JavaScript

Introduction –

JavaScript is –

- ✓ the world's most popular programming language
- ✓ the programming language of the Web.
- ✓ easy to learn.

Complete JavaScript Reference - https://www.w3schools.com/jsref/default.asp

Where to write JavaScript –

- ✓ **JavaScript code is inserted between <script> and </script> tags.**
- ✓ **Scripts can be placed in the <body>, or in the <head> section of an HTML page, or in both.**
- ✓ **Scripts can also be placed in external files**
- ✓ **JavaScript files have the file extension .js**
- ✓ **To use an external script - <script src="myScript.js"></script>**

JavaScript Output – (Demo)

- ✓ **using innerHTML - will see in details after HTML DOM**
- ✓ **document.write()**
- ✓ **window.alert()**
- ✓ **console.log()**

JavaScript Literals -

- ✓ **Numbers - 10 , 23.5**
- ✓ **String –**
  - o **'Gopal Gupta'**
  - o **"Gopal Gupta"**

JavaScript Variables –

- ✓ **JavaScript uses the keywords var, let and const to declare variables.**
- ✓ **var keyword was used in all JavaScript code from 1995 to 2015.**
- ✓ **let and const keywords were added to JavaScript in 2015.**
- ✓ **var keyword should only be used in code written for older browsers.**

When to Use let, or const?

- ✓ **use const if the value should not be changed**
- ✓ **use const if the type should not be changed (Arrays and Objects)**
- ✓ **use let if you can't use const**

JavaScript Identifiers / Names - Identifiers are JavaScript names.

JavaScript name must begin with:

- ✓ A letter (A-Z or a-z)
- ✓ A dollar sign ($)
- ✓ Or an underscore (_)
- ✓ JavaScript identifiers are case sensitive.

JavaScript Comments –

- ✓ **Single line comments start with //**
- ✓ **Multi-line comments start with /* and end with */.**

Declaring a JavaScript Variable –

```
let a;

const b;
```

After the declaration, the variable has no value (technically it is undefined).

```
let a=10;

const b = 20;

let a=10,b=20,c=30;
```

JavaScript Operators -

- Arithmetic Operators (+, -, *, **, /, %, ++, --)
- Assignment Operators (=, +=, *=, -=, /=, etc)
- Comparison Operators (==, ===, !=, >, <, >=, <=, ?)
- String Operators (all comparison operator, +,
- Logical Operators (&&, ||, !)
- Bitwise Operators (&,|,~,^,<<,>>,>>>)
- Ternary Operators – ternary operator
- Type Operators – return the type of a variable.

JavaScript Data Types –

JavaScript has 8 Datatypes

1. **String**
2. **Number**
3. **Bigint**
4. **Boolean**
5. **Undefined**
6. **Null**
7. **Symbol**
8. **Object**

```
// Strings:
let color = "Yellow";
// Numbers:
let length = 16;
// Booleans
let x = true;
//BigInt
// All JavaScript numbers are stored in a a 64-bit
// floating-point format.
//JavaScript BigInt is a new datatype (ES2020)
let x = BigInt("12345678901234567890123456789O");
```

Note –

- When adding a number and a string, JavaScript will treat the number as a string.
- JavaScript evaluates expressions from left to right.
- Different sequences can produce different results:
  - let x = 16 + 4 + "Volvo"; gives output → 20Volvo
  - let x = "Volvo" + 16 + 4; gives output → Volvo164

Undefined - a variable without a value, has the value undefined. The type is also undefined.

JavaScript Functions

- ✓ JavaScript functions are defined with the function keyword.
- ✓ A JavaScript function is a block of code designed to perform a particular task.

```
function name(parameter1, parameter2, parameter3) {
// code to be executed
}
```

Example - write a function that takes two number as arguments and return sum of both.

```
function Addition (a, b) {
    return a+b;
}

console.log(Addition (10,20));
```

Function Expressions –

A JavaScript function can also be defined using an expression.

```
const x = function (a,b) {return a+b;}
console.log(x(2,3));
```

Note - The function above is actually an anonymous function (a function without a name).

Self-Invoking Functions –

- ✓ A self-invoking expression is invoked (started) automatically, without being called.
- ✓ Function expressions will execute automatically if the expression is followed by ().
- ✓ You cannot self-invoke a function declaration.
- ✓ You have to add parentheses around the function to indicate that it is a function expression:

```
// Self-Invoking Functions
(function(a,b){
    console.log(a+b);
})(10,20);
```

Functions are Objects –

- ✓ JavaScript functions can best be described as objects.
- ✓ JavaScript functions have both properties and methods.
- ✓ **The arguments.length** property returns the number of arguments received when the function was invoked.
- ✓ The toString() method returns the function as a string

```
// Functions are Objects
function display(a,b,c){
    console.log(arguments.length)
}
display(12,23,34);
```

Arrow Functions – Arrow functions allows a short syntax for writing function expressions.

Function Expression

```
let x = function(a,b){
        return a+b;
}
```

Arrow Function

```
const x = (a,b) => {return a+b};
```

Note –

- ✓ **You don't need the function keyword**
- ✓ **Arrow functions do not have their own this**
- ✓ **They are not well suited for defining object methods.**

Function Parameters –

- ✓ **Default Parameters –** If a function is called with missing arguments (less than declared)

```
function abc(p,r,t=1){
    return (p*r*t)/100;
}
// 3rd arguments is not passed
console.log(abc(1000,10));
// 3rd argument is passed
console.log(abc(1000,10,3));

output –
100
300
```

- ✓ **Rest Parameter –** n number of arguments

```
// Rest paramenter
function nargs(...n){
    console.log(n);
}

nargs(12,34,45,56,67);

Output – argument as array.
[ 12, 34, 45, 56, 67 ]
```

Arguments are Passed by Value –

```
// Arguments are Passed by Value
function f3(){
    let x=10;
    f4(x);
    console.log(x);
}
function f4(x){
    x=200;
}
f3();
Output - 10
```

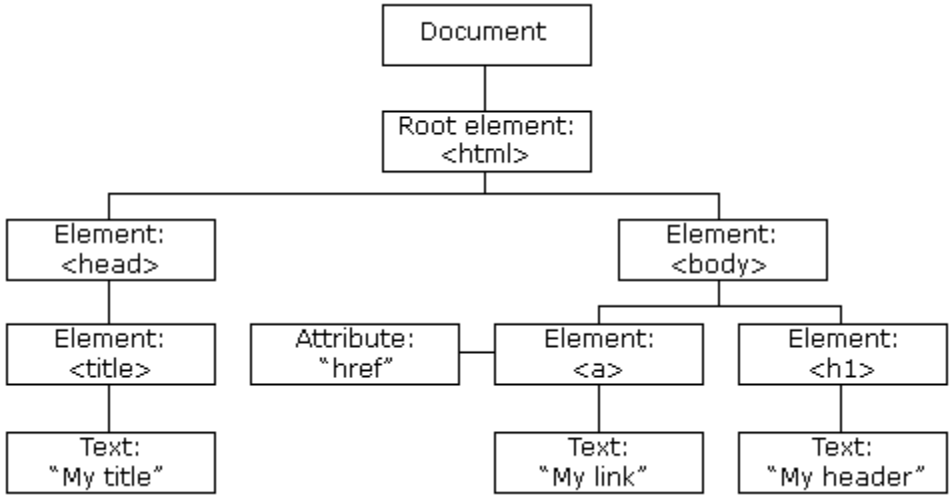
Objects are Passed by Reference

```
//Objects are Passed by Reference
function f1(){
    const arr = [12,23,34,45];
    f2(arr);
    console.log(arr);
}
function f2(arr){
    arr[0]=100;
}
f1();
Output –
[ 100, 23, 34, 45 ]
```

JavaScript HTML DOM

When a web page is loaded, the browser creates a **D**ocument **O**bject **M**odel of the page.

The **HTML DOM** model is constructed as a tree of **Objects**:



With the object model, JavaScript gets all the power it needs to create dynamic HTML:

- JavaScript can change
  - all the HTML elements in the page
  - all the HTML attributes in the page
  - all the CSS styles in the page
- JavaScript can remove existing HTML elements and attributes
- JavaScript can add new HTML elements and attributes
- JavaScript can react to all existing HTML events in the page
- JavaScript can create new HTML events in the page

What is the HTML DOM?

- The HTML DOM is a standard for how to get, change, add, or delete HTML elements.
- The HTML DOM can be accessed with JavaScript (and with other programming languages).
- In the DOM, all HTML elements are defined as objects.

HTML DOM Methods

Finding HTML Elements

Method	Description
document.getElementById(*id*)	Find an element by element id
document.getElementsByTagName(*name*)	Find elements by tag name
document.getElementsByClassName(*name*)	Find elements by class name

Changing HTML Elements

Property	Description
element*.innerHTML = *new html content	Change the inner HTML of an element
element.attribute = new value	Change the attribute value of an HTML element
element*.style.*property = new style	Change the style of an HTML element
Method	**Description**
element*.setAttribute(*attribute, value)	Change the attribute value of an HTML element

Adding and Deleting Elements

Method	Description
document.createElement(*element*)	Create an HTML element
document.removeChild(*element*)	Remove an HTML element
document.appendChild(*element*)	Add an HTML element
document.replaceChild(*new, old*)	Replace an HTML element
document.write(*text*)	Write into the HTML output stream

Finding HTML Elements

Demo

Changing HTML Content

Demo

Changing the Value of an Attribute

Demo

document.write() - can be used to write directly to the HTML output stream

JavaScript HTML DOM - Changing CSS - The HTML DOM allows JavaScript to change the style of HTML elements

`document.getElementById(id).style.property = new style`

Demo -

if, else, and else if

```
if (condition) {
    //block of code to be executed if the condition is true
  }
```

```
if (condition) {
    //  block of code to be executed if the condition is true
  } else {
    //  block of code to be executed if the condition is false
  }
```

```
if (condition1) {
    //  block of code to be executed if condition1 is true
  } else if (condition2) {
    //  block of code to be executed if the condition1 is false and condition2 is true
  } else {
    //  block of code to be executed if the condition1 is false and condition2 is false
  }
```

Loops

```
for (expression 1; expression 2; expression 3) {
    // code block to be executed
  }
```

For In Loop - The JavaScript for in statement loops through the properties of an Object:

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

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

```
const numbers = [45, 4, 9, 16, 25];
for(let x in numbers){
    console.log(x);
}
```

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

```
const numbers = [5, 4, 9, 6, 2];
```

```
function double1(n){
    console.log(n);
}
numbers.forEach(double1);
```

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.

```
const color = ["red","green","yellow"];
for(let c of color){
    console.log(c);
}
```

Break and Continue Statement - Same as Java, C and C++

JavaScript Objects

- ✓ Objects in JavaScript represent data in the form of key-value pairs.
- ✓ Each key-value pair is called property and each pair is separated by a colon.

Creating a JavaScript Object –

1. **Using Object Literal Syntax**

```
let student = {
    id:1,
    name:"Gopal",
    branch:"CSE"
};
```

2. **Using Object Constructor –**

```
let obj = new Object();
obj.id=1;
obj.name="Gopal";
obj.branch = "CSE";
console.log(obj);
```

3. **Using the Object.create Method –**

This method is used when we wish to inherit properties from an existing object while creating a new object.

```
let obj={
    id:1,
    name:"gopal"
};
let new_obj = Object.create(obj);
console.log(new_obj.name);
```

4. **Using Object.assign Method –**

This method is used when we wish to include properties from multiple other objects into the new object we are creating. Before seeing an example, let's know its syntax:

```
Object.assign(targetObject, sourceObject1, sourseObject2)
```

```
let obj1 = {
    id:1,
    name:"gopal"
};
let obj2 = {
```

```
    branch:"CSE"
};
let new_obj =
Object.assign({},obj1,obj2);
console.log(new_obj.branch);
```

Nested Objects – Values in an object can be another object.

```
let obj = {
    id:1,
    name:"gopal",
    address:{
        house_no:123,
        street_name:"vijay nagar",
        city:"Ghaziabad"
    }
};
console.log(obj);
```

access nested objects using the dot notation

```
let obj = {
    id:1,
    name:"gopal",
    address:{
        house_no:123,
        street_name:"vijay nagar",
```

```
        city:"Ghaziabad"
    }
};
console.log(obj.address.city);
```

JavaScript Object Methods

```
let obj = {
    id:123,
    name:"Gopal",
    branch:"CSE",
    info:function(){
        return "Welcome Mr./Ms. " + this.name + " your branch is " + this.branch;
    }
}
console.log(obj.info());
```

What is this? - In JavaScript, this keyword refers to an object.

Loop over Object

```
const student = {
  id: 1,
  name: "Gopal",
};

for (let key in student)
console.log(student[key]);
```

JavaScript Events

- ✓ The change in the state of an object is known as an Event.
- ✓ Here are some examples of HTML events:
  - An HTML web page has finished loading
  - An HTML input field was changed
  - An HTML button was clicked
- ✓ When JavaScript is used in HTML pages, JavaScript can "react" on these events.
- ✓ This process of reacting over the events is called Event Handling and js handles the HTML events via Event Handlers.

Example –

```
<button onclick="document.getElementById('demo').innerHTML = Date()">The time is?</button>
```

Mouse events:

Event Performed	Event Handler	Description
click	onclick	When mouse click on an element
mouseover	onmouseover	When the cursor of the mouse comes over the element
mouseout	onmouseout	When the cursor of the mouse leaves an element
mousedown	onmousedown	When the mouse button is pressed over the element
mouseup	onmouseup	When the mouse button is released over the element
mousemove	onmousemove	When the mouse movement takes place.

Form events:

Event Performed	Event Handler	Description
focus	onfocus	When the user focuses on an element
submit	onsubmit	When the user submits the form
blur	onblur	When the focus is away from a form element
change	onchange	When the user modifies or changes the value of a form element

Window/Document events

Event Performed	Event Handler	Description
load	onload	When the browser finishes the loading of the page
unload	onunload	When the visitor leaves the current webpage, the browser unloads it. (Due to different browser settings, this event may not always work as expected.)
resize	onresize	When the visitor resizes the window of the browser

Demo -

JavaScript String

- ✓ A JavaScript string is zero or more characters written inside quotes.
- ✓ You can use single or double quotes:
  - let collegeName1 = "ABESIT"; // Double quotes
  - let collegeName2 = 'ABESIT'; // Single quotes

String Methods

Methods	Description
charAt()	It provides the char value present at the specified index.

charCodeAt()	It provides the Unicode value of a character present at the specified index.
concat()	It provides a combination of two or more strings.
indexOf()	It provides the position of a char value present in the given string.
lastIndexOf()	It provides the position of a char value present in the given string by searching a character from the last position.
search()	It searches a specified regular expression in a given string and returns its position if a match occurs.
match()	It searches a specified regular expression in a given string and returns that regular expression if a match occurs.
replace()	It replaces a given string with the specified replacement.
substring()	It is used to fetch the part of the given string on the basis of the specified index.
slice()	It is used to fetch the part of the given string. It allows us to assign positive as well negative index.
toLowerCase()	It converts the given string into lowercase letter.
toUpperCase()	It converts the given string into uppercase letter.
toString()	It provides a string representing the particular object.
valueOf()	It provides the primitive value of string object.
split()	It splits a string into substring array, then returns that newly created array.
trim()	It trims the white space from the left and right side of the string.
includes()	includes() method returns true if a string contains a specified value. Otherwise it returns false.
startsWith()	startsWith() method returns true if a string begins with a specified value. Otherwise it returns false
endsWith()	endsWith() method returns true if a string ends with a specified value. Otherwise it returns false

```
let collegeName = "abesit Engineering College";
console.log(collegeName.length);
console.log(collegeName.charAt(1));
```

```
console.log(collegeName.charCodeAt(0));
console.log(collegeName.indexOf('i'));
console.log(collegeName.substring(7,18));
console.log(collegeName.split(' '));
console.log(collegeName.toLowerCase());
console.log(collegeName.toUpperCase());
```

```
output –
26
b
97
4
Engineering
[ 'abesit', 'Engineering', 'College' ]
PS D:\ABESIT\2022-23\Even Sem\Summer
Training\JavaScript> node "d:\ABESIT\2022-23\Even
Sem\Summer Training\JavaScript\string1.js"
26
b
97
4
Engineering
[ 'abesit', 'Engineering', 'College' ]
abesit engineering college
ABESIT ENGINEERING COLLEGE
```

String search() Method - search() method is used to search the regular expression in the given string. This method returns -1, if match is not found.

```
let para = "I like Javascript. I also like Java";
console.log(para.search('Java'));
console.log(para.search('like'));
```

String replace() Method

- ✓ replace() method is used to replace a part of a given string with a new substring.
- ✓ It returns the new string with the specified replacement.

```
let para = "I like Javascript. I also like Java";
console.log(para.replace('Javascript','Python'));
```

Interpolation

Template literals provide an easy way to interpolate variables and expressions into strings.

The method is called string interpolation.

Variable Substitutions –

```
let a = 1,b=2;
let st = `sum of ${a} and ${b} = ${a+b}`;
console.log(st);
```

Note – Templet literal represent using ``.

Arrays –

- ✓ JavaScript array is an object that represents a collection of similar type of elements.

Creating an Array using literal–

```
let arr = ['abesit','engg','gzb'];
console.log(arr);
```

Creating an Array using new keyword –

```
let arr1 = new Array();
arr1[0]=10;
arr1[1]=20;
console.log(arr1);
```

Note – we can also add empty items.

```
let arr1 = new Array();
arr1[0]=10;
arr1[1]=20;
arr1[4]=30;
console.log(arr1);

[ 10, 20, <2 empty items>, 30 ]
```

Array Methods

Methods	Description
concat()	It returns a new array object that contains two or more merged arrays.

entries()	It creates an iterator object and a loop that iterates over each key/value pair.
every()	It determines whether all the elements of an array are satisfying the provided function conditions.
flatMap()	It maps all array elements via mapping function, then flattens the result into a new array.
fill()	It fills elements into an array with static values.
from()	It creates a new array carrying the exact copy of another array element.
filter()	It returns the new array containing the elements that pass the provided function conditions.
find()	It returns the value of the first element in the given array that satisfies the specified condition.
findIndex()	It returns the index value of the first element in the given array that satisfies the specified condition.
forEach()	It invokes the provided function once for each element of an array.
includes()	It checks whether the given array contains the specified element.
indexOf()	It searches the specified element in the given array and returns the index of the first match.
join()	It joins the elements of an array as a string.
lastIndexOf()	It searches the specified element in the given array and returns the index of the last match.
map()	It calls the specified function for every array element and returns the new array
pop()	It removes and returns the last element of an array.
push()	It adds one or more elements to the end of an array.
reverse()	It reverses the elements of given array.
shift()	It removes and returns the first element of an array.
unshift()	It adds one or more elements in the beginning of the given array.
slice()	It returns a new array containing the copy of the part of the given array.
sort()	It returns the element of the given array in a sorted order. sort() function sorts values as strings.
toString()	It converts the elements of a specified array into string form, without affecting the original array.

Demo -

```
let arr = [10,20,30,40,50,60];
console.log(arr.concat([1,2,3]));
console.log(arr.every(function (n){return n<80}));
//or
console.log(arr.every(n=>n<80));
console.log(arr.flatMap(x=>x*2));
console.log(arr.fill(0,2,4));//arr.fill(value[, start[, end]])
console.log(Array.from(arr));
console.log(arr.filter(x=>x>30));
console.log(arr.find(x => x>30));
console.log(arr.findIndex(x => x>30));
arr = [10,20,10,40,20,60];
console.log(arr.join(','));
console.log(arr.includes(30));
console.log(arr.lastIndexOf(20));
console.log(arr.map(x => x*2));
console.log(arr.pop());
arr.push(100);
console.log(arr);
console.log(arr.sort((a,b) => a-b));
```

Iteration –

```
let arr = ['apple','mango','orange'];
for(let i=0;i<arr.length;i++){
```

```
    console.log(arr[i]);
}

for(i in arr){ // for(key in object)
    console.log(arr[i]);
}
console.log("\n");
arr.forEach(display)
function display(x){
    console.log(x);
}
console.log("\n");
arr.map(display);

console.log("\n");
for(const x of arr)
    console.log(x);
```

The typeof Operator

typeof operator is used to find the data type of a JavaScript variable.

```
console.log(typeof(12));
console.log(typeof('12'));
console.log(typeof(12.3));
console.log(typeof([1,2,3,4]));
console.log(typeof("abesit"));
console.log(typeof(true));
```

```
console.log(typeof(x));
console.log(typeof({1:'abesit'}));
console.log(12/'abesit');
```

number

string

number

object

string

boolean

undefined

object

NaN

JavaScript Type Conversion

Number Methods -

Method	Description
Number()	Returns a number, converted from its argument
parseFloat()	Parses a string and returns a floating point number
parseInt()	Parses a string and returns an integer

Numbers to Strings

String(x) // returns a string from a number variable x

Automatic Type Conversion -

```
console.log(2 + Number('23'));

console.log(2 + parseFloat('23.555'));
```

```
console.log(2 + parseInt('23.555'));
console.log(String(23) + 12);
console.log(5 + null);    // returns 5         because null is converted
to 0
console.log("5" + null);  // returns "5null"   because null is converted
to "null"
console.log("5" + 2);     // returns "52"      because 2 is converted to
"2"
console.log("5" - 2);     // returns 3         because "5" is converted to
5
console.log("5" * "2");   // returns 10        because "5" and "2" are
converted to 5 and 2
```

JavaScript Sets

- A JavaScript Set is a collection of unique values.
- Each value can only occur once in a Set.
- A Set can hold any value of any data type.

How to Create a Set

```
let s = new Set([1,2,3,4,4,3]);
console.log(s);
//or
let s1 = new Set();
console.log(s1);
```

Set Methods

Method	Description
new Set()	Creates a new Set
add()	Adds a new element to the Set
delete()	Removes an element from a Set
has()	Returns true if a value exists

clear()	Removes all elements from a Set
forEach()	Invokes a callback for each element
values()	Returns an Iterator with all the values in a Set
keys()	Same as values()
entries()	Returns an Iterator with the [value,value] pairs from a Set

```
let s = new Set();
s.add(20);
s.add(30);
s.add(40);
s.add(50);
console.log(s);
s.delete(30);
console.log(s);
console.log(s.has(40));
console.log(s.values());
console.log(s.entries());
```

forEach method - The forEach() method invokes a function for each Set element

```
let s = new Set(['a','b',"Gopal",1]);
s.forEach(function(value){
    console.log(value);
});
```

JavaScript Maps

A Map holds key-value pairs where the keys can be any datatype.

Map Methods

Method	Description
new Map()	Creates a new Map object
set()	Sets the value for a key in a Map
get()	Gets the value for a key in a Map
clear()	Removes all the elements from a Map
delete()	Removes a Map element specified by a key
has()	Returns true if a key exists in a Map
forEach()	Invokes a callback for each key/value pair in a Map

entries()	Returns an iterator object with the [key, value] pairs in a Map
keys()	Returns an iterator object with the keys in a Map
values()	Returns an iterator object of the values in a Map
size	Returns the number of Map elements

Create a Map

```
let m = new Map([
                    ["id",102],
                    ["Name","Gopal"],
                    ["Branch","CSE"]
                ]);
console.log(m);
```

```
let m = new Map(); // Create an empty Map
m.set("roll",101); // Add key:value
m.set("name","rahul");
console.log(m);
```

```
let m = new Map([
    ["id",102],
    ["Name","Gopal"],
```

```
    ["Branch","CSE"],
    ["address","Ghaziabad"]
]);
console.log(m.size);
console.log(m.get("Name"));
console.log(m.has("id"));
m.delete("id");
console.log(m);
console.log(m.values());
console.log(m.keys());
console.log(m.entries());

Output –
4
Gopal
true
Map(3) {
  'Name' => 'Gopal',
  'Branch' => 'CSE',
  'address' => 'Ghaziabad'
}
[Map Iterator] { 'Gopal', 'CSE',
'Ghaziabad' }
[Map Iterator] { 'Name', 'Branch',
'address' }
[Map Entries] {
  [ 'Name', 'Gopal' ],
```

```
  [ 'Branch', 'CSE' ],
  [ 'address', 'Ghaziabad' ]
}
```

Map.forEach()

```
let m = new Map([
    ["id",102],
    ["Name","Gopal"],
    ["Branch","CSE"],
    ["address","Ghaziabad"]
]);

m.forEach(function(v,k){
    console.log("Value of key "+ k + " is " + v);
});
//using arrow function
m.forEach((v,k) => console.log("Value of key "+ k + " is " + v));
```

Loop over Map

```
let m = new Map([
    ["id",102],
    ["Name","Gopal"],
    ["Branch","CSE"],
    ["address","Ghaziabad"]
]);
```

```
for(kv of m){
    console.log(kv);
}
for(value of m.values())
    console.log(value);
for(key of m.keys())
    console.log(key);
```

JavaScript Callbacks –

- A callback is a function passed as an argument to another function
- This technique allows a function to call another function
- A callback function can run after another function has finished

When to Use a Callback?

- The examples above are not very exciting.
- They are simplified to teach you the callback syntax.
- Where callbacks really shine are in **asynchronous functions**, where one function has to wait for another function

Asynchronous JavaScript

- Functions running in parallel with other functions are called asynchronous
- A good example is JavaScript setTimeout()

```
setTimeout(myFunction, 3000);

function myFunction() {
  document.getElementById("demo").innerHTML = "I love You !!";
}
```

Waiting for Intervals:

When using the JavaScript function setInterval(), you can specify a callback function to be executed for each interval:

Javascript ES6

- The let keyword
- The const keyword
- Arrow Functions
- The ... Operator
- For/of
- Map Objects
- Set Objects
- Classes
- Promises
- Symbol
- Default Parameters
- Function Rest Parameter
- String.includes()
- String.startsWith()
- String.endsWith()
- Array.from()
- Array keys()
- Array find()
- Array findIndex()
- New Math Methods

- New Number Properties
- New Number Methods
- New Global Methods
- Object entries
- JavaScript Modules

Classes

- JavaScript Classes are templates for JavaScript Objects.
- Use the keyword class to create a class.
- Always add a method named constructor()