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.

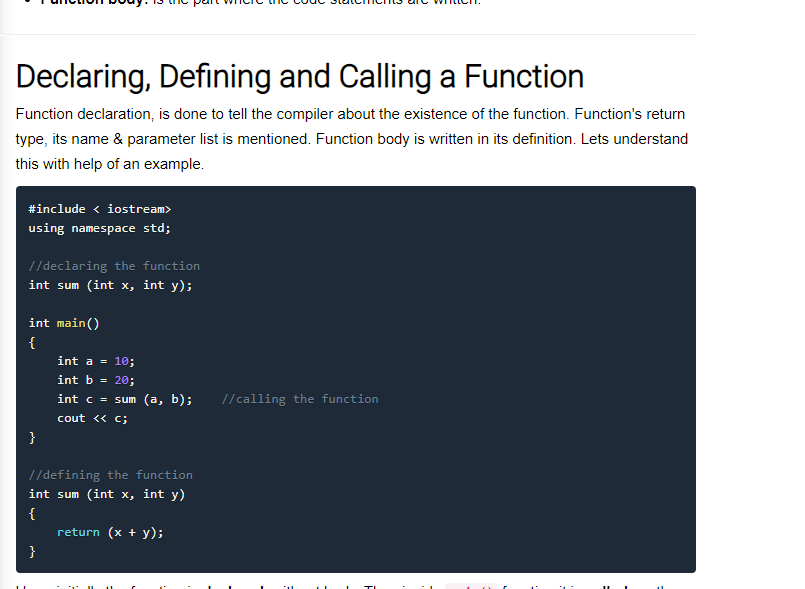
ECMA Script6’s implementation discussed here covers the following new features −

* 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

1.ES 6 supports hoisting

Declaring the function at top

Variable hoisting allows the use of a variable in a JavaScript program, even before it is declared



2.The Let and Block Scope

Let and Var

>>The **var** keyword assigns a **function scope** to the variable.

>>The **let** keyword allows the script to restrict access to the variable to the nearest enclosing **block scope**.

>> **Constants** are **block-scoped**, much like variables defined using the let statement. The value of a constant cannot change through re-assignment, and it can't be re-declared,**immutable.**

# **3. ES6 - The for…in loop**

The for...in loop is used to loop through an object's properties.

var x=function() {

var names=[{id:1,"name":"abc"},{id:2,"name":"ab"},{id:3,"name":"ac"}]

for(var key in names){

console.log("name=>"+JSON.stringify(key)); //shows indexes

console.log("name=>"+JSON.stringify(names[key])); //shows obj values

}

}();

**4.for of**

var x=function() {

var names=[{id:1,"name":"abc"},{id:2,"name":"ab"},{id:3,"name":"ac"}]

for(var key of names){

console.log("name=>"+JSON.stringify(key)); //shows obj values

console.log("name=>"+JSON.stringify(names[key])); //undef

}

}();

**5.Rest para**

 rest parameters act as placeholders for multiple arguments of the same type.

To declare a rest parameter, the parameter name is prefixed with three periods, known as the **spread operator**.

//...rest parameters

var lnm=["a","b","c"]

var restPara=function(a,b,...names){ //default paras will not work with rest paras

console.log("a=>"+a)

console.log("b=>"+b)

console.log("name=>"+names);

}(1,0,lnm);

# **6.ES6 - String Method slice()**

This method extracts a section of a string and returns a new string.

## Syntax

string.slice( beginslice [, endSlice] );

## Argument Details

* **beginSlice** − The zero-based index at which to begin extraction.
* **endSlice** − The zero-based index at which to end extraction. If omitted, slice extracts to the end of the string.

## Return Value

If successful, **slice** returns the index of the regular expression inside the string. Otherwise, it returns -1.

## Example

var str = "Apples are round, and apples are juicy.";

var sliced = str.slice(3, -2);

console.log(sliced);

## Output

les are round, and apples are juic

//5.string methods: slice()

var show=function(){

var str="qqqqabdbdbdbdooo";

var result=str.slice(0,-2);

console.log("result=>"+result); //result=>qqqqabdbdbdbdo

}();

# 7. **String Method split()**

This method splits a String object into an array of strings by separating the string into substrings.

## Syntax

string.split([separator][, limit]);

## Argument Details

* **separator** − Specifies the character to use for separating the string. Ifseparator is omitted, the array returned contains one element consisting of the entire string.
* **limit** − Integer specifying a limit on the number of splits to be found.

## Return Value

The split method returns the new array. Also, when the string is empty, split returns an array containing one empty string, rather than an empty array.

## Example

var str = "Apples are round, and apples are juicy.";

var splitted = str.split(" ", 3);

console.log(splitted)

## Output

[ 'Apples', 'are', 'round,' ]

var show=function(){

var str="qqqqabdbdbdbdooo sdfrf werfewrf wrwr rwrwr wrwqr";

var l=str.length;

console.log("length=>"+l)

var result=str.split(" ",l);

console.log("result=>"+JSON.stringify(result));

//length=>48

//result=>["qqqqabdbdbdbdooo","sdfrf","werfewrf","wrwr","rwrwr","wrwqr"]

}();

8.Filter

//7.filter

var names=["abc","abcd","abcd","aa","bb","absdf","sss"];

var show=function(val,query){

//console.log("values=>"+val);

return val.filter(function(el){

// console.log("el=>"+el);

//console.log("query=>"+query);

// var output=el.indexOf(query.toLowerCase())!==-1;

//console.log("result=>"+(output)); //return type is boolean

return el.toLowerCase().indexOf(query.toLowerCase()) !== -1; //return values

});

}

//}(names,"ab"); //not work

console.log("show=>"+show(names,"ab"));

console.log("show 2=>"+show(names,"s"));

// var fruits = ['apple', 'banana', 'grapes', 'mango', 'orange'];

// /\*\*

// \* Filter array items based on search criteria (query)

// \*/

// var filterItems=function (arr, query) {

// return arr.filter(function(el) {

// return el.toLowerCase().indexOf(query.toLowerCase()) !== -1;

// })

// };

// console.log(filterItems(fruits, 'ap')); // ['apple', 'grapes']

// console.log(filterItems(fruits, 'an')); // ['banana', 'mango', 'orange']

9. The **forEach()** method executes a provided function once for each array element.

//8.foreach()

var fruits = ['apple', 'banana', 'grapes', 'mango', 'orange'];

var num=[1,2,3,1];

var show=function(val){

console.log("type=>"+typeof(val));

return val.forEach(element => {

console.log("result=>"+(element+10));

});

}(num);

/\*

type=>object

result=>11

result=>12

result=>13

result=>11

\*/

10. The **map()** method **creates a new array** with the results of calling a provided function on every element in the calling array.

//9. map()

var num=[1,2,3,1];

var show=function(val){

//console.log("val=>"+val);

// val.forEach(function(val){

// if(val === Number){console.l}

// })

return val.map(function(value){

return value+100;

})

}(num);

console.log("values==>"+show);