford"];  
let i = 0;  
let text = "";  
  
for (;cars[i];) {  
  text += cars[i];  
  i++;  
}

[Try it Yourself »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_loop_for_cars)

The loop in this example uses a while loop to collect the car names from the cars array:

### **Example**

const cars = ["BMW", "Volvo", "Saab", "Ford"];  
let i = 0;  
let text = "";  
  
while (cars[i]) {  
  text += cars[i];  
  i++;  
}

# JavaScript Break and Continue

[❮ Previous](https://www.w3schools.com/js/js_loop_while.asp)[Next ❯](https://www.w3schools.com/js/js_iterables.asp)

The break statement "jumps out" of a loop.

The continue statement "jumps over" one iteration in the loop.

## **The Break Statement**

You have already seen the break statement used in an earlier chapter of this tutorial. It was used to "jump out" of a switch() statement.

The break statement can also be used to jump out of a loop:

### **Example**

for (let i = 0; i < 10; i++) {  
  if (i === 3) { break; }  
  text += "The number is " + i + "<br>";  
}

[Try it Yourself »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_break)

In the example above, the break statement ends the loop ("breaks" the loop) when the loop counter (i) is 3.

## **The Continue Statement**

The continue statement breaks one iteration (in the loop), if a specified condition occurs, and continues with the next iteration in the loop.

This example skips the value of 3:

### **Example**

for (let i = 0; i < 10; i++) {  
  if (i === 3) { continue; }  
  text += "The number is " + i + "<br>";  
}

[Try it Yourself »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_continue)

ADVERTISEMENT

## **JavaScript Labels**

To label JavaScript statements you precede the statements with a label name and a colon:

label:  
statements

The break and the continue statements are the only JavaScript statements that can "jump out of" a code block.

Syntax:

break labelname;  
  
continue labelname;

The continue statement (with or without a label reference) can only be used to **skip one loop iteration**.

The break statement, without a label reference, can only be used to **jump out of a loop or a switch**.

With a label reference, the break statement can be used to **jump out of any code block**:

### **Example**

const cars = ["BMW", "Volvo", "Saab", "Ford"];  
list: {  
  text += cars[0] + "<br>";  
  text += cars[1] + "<br>";  
  break list;  
  text += cars[2] + "<br>";  
  text += cars[3] + "<br>";  
}

[Try it Yourself »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_break_list)

# JavaScript Iterables

[❮ Previous](https://www.w3schools.com/js/js_break.asp)[Next ❯](https://www.w3schools.com/js/js_sets.asp)

Iterables are iterable objects (like Arrays).

Iterables can be accessed with simple and efficient code.

Iterables can be iterated over with for..of loops

## **The For Of Loop**

The JavaScript for..of statement loops through the elements of an iterable object.

### **Syntax**

for (variable of iterable) {  
  // *code block to be executed*  
}

## **Iterating**

Iterating is easy to understand.

It simply means looping over a sequence of elements.

Here are some easy examples:

* Iterating over a String
* Iterating over an Array

## **Iterating Over a String**

You can use a for..of loop to iterate over the elements of a string:

### **Example**

const name = "W3Schools";  
  
for (const x of name) {  
  // *code block to be executed*  
}

[Try it Yourself »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_iterate_string)

## **Iterating Over an Array**

You can use a for..of loop to iterate over the elements of an Array:

### **Example**

const letters = ["a","b","c"];  
  
for (const x of letters) {  
  // *code block to be executed*  
}

[Try it Yourself »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_iterate_array)

You can learn more details about Iterables in the chapter [JS Object Iterables](https://www.w3schools.com/js/js_object_iterables.asp).

ADVERTISEMENT

## **Iterating Over a Set**

You can use a for..of loop to iterate over the elements of a Set:

### **Example**

const letters = new Set(["a","b","c"]);  
  
for (const x of letters) {  
  // *code block to be executed*  
}

[Try it Yourself »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_iterate_set)

Sets and Maps are covered in the next chapters.

## **Iterating Over a Map**

You can use a for..of loop to iterate over the elements of a Map:

### **Example**

const fruits = new Map([  
  ["apples", 500],  
  ["bananas", 300],  
  ["oranges", 200]  
]);  
  
for (const x of fruits) {  
  // *code block to be executed*  
}