ECMAScript (ES) is a scripting language specification standardized by ECMAScript International. It is used by applications to enable client-side scripting. The specification is influenced by programming languages like Self, Perl, Python, Java etc. Languages like JavaScript, Jscript and ActionScript are governed by this specification.

This tutorial introduces you to ES6 implementation in JavaScript.

JavaScript

JavaScript was developed by Brendan Eich, a developer at Netscape Communications Corporation, in 1995.JavaScript started life with the name Mocha, and was briefly named LiveScript before being officially renamed to JavaScript. It is a scripting language that is executed by the browser, i.e. on the client’s end. It is used in conjunction with HTML to develop responsive webpages.

ECMA Script6’s implementation discussed here covers the following new features −

* Support for constants
* Block Scope
* Arrow Functions
* Extended Parameter Handling
* Template Literals
* Extended Literals
* Enhanced Object Properties
* De-structuring Assignment
* Modules
* Classes
* Iterators
* Generators
* Collections
* New built in methods for various classes
* Promises

## The Strict Mode

The fifth edition of the ECMAScript specification introduced the Strict Mode. The Strict Mode imposes a layer of constraint on JavaScript. It makes several changes to normal JavaScript semantics.

The code can be transitioned to work in the Strict Mode by including the following −

// Whole-script strict mode syntax

"use strict";

v = "Hi! I'm a strict mode script!"; // ERROR: Variable v is not declared

In the above snippet, the entire code runs as a constrained variant of JavaScript.

JavaScript also allows to restrict, the Strict Mode within a block’s scope as that of a function. This is illustrated as follows −

v = 15

function f1() {

"use strict";

var v = "Hi! I'm a strict mode script!";

}

In, the above snippet, any code outside the function will run in the non-strict mode. All statements within the function will be executed in the Strict Mode.

The for loop executes the code block for a specified number of times. It can be used to iterate over a fixed set of values, such as an array. Following is the syntax of the for loop.

var num = 5

var factorial = 1;

for( let i = num ; i >= 1; i-- ) {

factorial \*= i ;

}

console.log(factorial);

The for loop has three parts: the initializer (i = num), the condition ( i>=1) and the final expression (i--).

The program calculates the factorial of the number 5 and displays the same. The for loop generates the sequence of numbers from 5 to 1, calculating the product of the numbers in every iteration.

Multiple assignments and final expressions can be combined in a for loop, by using the comma operator (,). For example, the following for loop prints the first eight Fibonacci numbers −

"use strict"

for(let temp, i = 0, j = 1; j<30; temp = i, i = j, j = i + temp)

console.log(j);

Features of an Array

* An array declaration allocates sequential memory blocks.
* Arrays are static. This means that an array once initialized cannot be resized.
* Each memory block represents an array element.
* Array elements are identified by a unique integer called as the subscript/index of the element.
* Arrays too, like variables, should be declared before they are used.
* Array initialization refers to populating the array elements.
* Array element values can be updated or modified but cannot be deleted.

Declaring and Initializing Arrays

To declare and initialize an array in JavaScript use the following syntax −

var array\_name; //declaration

array\_name = [val1,val2,valn..] //initialization

OR

var array\_name = [val1,val2…valn]

**Note** − The pair of [] is called the dimension of the array.

For example, a declaration like: **var numlist = [2,4,6,8]** will create an array as shown in the following figure.

var alphas;

alphas = ["1","2","3","4"]

console.log(alphas[0]);

console.log(alphas[1]);

The following output is displayed on successful execution of the above code.

1

2

### Example: Single Statement Declaration and Initialization

var nums = [1,2,3,3]

console.log(nums[0]);

console.log(nums[1]);

console.log(nums[2]);

console.log(nums[3]);

The following output is displayed on successful execution of the above code.

1

2

3

3

## Array Object

An array can also be created using the Array object. The Array constructor can be passed as −

* A numeric value that represents the size of the array or.
* A list of comma separated values.

The following Examples create an array using this method.

## Example

var arr\_names = new Array(4)

for(var i = 0;i<arr\_names.length;i++) {

arr\_names[i] = i \* 2

console.log(arr\_names[i])

}

The following output is displayed on successful execution of the above code.

0

2

4

6

### Example: Array Constructor Accepts Comma-separated Values

var names = new Array("Mary","Tom","Jack","Jill")

for(var i = 0;i<names.length;i++) {

console.log(names[i])

}

The following output is displayed on successful execution of the above code.

Mary

Tom

Jack

Jill