**History Of JavaScript/ECMA script/livescript**

JavaScript was **invented by Brendan Eich in 1995**. It was developed for Netscape 2, and became the ECMA-262 standard in 1997. After Netscape handed JavaScript over to ECMA, the Mozilla foundation continued to develop JavaScript for the Firefox browser. Mozilla's latest version was 1.8

"JavaScript" is a trademark of **Oracle Corporation** in the United States.

JavaScript is a cross-platform, object-oriented scripting language used to make webpages interactive (e.g., having complex animations, clickable buttons, popup menus, etc.). There are also more advanced server side versions of JavaScript such as Node.js, which allow you to add more functionality to a website than downloading files (such as realtime collaboration between multiple computers). Inside a host environment (for example, a web browser), JavaScript can be connected to the objects of its environment to provide programmatic control over them.

This means that in the browser, JavaScript can change the way the webpage (DOM) looks. And, likewise, Node.js JavaScript on the server can respond to custom requests sent by code executed in the browser.

**What is binary and assembly level language?**

The major difference between machine language and assembly language is that **machine language is referred to as a binary language**. It can be run on a computer directly. While an assembly language is a low-level programming language that must be converted into machine code using software called an assembler.

The most commonly used assembly languages include ARM, MIPS, and x86.

**Translators**

Translators are used to translate assembly level and high level language into Machinelevel language.

* Assemblers.
* Interpreter .
* Compiler.

**How JavaScript is executed in Browsers without any interpreter?**

Since, In every browser there will be a inbuilt JavaScript Engine we can JavaScript in browsers

Ex: Chrome -V 8, InternetExplorer-chatra.

**Important Points:**

JS is a CaseSensitive language.

JS is a looselytyped language.

JS is a scripting language.

JS is a Interpreted language.

JS is a ObjectOriented/ObjectBased Programming language.

**How to declare a variable in JS?**

We can declare variable by using keywords like var,let,const.

**var:**

if we are using var we can redeclare and reinitialize the variable.

Ex: var a=10 **✓**

var a= 20 **✓** ☓

**let:**

if we are using let we cannot redeclare but we can reinitialize the variable.

Ex: let a=10 **✓**

let a= 20 ☓

a=30 **✓**

**const:**

if we are using const we cannot redeclare /reinitialize the variable.

Ex: let a=10 **✓**

let a= 20 ☓

a=30 ☓

**DataTypes In JavaScript:**

**Primitive datatype:**

String: It is one of the datatype We can specify by using ‘’ /” ” .

Boolean: true/false.

Number:

Undefined: whenever we didn’t declare /initialize the variable.

Null:

**Non Primitive datatype:**

Arrays:

Object:

Functions:

**Example**

var a=" Nithish"

var a="jagadeesh"//we can redeclare and reintialize

let b=56

    b=66 // we cannot declare again but we can reintialize

const c=true // we cannot reintialize and redeclare

var e=null

console.log(typeof(a)) //this will reflect in console(developer side)//String

document.write("nithish")//this will reflect on webpages(client side)

console.log(typeof(b))//number

console.log(typeof(c))//boolean

console.log(typeof(d))//undefined

console.log(typeof(e))//object

**Functions in JavaScript:**

**typeof(variable/data):** it will return the datatype.

Ex:var a=”nithi”

Console.log( typeof(a)) //string

**How to change the datatype explicitely?**

var a=10

console.log(typeof(String(a))) //string

**Difference between document.write and console.log**

**document.write:** it is reflected in webpage(client side).

**console.log:** it is reflected in a console window(developer side).

**Inline Embedded javascript v/s external javascript.**

JavaScript code that is embedded with an HTML document is referred to as inline JavaScript. On the other hand, the HTML document may refer to a separate file that contains the JavaScript program(.js file), in which case codit is referred to as external JavaScript.

**Internal execution:**

<html lang="en">

<head>

    <title>Nithish</title>

</head>

<body>

    <Script>document.write("kk")</Script>

</body>

</html>

**External execution:**

<html lang="en">

<head>

    <title>Nithish</title>

</head>

<body>

    <Script src="./Demo.js"></Script>

</body>

</html>

**Demo.js (External Javascript file)**

var a=" ";

console.log(typeof(a))

document.write("nithish")

**JavaScript libraries:**

Libraries are developed by JS developers which contains reusable functons and reusable code .

By using JS libraries we can achieve codeOptimization.

**Ex:**Bootsrap,JQuery,Load# etc..,

**Javascript Frameworks:**

* **Node JS** ------ for developing webapplication.
* **React JS** ------ for developing webapplication.
* **React Native** ------ for developing client- server application.
* **Angular JS** ------ for developing single page web application(gmail,map).
* **Electron JS** ------ for developing stand-alone application.
* **Tensor flow** ----- for developing Artificial intelligence and machine learning application.

**Conditional statements**

1. if .
2. if else.
3. else if ladder.
4. switch.

**1.if statement:** whenever the given condition is true then,if block will execute.

var pageTitle="Doctors-Dashboard"

if(pageTitle==="Doctors-Dashboard")

console.log("Dashboard page is displayed")

**2.if else statement:** whenever the given condition is true then,if block will execute otherwise else block will executes.

var pageTitle="Doctors-Dashboard"

if(pageTitle==="Doctors-Dashboard")

console.log("Dashboard page is displayed")

else

console.log("Dashboard page is not displayed")

**3.else if statement:** whenever the given condition is true then, that respective block will executes otherwise else block will executes.

var pageTitle="Doctors-login"

if(pageTitle==="Doctors-login")

console.log("Still in Home page,not navigated")

else if(pageTitle==="Doctors-Dashboard")

console.log("Dashboard page is displayed")

else

console.log("Dashboard page is not displayed")

**4.switch statement:** Here it will search for the particular value in case block ,then that particular will executes if there is no matching case block then default block will executes.

var Dashboardname="Admin-dashboard"

switch(Dashboardname){

    case "Doctors-dashboard":console.log("Doctors-dashboard is displayed")

    break

    case "Admin-dashboard":console.log("admin-dashboard is displayed")

    break

    case "Patient-dashboard":console.log("Doctors-dashboard is displayed")

    break

     default: console.log("invalid page")

}

**Loops :**

1. for. 🡪 (to prints everything even it is undefined)
2. while.
3. do while.
4. for in🡪 (it will skip index of unassigned/undefined value)
5. for of🡪(to prints everything even it is undefined)
6. forEach( ) 🡪(it will skip unassigned/undefined value)

**1.for loop:**  Entry controlled loop ,Before executing statements itself its checking for condition.

In for loop if we wan to make infinite loop we no need to specify condition,only 2 semi-colon(;) is mandatory.

var doc=["nithish",121,"sathish",122,"suresh",123,"mahesh",124]

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

    if(doc[index]==="mahesh"){

    console.log("Doctor's profile has been added successfully and his id is "+doc[++index] )

    var flag =true

    break;

    }

    index++;

}

if(flag ==false)

console.log("doctor profile is not added")

**2.while loop:** Entry controlled loop ,Before executing statements itself its checking for condition. Here providing condition is mandatory.

var doc=["nithish",121,"sathish",122,"suresh",123,"mahesh",124]

var index=0

while(index < doc.length){

    if(doc[index]==="mahesh"){

        console.log("Doctor's profile has been added successfullyand his id is "+doc[++index] )

        var flag=true

        }

       index= index+2

    }

    if(flag ==false)

    console.log("doctor profile is not added")

**3.do while loop:** Exit controlled loop, it will executes once before checking for condition.

var doc=["nithish",121,"sathish",122,"suresh",123,"mahesh",124]

var index=0

do {

    if(doc[index]==="mahesh"){

        console.log("Doctor's profile has been added successfully and his id is"+doc[++index])

        var flag=true

        break;

        }

index=index+2

} while (index < doc.length);

if(flag==false)

console.log("doctor profile is not added")

**4.for in loop:** JS developers developed this loop in such a way that, it should iterates an array from 0th index to array.Length based on index value .

var doc= new Array("nithish","sathish","suresh","mahesh")

for (const index in doc) {

    console.log(index+ "--->"+doc[index])

    if(doc[index]==="mahesh"){

        console.log("Doctor's profile has been added successfully ")

        var flag=true

        break;

        }

    }

    if(flag==false)

    console.log("doctor profile is not added")

**5.for of loop:** JS developers developed this loop in such a way that, it should iterates one by one value from an array from 0th index to array.Length directly .

var doc=["nithish",121,"sathish",122,"suresh",123,"mahesh",124]

for (const names of doc) {

    if(names==="mahesh"){

        console.log("Doctor's profile has been added successfully")

        var flag=true

        break;

        }

    }

    if(flag==false)

    console.log("doctor profile is not added")

**6.forEach() :** It is an predefined method of Array By using array reference we are calling this method by passing callback function which takes three argumments (value,index,array) as a parameter .

**Functions :**

1. Standard function/function declaration.
2. Function expression.
3. Immediate invoke functions.
4. Arrow functions.

**1.Standard function/function declaration:**

**function doc(name){**

**console.log("doctor name is "+name);**

**}**

**2. Function Expression:**

**var doc=function (name){**

**console.log("doctor name is "+name);**

**}**

**3. Immediate invoke function:**

**(function (name){**

**console.log("doctor name is "+name);**

**})(“nithish”)**

This function invoke immediately once after declared.

**Avoid polluting the global namespace.**

Because our application could include many functions and global variables from different source files, it's important to limit the number of global variables. If we have some initiation code that we don't need to use again, we could use the IIFE pattern. As we will not reuse the code again, using IIFE in this case is better than using a function declaration or a function expression.

**4. Arrow function:**

**a) Var doc= (name)=>{console.log("doctor name is "+name)**

**console.log("Hi”)**

**}**

**b) Var doc= (name)=>console.log("doctor name is "+name)// if it is single statement then, no need of braces and no need of using return keyword for single return statement.**

**Difference between function declaration and function expression:**

|  |  |
| --- | --- |
| **Function declaration:**  **doc("nithish")//we can call it before also**  **function doc(name){**  **console.log("doctor name is "+name);**  **}**  **doc("sathish")** | **Function expression:**  **//doc("nithish") if we call it before it will through error i.e. doc is not a function**  **var doc=function (name){**  **console.log("doctor name is "+name);**  **}**  **doc("nithish")** |

**Callback Functions:** Calling one function inside another function is known as callback functions.

function doctor(dname,pname,operation){

    console.log (dname+'is added successfully')

    operation(pname)

}

function patient(pname){

    console.log (pname+' is admitted successfully')

}

doctor("nithish","vinod",patient)

Output

nithishis added successfully

vinod is admitted successfully

**Closure Property:**

Hiding /closing accessibility of an variable outside the functions.

Or

A closure can be defined as a JavaScript feature in which the inner function has access to the outer function variable. In [JavaScript](https://www.javatpoint.com/javascript-tutorial), every time a closure is created with the creation of a function.

The closure has three scope chains listed as follows:

* Access to its own scope.
* Access to the variables of the outer function.
* Access to the global variables.

**Example:** (Access to the variables of the outer function.)

**function fun()**

**{**

**var a = 4; // 'a' is the local variable, created by the fun()**

**function innerfun(b) // the innerfun() is the inner function, or a closure**

**{**

**return a\*b;**

**}**

**return innerfun;**

**}**

**var output = fun();**

**document.write(output(5)); // 20**

**Explination:** Whenever we declare variable inside a functions, it belongs to function scope itself whether it is var,let,const anyting.but we can access variable from global scope inside the functions.

Whenever we declare variable inside a block, it belongs to that scope itself when they declare with let and const keyword but var is gloabally accessible .but we can access variable from global scope inside the blocks.

|  |  |
| --- | --- |
| **Functions** | **Blocks** |
| **var a = 10;**  **let b=20;**  **const c=30;**  **function add( ) {**  **var x = 2;**  **let y=2;**  **const z=3;**  **console.log(a);//10**  **console.log(b)//20**  **console.log(c);//30**  **console.log(x);//2**  **console.log(y)//2**  **console.log(z);//3**  **}**  **console.log(a);//10**  **console.log(b)//20**  **console.log(c);//30**  **console.log(x);//undefined**  **console.log(y)//undefined**  **console.log(x);//undefined** | **var a = 2;**  **let b=2;**  **const c=3;**  **if (Math.random() > 0.5) {**  **var x = 1;**  **let y=2;**  **const z=3;**  **console.log(a);//10**  **console.log(b)//20**  **console.log(c);//30**  **console.log(x);//2**  **console.log(y)//2**  **console.log(z);//3**  **} else {**  **var x = 2;**  **let y=2;**  **const z=3;**  **}**  **console.log(a);//10**  **console.log(b)//20**  **console.log(c);//30**  **console.log(x);//2 ,since it is var we can access**  **console.log(y)//undefined**  **console.log(x);//undefined** |

**Note:**Variables created without a declaration keyword (var, let, or const) are always global, even if they are created inside a function.

A **closure** is the combination of a function bundled together (enclosed) with references to its surrounding state (the **lexical environment**). In other words, a closure gives you access to an outer function's scope from an inner function. In JavaScript, closures are created every time a function is created, at function creation time.

**Object :**

A javaScript object is an entity having state and behavior (properties and method). For example: car, pen, bike, chair, glass, keyboard, monitor etc. JavaScript is an object-based language. Everything is an object in JavaScript.

**In JavaScript, almost "everything" is an object.**

* Booleans can be objects (if defined with the new keyword)
* Numbers can be objects (if defined with the new keyword)
* Strings can be objects (if defined with the new keyword)
* Dates are always objects
* Maths are always objects
* Regular expressions are always objects
* Arrays are always objects
* Functions are always objects
* Objects are always objects

All JavaScript values, except primitives, are objects.

**JavaScript Primitives**

A primitive value is a value that has no properties or methods.

3.14 is a primitive value

A primitive data type is data that has a primitive value.

JavaScript defines 7 types of primitive data types:

**Examples**

* **string**
* **number**
* **boolean**
* **null**
* **undefined**
* **symbol**
* **bigint**

**Immutable**

Primitive values are immutable (they are hardcoded and cannot be changed).

if x = 3.14, you can change the value of x, but you cannot change the value of 3.14.

**Objects are Variables**

JavaScript variables can contain single values:

Example

let person = "John Doe";

JavaScript variables can also contain many values.

Objects are variables too. But objects can contain many values.

Object values are written as **name : value** pairs (name and value separated by a colon).

**Example**

**const person = {firstName:"John", lastName:"Doe", age:50, eyeColor:"blue"};**

A JavaScript object is a collection of **named values**

It is a common practice to declare objects with the const keyword.

1. **Using object literals:**

Literals are smaller and simpler ways to define objects.We simple define the property and values inside curly braces as shown below:

|  |
| --- |
| **//creating js objects with object literal**  **let car = {**  **name : 'GT',**  **maker : 'BMW',**  **engine : '1998cc'**  **start : function(){**  **console.log('Starting the engine...');**  **};**  **//property accessor**  **console.log(car.name); //dot notation**  **console.log(car['maker']); //bracket notation**  **car.start();//invoking fnction**  **To add new Property**  **Car.stop= function(){**  **console.log('Stopping the engine...');**  **};**  **Car.stop()// Stopping the engine...**  **To modify Property**  **Car.stop= function(){**  **console.log('switchoff the engine...');**  **};**  **Car.stop()//switchoff the engine...**  **To delete Property**  **delete Car.stop**  **Car.stop()//undefined** |

1. **Creating object with a constructor:**

One of the easiest ways to instantiate an object in JavaScript. Constructor is nothing but a function and with help of new keyword, constructor function allows to create multiple objects of same flavor as shown below:

|  |  |
| --- | --- |
| **//simple function**  **function vehicle(name,maker,engine){**  **this.name = name;**  **this.maker = maker;**  **this.engine = engine;**  **}**  **//new keyword to create an object**  **let car  = new vehicle('GT','BMW','1998cc');**  **//property accessors**  **console.log(car.name);**  **console.log(car.maker);**  **console.log(car['engine']);**   1. **Using new Keyword:**   By using new Keyword we can create object.  **var doctor= new Object()**  **doctor.name="nithish"**  **doctor.age=25**  **doctor.patient=()=>"vinod is a patient"**  **console.log(doctor.patient());**  **console.log(doctor.age);**   1. **Using es6 classes:**   ES6 supports class concept like any other Statically typed or object oriented language. So, object can be created out of a class in javascript as well as shown below:   |  | | --- | | **class Vehicle {**  **constructor(name, maker, engine) {**  **this.name = name;**  **this.maker =  maker;**  **this.engine = engine;**  **}**  **}**    **let car1 = new Vehicle('GT', 'BMW', '1998cc');**    **console.log(car1.name);  //GT** |   **Array:** |