**Current Version : ES14 (Ecma Script 14)**

1. **What is DOM ?**

DOM Stands for Document Object Module.

We can GET, SET, create and remove html elements by using DOM.

var email = document.getElementById("email").value ; // Get value

document.getElementById("email").value = "My value"; // Set Value

Create html elements by DOM:

<button onclick="myFunction()">Create HTML Element</button>

<script>

function myFunction() {

var btn = document.createElement("BUTTON");

document.body.appendChild(btn);

}

</script>

Remove html elements by DOM:

<p id="demo">Click the button, and this paragraph will be removed from the DOM.</p>

<button onclick="myFunction()">Remove paragraph</button>

<script>

function myFunction() {

var myobj = document.getElementById("demo");

myobj.remove();

}

</script>

1. **What is closure ?**

A closure is a feature of JavaScript where inner function can access to the variables of outer (enclosing) function’s.

Features:

* it can access it’s own scope.
* it can access outer function’s variables.
* it can access global variables.

Example:

function outer()

{

var b = 10;  
 function inner()

{  
   
 var a = 20;   
 console.log(a+b);  
 }  
 return inner;  
}

1. **What is the chaining process in jquery ?**

$("#p1").css("color", "red").slideUp(2000).slideDown(2000);

1. **What is the difference between ES5 and ES6 ?**

|  |  |  |
| --- | --- | --- |
|  | **ES5** | **ES6** |
| 1. | There are only one way to define the variables by using the var keyword. | There are three ways to define variables that are var, let and const. |
| 2. | We can not use oops concept. | We can use oops concept. |

1. **What are the differences among var, let and const ?**

**var :** Generally, It's scope remains on the entire page and we can change it's value.

**let :** Generally, It's scope remains on the entire page and we can change it's value.

**const** **:** Generally, It's scope remains on the entire page and we can't change it's value.

**var is function scope and let, const are block scope.**

**Note**:

* + 1. Only variable declaration, is valid for “var” and “let” keyword.

Exp: var a; let b;

* + 1. For “const” variable, we need to must declaration and initialization together.

Exp: const a=10;

* + 1. var is function scope and let, const are block scope.
    2. If we will declare any variable by using var,let or const keyword outside function, then it’s scope remains on the

entire function/page.

* + 1. If we will declare any variable by using var keyword any where with in function, then it’s scope remains on the

entire function. We can not access this variable out side of function.

* + 1. If we will declare any variable by using let or const keyword, out side of all block(like: if block, else block) with in

function, then it’s scope remains on the entire function. We can not access this variable out side of function.

* + 1. If we will declare any variable by using let or const keyword, inside any block with in function, then it’s scope remains

on the particular block(where it is defined). We can not access this variable out side of this block.

* + 1. If we we will declare same variable out side of block and inside of block with in function, then the javascript compiler

assumes both are different variables. (it is applicable for var,let and const)

(g) Similaraly, If we will declare same variable out side function and inside function, then the javascript compiler assumes

both are different variables. (it is applicable for var,let and const)

(g) We can declare same variable multiple times on same location by using var keyword, it will work according to last

declaration. But we can not declare same variable multiple time on same location by using let and const keyword.

1. **What is temporal dead zone ?**

Temporal dead zone is a behaviour that occurs with let and const variables, when we try to access variables before it’s declaration. If we will try to access var before it’s declaration than it will show **undefined,** Similarly if we will try to access let and const before it’s declaration it will throw **Reference Error**.

1. **What is the constructor ?**

Constructor is a function when we create an object then the constructor will call automatically.

<p id="demo"></p>

<script>

// Constructor function for Person objects

function Person(first, last, age, eye) {

this.firstName = first;

this.lastName = last;

this.age = age;

this.eyeColor = eye;

}

// Create a Person object

var myFather = new Person("John", "Doe", 50, "blue");

// Display age

document.getElementById("demo").innerHTML =

"My father age is " + myFather.age + ".";

</script>

O/P: My father age is 50.

1. **What is the anonymous function ?**

Anonymous function is a [function](https://www.javascripttutorial.net/javascript-function/) without a name. Anonymous function is often not accessible after its initial creation.

Example:

setTimeout(function () { console.log('Execute later after 1 second'); }, 1000 );

1. **What is the call back function ?**

In callback function, we can pass a function as a parameter. We can call multiple function by using a single function.

Ex:

<script>

function sayHello()

{

console.log(“Hello”);

}

function sayHi()

{

console.log(“Hi”);

}

function add(num1,num2,callback)

{

console.log(num1+num2);

callback();

}

add(“10”,”20”,sayHello); //it will call sayHello function.

add(“30”,”40”,sayHi); //it will call sayHi function.

//output:

30Hello

70Hi

</script>

1. **What is the NAN property ?**

NaN is a value that represents not a number. NaN is never equal to any number, including NaN itself.

**Syntax**

NaN

1. **What is the Recursion ?**

Recursion is best applied when you need to call the same function repeately with different parameters within a loop.

Example to find factorial of a number:

<script>

function fact(n)

{

if (n == 0 || n == 1)

{

return 1;

}

else

{

return n \* fact(n - 1);

}

}

var factorial = fact(5);

alert( “Factorial is ”+factorial );

</script>

1. **What are the differences between web storage(Cookies), local storage and session storage ?**

|  |  |  |  |
| --- | --- | --- | --- |
|  | **web Storage (Cookies)** | **Local Storage** | **Session Storage** |
| Size | Maximum size of cookies is 4kb. | Maximum size of local storage is 10mb. | Maximum size of session storage is 5mb. |
| HTML version | It is supported by HTML 4 and HTML 5 version. | It is supported by HTML 5 version. | It is supported by HTML 5 version. |
| Accessible | It is accessible from any window. | It is accessible from any window. | It is accessible on same tab. |
| Expire | To expire cookies we set manually. | Local storage never expire. | Session storage will expire on tab closing. |
| Storage Location | Cookies store in browser and server. | Local storage store in browser only. | Session storage store in browser only. |
| Send with request | We can send cookies with request. | We can not send request. | We can not send request. |

**To check in console of browser - Press ctrl + shift + i => Application**

**LocalStorage:**

localStorage.setItem(‘FirstName’,’Vishal’) ; // to set

alert(localStorage.getItem(‘FirstName’)); // to get

localStorage.removeItem(‘FirstName’); // to remove

**Same as sessionStorage**:

**SessionStorage:**

sessionStorage.setItem(‘FirstName’,’Vishal’) ; // to set

alert(sessionStorage.getItem(‘FirstName’)); // to get

sessionStorage.removeItem(‘FirstName’); // to remove

1. **What is clear ?**

A clear is utilized when you don’t need an element to wrap around another element, such as a float.

1. **What is the differences between host object and native object ?**

Host Objects  – It is provided by a particular environment.

Native Objects – It is provided by Javascript.

1. **What is the differences between HTML and XHTML ?**

HTML and XHTML both are markup languages which are used to create web page. The main difference is that, HTML syntax is based on SGML while XHTML syntax is based on XML.

1. **Explain how variables differ in java script than coffee script ?**

In JavaScript, we have to add a semi-colon for termination of statement. In CoffeeScript, there is no need to add semi-colon for termination of statement.

1. **Explain the importance of DOCTYPE in HTML?**

DOCTYPE is an introduction with browser that which version of HTML we are going to use.

1. **What is the feature of javascript ?**

(1) It is light weight, object based scripting language.

(2) we can validates user inputs before sending to server.

(3) Cross platform.

(4) It can generate dynamic html content.

(5) It is case sensitive.

1. **Describe below points ?**

**>> this keyword** : It refers to the object where it was called.

>> **NULL** : It represents no value or no Object

>> **Nan** – It represents not a number

>> **isNan() –** To identify that, the value is number or not.

>> **typeof()** : To check data type.

Exp: var a=10;

typeof(a); // it will return data type of variable a

>> **Number()** : To convert in integer value.

Exp: var a=”10”;

Number(a);

>> **Objects:**

var obj = { firstname:’Anil’ , lastname:’Singh’ };

console.log(obj.firstname);

>> **Array:**

var colors = ["Red", "Yellow", "Green", "Orange"];

var cities = ["Noida", "Delhi", "Ghaziabad"];

alert(colors[2]);   // Output: Green

alert(cities[1]);   // Output: Delhi

1. **What is BOM ?**

**BOM** stands for Browser Object Model. It provides interaction with the browser. The default object of browser is window. So, we can access the functions of window directly. The window object provides various properties like document, history, screen, navigator, location, innerHeight, innerWidth etc.

1. **What are the different data types in javascript ?**

There are two types of data types in JavaScript:

o Primitive data types (String, Number, Boolean, BigInt, Undefined, Null, Symbol, typeof)

o Non- Primitive data types (Object, Array)

Primitive data types

(1)String:

Example:

var str1 = "Hello JavaTpoint"; //using double quotes

var str2 = 'Hello Javatpoint'; //using single quotes

(2)Number:

Example:

var x = 5; //without decimal

var y = 5.0; //with decimal

(3)Boolean:

Example:

var x = true;

var y = false;

(4)BigInt:

Example:

var bigInteger = 123456789012345678901234567890;

(5)Undefined:

Example:

var x; // value of x is undefined

var y = undefined; // You can also set the value of a variable as undefined.

(6)Null:

Example:

var x = null;

(7)Symbol:

Example:

var symbol1 = Symbol('symbol');

(8)typeof:

Example:

var str1 = "Hello JavaTpoint";

typeof (strl);

Non-Primitive data types

(1)Object:

Example:

var obj1 = { x:123, y:"Welcome to JavaTpoint", z: function(){ return this.x; } }

(2)Array:

Example:

var colors = ["Red", "Yellow", "Green", "Orange"];

var cities = ["Noida", "Delhi", "Ghaziabad"];

alert(colors[2]); // Output: Green

alert(cities[1]); // Output: Delhi

1. **What are the differences between == and === operator ?**

The == operator is used to compare value and === operator is used to compare value with data type.

1. **How to write HTML code dynamically using javascript ?**

example:

document.getElementById('mylocation').innerHTML="<h2>This is heading using JavaScript</h2>";

1. **What is the real name of javascript ?**

The original name was **Mocha**, a name chosen by Marc Andreessen, founder of Netscape. In September of 1995, the name was changed to LiveScript. In December 1995, after receiving a trademark license from Sun, the name JavaScript was adopted.

1. **What is the differences between event.preventDefault() and event.stopPropagation() method in javascript ?**

event.preventDefault() method is used to prevent the default behavior of an element.

For example: If it is used in form element, then it prevents form submitting. If it is used in anchor element, it prevents navigating.

The event.stopPropagation() method is used to stop the propagation of an event.

1. **How to set the cursor to wait in javascript ?**

Example:

1. **<script>**
2. window.document.body.style.cursor = "wait";
3. **</script>**
4. **What is this [[[]]] ?**

This is a three-dimensional array.

1. var myArray = [[[]]];
2. **What is the role of strict mode in javascript ?**

strict mode is used to generates silent errors. It provides "use strict"; expression to enable the strict mode. This expression can only be placed as the first statement in a script or a function.

For example:

"use strict";

x=10;

console.log(x);

1. **What do you understand by hoisting in javascript ?**

### Hoisting is the default behavior of JavaScript where all variable declarations and function declarations are moved on top bydefault ( Only declaration part will move on top ).

Example:

hoistedFunction(); // Outputs " Welcome to JavaTpoint " even when the function is declared after calling

function hoistedFunction() //this is function declaration

{

console.log(" Welcome to JavaTpoint ");

}

**Note:**

* + 1. Declaration and initialization both are different.

Ex : var a; // this is declaration ( Only declaration part moves on top bydefault )

a=10; // this is initialization

var a=10; // this is declaration and initialization

* + 1. function show() { console.log(“welcome”); } // this is function declaration
    2. Only declaration part will move on top in hidden.
    3. Hoisting is valid only for “var” keyword.
    4. Hoisting is valid only for normal function ( it is not valid for function expression and arrow function ).
    5. Javascript compiler reads code from top to bottom.
    6. Only variable declaration, is valid for “var” and “let” keyword.

Exp: var a; let b;

* + 1. For “const” variable, we need to must declaration and initialization together.

Exp: const a=10;

**Example of “var” keyword:**

<script>

            var a = 10;

            document.write(a);         // output: 10

</script>

 <script>

            document.write(a);         // output: undefined

            var a = 10;

 </script>

**Note:** Only declaration part will move on top in hidden. Javascript compiler reads top to bottom code.

 <script>

            document.write(a);         // output: Uncaught ReferenceError: a is not defined

 </script>

    <script>

           function show()

           {

            document.write("welcome");            // output: welcome

           }

           show();

    </script>

   <script>

           show();

           function show()

           {

            document.write("welcome");            // output: welcome

           }

   </script>

**Note:** Only declaration part will move on top in hidden. Javascript compiler reads top to bottom code.

 <script>

           var a = 10;

           show();

           function show()

           {

            document.write("welcome"+" "+ a);            // output: welcome 10

           }

 </script>

 <script>

           show();

           function show()

           {

            document.write("welcome"+" "+ a);            // output: welcome undefined

           }

           var a = 10;

 </script>

**Note:** Only declaration part will move on top in hidden. Javascript compiler reads top to bottom code.

**Example of “let” keyword:**

<script>

            document.write("welcome"+" "+ a);   //output: Uncaught ReferenceError: Cannot access 'a' before initialization

            let a = 10;

 </script>

**Note:** Declaration part doesn’t move bydefault on top in hidden. Javascript compiler reads top to bottom code.

**Example of “const”keyword:**

 <script>

            document.write("welcome"+" "+ b );   //output: Uncaught ReferenceError: Cannot access 'b' before initialization

            const b = 20;

 </script>

**Note:** We need to must declare and initialization first than we can use .

1. **What are the falsy values in javascript and how can we check the value is falsy or not?**

Those values which become false while converting to Boolean are called falsy values.

We can check the value is falsy or not by using the Boolean function or Double NOT operator (!!).

1. **How to select even/odd number div by jquery ?**

<script>

$(document).ready(function(){

$("div:even").css("background-color", "yellow");

});

</script>

1. **What is javascript object ?**

JavaScript object is a standalone entity, with properties and type. which define characteristics.

Exp:

var obj = { name:”Ram” , age:25 , z: function() {return this.name} };

1. **What is the use of this keyword ?**

<script>

function Human(firstName, lastName)

{

this.firstName = firstName,

this.lastName = lastName,

this.fullName = function() {

return this.firstName + " " + this.lastName;

}

}

var person1 = new Human("Virat", "Kohli");

console.log(person1)

</script>

Note: variable name and parameter name both are same.

1. **What is the different way of declaring function ?**

Hello();

1st Method: function Hello(){ alert(“Hello”); }

2nd Method: var Hello = function(){ alert(“Hello”); }

3rd Method: var Hello = ()=>{ alert(“Hello”); }

1. **What is the rest operator(…) ?**

Old method:

function Human(a,b,c,d,e)

{

alert(a+b+c+d+e);

}

Human(10,20,30,40,50);

New method ( Ignore this ) :

function Sum()

{

var sum=0;

for(let i in arguments)

{

sum = sum + arguments[i]

}

alert(sum);

}

Sum(100,200,300,400,500);

New Method2:

function Sum(name,...args) //args[0]=100, args[1]=200, args[2]=300, args[3]=400, args[4]=500

{

var sum=0;

for(let i in args)

{

sum = sum + args[i]

}

alert(name+"="+sum);

}

Sum("Total",100,200,300,400,500);

**Note:**

(1) ...(three dots) is rest operator i.e. first para will store in name variable and rest para will store in arguments variable.

(2) arguments and args is a predefined keyword.

(3) ...args can only be placed as a last parameter.

(4) we use rest operator for receiving value.

1. **What is the spread operator(…) ?**

<script>

function Sum(name,...args) // this is rest operator

{

var sum=0;

for(let i in args)

{

sum = sum + args[i]

}

alert(name+"="+sum);

}

var arr = [10,20,30];

arr.push(40); // it will add one value "40" in arr, i.e. it will consider var arr = [10,20,30,40]

arr.pop(10); //it will remove the value 10,i.e. it will consider var arr = [20,30,40]

Sum("Total",...arr); // this is spread operator.

//it will pass 4 values while without spread operator it will consider one value

/\*

Note:

(1) we use spread operator for sending value and rest operator for receiving value.

(2) we use spread operator only calling time.

(3)...args can only be placed as a last parameter.

(4) we use spread operator for sending value.

\*/

</script>

1. **What is the prototype or prototype object ?**

Prototype is an object where we can attach methods and properties in a prototype object, which enables all other objects to inherit these methods and properties.

<script>

var a = {name:"Hurry",language:"Javascript",run:()=>{alert("self run")}}

var p = {name:"jaiky",email:"test@email.com",run:()=>{alert("prototype run")}}

a.\_\_proto\_\_= p; //object p has stored in object a, so we can use all property of object p by using object a.

a.run(); // it will execute own key , if not found then seach in object p

alert(a.email);

p.\_\_proto\_\_ = {age:30} // we have added a key in object p by using prototype

alert(a.age);

</script>

<script>

// function constructor

function Person(name, job, yearOfBirth)

{

     this.name= name;

     this.job= job;

     this.yearOfBirth= yearOfBirth;

}

// this will show Person's prototype property.

console.log(Person.prototype);

</script>

1. **What is the concatenation of array and object ?**

<script>

var arr1 = [10,20,30];

var arr2 = [40,50,60];

var arr3 = arr1.concat(arr2); //both array arr1 and arr2 has concatinated in arr3 by simple method

var arr4 = [...arr1,...arr2]; //both array arr1 and arr2 has concatinated in arr4 by spread operator

var arr5 = [...arr2,...arr1]; //it will change the sequence of values

var arr6 = [70,...arr2,...arr1,90]; //it will add two values

var obj1 = {name:"Rajesh",surname:"verma"}

var obj2 = {age:25}

var obj3 = {...obj1,...obj2} //both objects obj1 and obj2 has concatinated in obj3 by spread operator

var obj4 = {...obj2,...obj1} ////it will change the sequence of objects

var obj5 = {course:"Powerapps",...obj2,...obj1,duration:"3 Month"} //it will add two variable

alert(arr3[0]);

alert(arr3[5]);

alert(arr4[0]);

alert(arr4[5]);

alert(obj3.name);

alert(obj5.course);

</script>

1. **What is Promise() method in javascript ?**

It takes two parameters – resolve and reject(both are user defined), if the condition will fulfill then resolve method will call other wise reject method will call. We can handle both cases by using then and catch method.

Exp:

      <script>

            let complete = false;

            let prom = new Promise(

                function(resolve,reject)   // Parameter is not mandatory, we can take according to need

                                            // first para for success case and second para for failure case

                {

                    if(complete)

                    {

                        resolve("I AM success");

                    }

                    else

                    {

                        reject("I AM Failure");

                    }

                }

            );

            //console.log(prom);

            //if we want to handle error then we will use below 3 statements

            prom.then((result)=>{ alert(result); }).catch((error)=>{ alert(error) });   //calling

            //if promise will resolve(condition will true) then then() function will execute other wise catch() function will execute.

        </script>

**Note:**

* 1. If condition is fullfill than resolve() function will call other wise reject() function will call. resolve and reject parameter both are user defined.
  2. result variable will stote the data which will returned by resolve() method().
  3. error variable will stote the data which will returned by reject() method().
  4. result and error both are user defined variables.
  5. then() and catch() both are call back functions , which don’t require to call.

1. **What is Promise.all() method in javascript ?**

We can handle multiple Promise by using Promise.all() method.

<script>

        let p1 = new Promise((resolve, reject) => { console.log("first promise"); resolve(20); });

        let p2 = new Promise((resolve, reject) => { console.log("second promise"); resolve(30); });

        let p3 = new Promise((resolve, reject) => { console.log("third promise"); resolve(40); });

        Promise.all([p1, p2, p3]).then((result) => { console.log(`Result:${result}`) }).catch((error) => { console.log(`Error:${error}`) });

        var total = 0;      // to find total of all values

        Promise.all([p1, p2, p3]).then((result) =>

        {

                    console.log(`Result:${result}`);

                    for (var i in result)

                    {

                        total = total + result[i]

                    }

                    console.log(`total:${total}`)

        }).catch((error) =>

        {

            console.log(`Error:${error}`)

        });

    </script>

Note:

* 1. if all promises will true then then() function will execute other wise catch() function will execute.
  2. we can also call / execute single promise.

1. **What is template string or template literals ( Concatination of variables in ES6) in javascript ?**

**//Old method**

var a = “welcome”;

var b = a + “Ram”;

alert(b);

**//New method by template string (using back tick)**

var a = “welcome”;

var b = `${a} Ram`; or ` “${a}” Ram`; or ` ${a} ‘Ram’`; //we can use multiple variable, single quote

document.write(b);

**// we can use in function**

var a = “welcome”;

var b = “Ram”;

function fullname(a,b)

{

return `${a} ${b}`;

}

var hello = `Good Morning ${fullname(a,b)}`;

alert(hello);

**// we can use in Arrow function**

var a = “welcome”;

var b = “Ram”;

let fullname = (a,b) =>

{

return `${a} ${b}`;

}

var hello = `Good Morning ${fullname(a,b)}`;

alert(hello);

1. **What is Object literals (Define object by ES6 method) in javascript ?**

**// Old method**

var name = “welcome”;

var course = “english”;

var obj = {name:name,course:course};//**Old Method** (property name and value name both are same )

var obj = {name, course}; // **New Method**

**2nd exp of New method(ignore):**

var name = “welcome”;

var obj1 = { [name]:”Raj”, age:25 }; i.e. var obj2 = { welcome:”Raj”, age:25 };

var obj2 = { [name + “back”]:”Raj”, age:25 }; i.e. var obj2 = { welcomeback:”Raj”, age:25 };

var obj3 = {

[name + “back”]:”Raj”,

age:25,

details:function(){return `${this.welcomeback},${this.age}`}

};

console.log(obj3.details()); or console.log(obj3[‘details’]());

var obj4 = { name:”Rahul”, age(){ return this.name } // we can use function directly

1. **What is Array Destructuring(Storing array values in variables by ES6 method) in javascript ?**

**// Old method**

var arr = [“Welcome”, 25];

var a = arr[0];

var b =arr[1];

**// New method (Array destructuring)**

var arr = [“Welcome”, 25];

var[a,b] = arr; i.e. var a = “Welcome”; var b = 25;

**1st exp:**

var arr = [“Welcome”, 25,];

var[a,b,c=”delhi”] = arr; i.e. var a = “Welcome”; var b = 25; var c = “delhi”;

**2nd exp(ignore):**

var arr = [“Welcome”, 25, [“male”,2500] ];

var[a,b,[gender,salary]] = arr; i.e. var a = “Welcome”; var b = 25; var gender = “male”;

var salary = 2500;

**3rd exp(ignore):**

var arr = [“Welcome”, 25, “Delhi”];

var[a,…args] = arr; i.e. var a = “Welcome”; args[0] = 25; args[1]=”Delhi”;

**4th exp(ignore):**

function user([name,age,city])

{

console.log(name);

console.log(age);

console.log(city);

}

user([“Raj”, 25 ,”Delhi”]);

**5th exp(ignore):**

function user()

{

return [“Raj”,25,”Delhi”];

}

var [name,age,city] = user(); //i.e. var name=”Raj”; var age=25; var city=”Delhi”;

console.log(name);

console.log(age);

console.log(city);

1. **What is Object Destructuring(Storing object values in variables by ES6 Method) in javascript ?**

**// Old method**

var obj = {name:”Raj” , age:25};

console.log(obj.name);

console.log(obj.age);

**// New method**

var obj = {name:”Raj” , age:25};

var{name,age} = obj; //property name and variable name must be same like: name and age.

//i.e. var name = “Raj”; var age = 25;

**Another example(ignore):**

var obj = {name:”Raj” , age:25};

var{name:n , age:a } = obj; //property name and variable name must be same like: name and age.

//i.e. var n = “Raj”; var a = 25;

We can use any character or letter instead of n and a.

1. **OOP(Object Oriented programming) concept in javascript ?**

**1st Exp:**

class Hello

{

message()

{

alert(“message function called”);

}

}

var obj = new Hello();

obj.message();

**Types of method:**

1. constructor
2. prototype
3. static

**constructor**:

it will automatically call when we create object, we need to use predefined keyword “constructor”. We can also pass argument(parameter) to constructor.

**prototype:**

self defined methods are called prototype methods.

**static:**

we need to use predefined keyword “static”. We don’t require to create object. We can call static methods by class name.

**2nd Exp:**

class Hello

{

constructor()

{

var age ; // we can define variable inside function only.

alert(“constructor called”);

}

message()

{

alert(“message function called” + this.age);

}

}

var obj = new Hello();

obj.age = 25;

obj.message();

**3rd Exp:**

class Hello

{

constructor(name,age)

{

this.studentname = name;

this.studentage = age;

alert(“constructor called”);

}

message()

{

alert(“message function called” + this. Studentname + this.studentage);

OR

alert(`message function called ${this. Studentname} ${ this.studentage}`);

}

static demo()

{

alert(“demo function called”);

}

}

var obj = new Hello(“Rajesh”,25);

obj.message();

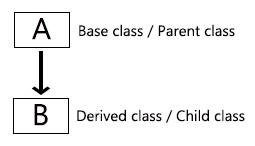
Hello.demo();

1. **Inheritance in javascript ?**

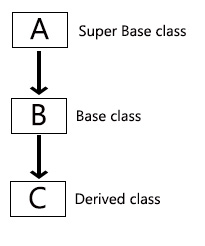
Inheritance is one of the feature of oops. It allows child class to access the properties and function of parent class.

Javascript supports 4 types of Inheritance:

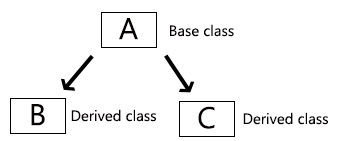
**>> Single Inheritance ( Pseudoclassical Inheritance )**



**>> Multilevel Inheritance**



**>> Hirarchical Inheritance**



**>> Prototype Inheritance**

1. Single Inheritance ( Pseudoclassical Inheritance )

<script>

            class A

            {

                Hi()

                {

                    alert("Hi");

                }

            }

            class B extends A

            {

                Hello()

                {

                    alert("Hello");

                }

            }

            var obj = new B();

            obj.Hi();

            obj.Hello();

        </script>

<script>

class employee //base class

{

constructor(name)

{

console.log(“base class constructor”+name);

}

}

class manager extends employee //derived class

{

constructor(name)

{

super(name); //to call base class constructor on first priority this is mandatory

console.log(“derived class constructor”+name);

}

}

var obj = new manager(“Rajesh”);

</script>

1. Multilevel Inheritance

 <script>

            class A

            {

                Hi()

                {

                    alert("Hi");

                }

            }

            class B extends A

            {

                Hello()

                {

                    alert("Hello");

                }

            }

            class C extends B

            {

                Welcome()

                {

                    alert("Welcome");

                }

            }

            var obj = new C();

            obj.Hi();

            obj.Hello();

            obj.Welcome();

        </script>

1. Hirarchical Inheritance

<script>

            class A

            {

                Hi()

                {

                    alert("Hi");

                }

            }

            class B extends A

            {

                Hello()

                {

                    alert("Hello");

                }

            }

            class C extends A

            {

                Welcome()

                {

                    alert("Welcome");

                }

            }

            var obj = new B();

            obj.Hi();

            obj.Hello();

            var obj2 = new C();

            obj2.Hi();

            obj2.Welcome();

        </script>

1. Prototype Inheritance

<script>

class employee //base class

{

constructor(name)

{

this.empname = name;

console.log(“base class constructor”);

}

info()

{

conslole.log(“emp name”+this.empname);

}

}

class manager extends employee //derived class

{

info()

{

super.info(); //to call base class info method by super keyword this is not mandatory

console.log(“managaer name”+this.empname);

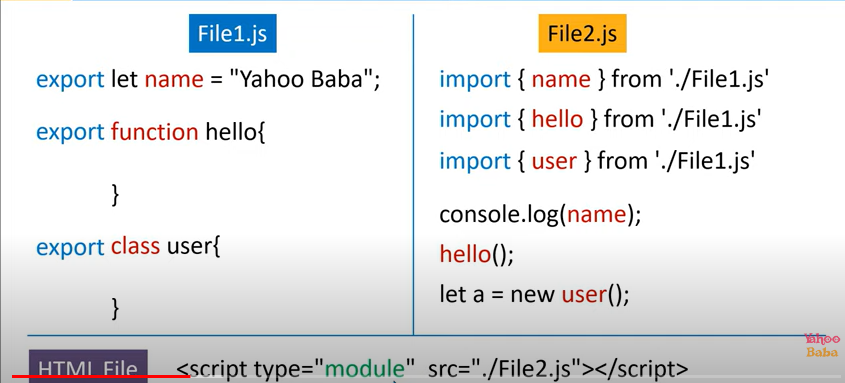
}

}

var obj = new manager(“Rajesh”);

</script>

1. **What is modules in javascript ?**

In Modules we can use another file’s variable, function and class in the current file.

Note:

1. it works with server.
2. We can also write **export { name,hello,user }** and remove **export** keyword from every line in File1.js.
3. We can also write **import { name,hello,user } from “./File1.js”** and remove **import** keyword from every line in File2.js.
4. We can also rename as **import { name,hello,user as us } from “./File2.js”** in File2.js. we can not use **user** because it has been renamed as **us**
5. We can also write **import \* as yahoo from “./File1.js”** to import all things and it will use by **yahoo** keyword

Exp:

console.log(yahoo.name); console.log(yahoo.hello());

1. Module has one default function, we can write below code in file1.js

export default function()

{

console.log(“default function called”);

}

To import write below code in file2.js

import {default as yahoo} from “./file1.js” // rename is mandatory , we can use any name

OR

import yahoo from “./file1.js” // it will automatically consider default function

yahoo(); //called

1. We can also bridge property in modules

We will create one more file bridge.js //we can put any name

i.e. it will total 3 files: file1.js , bridge.js and file2.js

**file1.js**

export var name = “welcome”;

**bridge.js**

export { name } from “./file1.js”;

**file2.js**

import { name } from “./file1.js”;

alert(name);

**note:** similaraly, we can use function and class.

1. **Ajax in javascript ?**

We can send and receive data from server without refreshing current page by using ajax.

<div id="demo"></div>

<button type="button" onclick="loadDoc()">Change Content</button>

<script>

function loadDoc() {

const xhttp = new XMLHttpRequest();

xhttp.onload = function() {

document.getElementById("demo").innerHTML =

this.responseText;

}

xhttp.open("GET", "test.txt"); // set file path

xhttp.send();

}

</script>

1. **fetch() method in javascript ?**

We can feth and display api data by using fetch method.

<script>

fetch('https://jsonplaceholder.typicode.com/users')

.then((response) => response.json())

.then((result) => {

console.log(result);

for(var x in result)

{

document.write(`${result[x].name},${result[x].email}<br/>`)

}

})

.catch((error)=>document.write("File not found"));

</script>

Note:

1. for showing text data we use **text()** method instead of **json()** in above code.
2. fec api link:

https://jsonplaceholder.typicode.com/

1. **what is currying function in javascript ?**

Currying transforms a function with multiple arguments into a nested series of functions, where each function takes a single argument. Currying helps to avoid passing the same variable multiple times and it helps to create a higher order function.

Exp:

   <script>

        function getSum(a)

        {

            return function(b)

            {

                return function(c)

                {

                    return function(d)

                    {

                        return function(e)

                        {

                            return(a+b+c+d+e)

                        };

                    };

                };

            };

        }

        const tot=getSum(5)(4)(3)(2)(1);

        console.log(tot)

        //                    OR

        // const getSum = (a)=>(b)=>(c)=>(d)=>(e)=> a+b+c+d+e;

        // const tot = getSum(5)(4)(3)(2)(1);

        // console.log(tot);

//Note:

// if we have single line return statement then we can use this trick and return keyword is not mandatory.

    </script>

1. **what is function expression in javascript ?**

Function Expression is used to define a function inside any expression. The Function Expression allows us to create an anonymous function.

Exp:

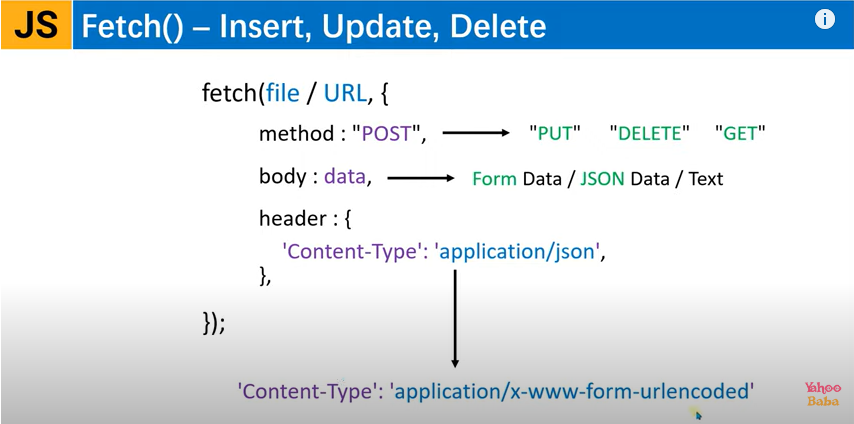
<script>

        let sum = function(a,b){ return (a+b) }; //This type declaration is called function expression

        console.log(sum(2,3));

    </script>

Note:

1. in function expression firstly we need to declare a function then we will call other wise we can not call the function.
2. Hoisting doesn’t work in function expression.
3. **Insert,Update and Delete records in live/local API By Using fetch() method in javascript ?**

**To Insert Record:**

<script>

fetch('https://jsonplaceholder.typicode.com/posts', {

method: 'POST',

body: JSON.stringify({

title: 'vidu',

body: 'vidubody',

userId: 1010,

}),

headers: {

'Content-Type': 'application/json; charset=UTF-8',

},

})

.then((response) => response.json())

.then((json) => console.log(json));

</script>

**Note:**

* + 1. To Update Record we will use **PUT** method instead of POST
    2. To Delete Record we will use below code:

fetch('https://jsonplaceholder.typicode.com/posts/1', {

method: 'DELETE',

}); //we will put id no. which record we want to delete

* + 1. We can put dynamic data instead of **vidu** , **vidubody** and **1010** by creating form, onclick event and function.

Like :

title: document.getElementById(‘titletxt’).value,

body: document.getElementById(‘bodytxt’).value,

userid: document.getElementById(‘useridtxt’).value,

* + 1. To shortcut We can use below code:

**body: new FormData(document.getElementById(‘MyForm’)),** // MyForm is the id of form

instead of

body: JSON.stringify({

title: 'vidu',

body: 'vidubody',

userId: 1010,

}),

and need to change header –

use below code

**headers: {**

**'Content-Type': 'application/x-www-form-urlencoded ',**

**},**

Instead of

headers: {

'Content-Type': 'application/json; charset=UTF-8',

},

* + 1. To check that browser is support or not fetch() method we use below code:

If(window.fetch)

{

}

else

{

}

1. **What is async and await in javascript ?**

**async**  allows to write **promise-based** function. async functions will always return a value. If promise is resolve then then() method will call other wise catch() method will call.

**await** is used to wait for the promise. It is used within the async block only.It makes the code to wait until the promise returns a result.

      <script>

            async function Test()

            {

                const response = await fetch(".....file path or url....");

                const resultdata = await response.json();    //const result = response.text();

                return resultdata;

                                        /\* OR

                return(await fetch(".....file path or url....")).json();

                                        \*/

            }

            Test().then((result)=>{console.log(Test())}).catch((error)=>{console.log(error)});

        </script>

Note:

We can also use try and catch block with in this function to handle error.

1. **What is error handling in javascript ?**

We can handle error by using try and catch block.

<script>

try

{

……..; //type the code which is hope for no error

……..;

}

catch(error)

{

…….; //type the code which is hope for getting error.

} …….;

</script>

1. **Explain Nested Array or Multidimantional Array ?**

<script>

var arr = [

["Harry",18,"Male","B.com"],

["Raj",19,"Male","B.C.A."],

["Rani",20,"Female","M.C.A."],

["Raja",21,"Male","M.B.A."],

["Gita",22,"Female","B.A."]

];

document.write(arr); // To show all array values

for(var i=0; i< arr.length; i++) //arr.length will show total no. of row

{

for(var j=0; j< arr[i].length; j++) //arr[i].length will show no. of colum in current row

{

document.write(arr[i][j]+" ");

}

document.write("<br/>");

}

</script>

**Note:**

Here(first row) arr[0][0]= "Harry", arr[0][1]= 18, arr[0][2]="Male" and arr[0][3]="B.com" ;

same as apply for second,third and fourth row.

1. **Explain Nested Object and array,nested array with in object ?**

<script>

var obj = {

name:"john",

exam:{ midterm:25, final:35 },

arr:[10,20,30],

nestarr: [

["Harry",18,"Male","B.com"],

["Raj",19,"Male","B.C.A."]

]

};

document.write(obj.name);

document.write("<br/>");

document.write(obj.exam.midterm);

document.write("<br/>");

document.write(obj.exam.final);

document.write("<br/>");

document.write(obj.arr[0]);

document.write("<br/>");

document.write(obj.arr[1]);

document.write("<br/>");

document.write(obj.nestarr[0][0]);

document.write("<br/>");

document.write(obj.nestarr[0][1]);

</script>

1. **What is the use of “for in” and “for of” loop in javascript ? [Array Traversing]**

**Case:1**

<script>

    const person = ["John", "Doe",25];

    let txt = "";

    for (let x in person)

    {

      txt += person[x] + " ";

       console.log(x)

    }

    document.write(txt);

</script>

Note:

1. If we will use **in** keyword then varable x will receive index number like : 0 , 1, 2
2. Document outout is: John Doe 25 and console output is: 0 , 1 , 2

**Case:2**

<script>

    const person = ["John", "Doe",25];

    let txt = "";

    for (let x of person)

    {

      txt += x + " ";

       console.log(x)

    }

    document.write(txt);

</script>

Note:

1. If we will use **of** keyword then varable x will receive value of array like : John , Doe, 25
2. Document outout is: John Doe 25 and console output is: John Doe 25

**Case:3**

<script>

    const person = {fname:"John", lname:"Doe", age:25};

    let txt = "";

    for (let x in person)

    {

        txt += person[x] + " ";

       console.log(x)

    }

    document.write(txt);

</script>

Note:

1. If we will use **in** keyword then varable x will receive index number like : fname , lname, age
2. Document outout is: John Doe 25 and console output is: fname lname age

**Case:4**

<script>

    const person = {fname:"John", lname:"Doe", age:25};

    let txt = "";

    for (let x of person)

    {

      txt += x + " ";

       console.log(x)

    }

    document.write(txt);

</script>

Note:

1. It will get error becaulse **of** keyword is not valid for object, becaue object is not iterable, we can not access value of object with in loop.
2. **Iterables in javascript ? [ignore]**

**Exp(1)**

   <script>

         const fruits = new Map([

                                    ["apples", 500],

                                    ["bananas", 300],

                                    ["oranges", 200]

                                ]);

        let text = "";

        for (const x of fruits)

        {

          text += x + "<br>";

        }

        document.write(text);

    </script>

Output:

apples,500  
bananas,300  
oranges,200

Note: (1) if we will use of keyword than variable x will receive value like: apples,500 than bananas,300

Than oranges,200

**Exp(2)**

<script>

        const letters = new Set(["a","b","c"]);

        let text = "";

        for (const x of letters) {

          text += x + "<br>";

        }

        document.write(text);

    </script>

**Output:**

a

b

c

**Exp(3)**

   <script>

        const name = "welcome";

        let text = ""

        for (const x of name)

        {

          text += x + "<br>";

        }

        document.write(text);

    </script>

**Output:**

w

e

l

c

o

m

e

Note: (1) if we will use **in** keyword than variable x will receive index number like : 0,1,2,3,4,5,6

1. **What is iterator in javascript ? [ignore]**

   <script>

        let numbers = [100,200,300];

        let iter = numbers[Symbol.iterator]();

        console.log(iter.next());

        console.log(iter.next());

        console.log(iter.next());

        console.log(iter.next());

    </script>

output:

{value: 100, done: false}

{value: 200, done: false}

{value: 300, done: false}

{value: undefined, done: true} // because all values have printed

Note:

(1) it is not compulsory to print all value, we can print according to need.

(2) if we want to skip first value then we will use 'iter.next()' instead of first line of 'console.log(iter.next());'

<script>

let numbers = [100,200,300];

let iter = numbers[Symbol.iterator]();

iter.next();

console.log(iter.next());

console.log(iter.next());

console.log(iter.next());

</script>

(3) We can use another statement, it will not affect

<script>

let numbers = [100,200,300];

let iter = numbers[Symbol.iterator]();

iter.next();

console.log(iter.next());

console.log("welcome");

console.log(iter.next());

console.log(iter.next());

</script>

(4) If we want to print only value,then we will use value keyword

<script>

let numbers = [100,200,300];

let iter = numbers[Symbol.iterator]();

iter.next();

console.log(iter.next().value);

console.log(iter.next().value);

console.log(iter.next().value);

</script>

(5) we can check all values has printed or not by done keyword

<script>

let numbers = [100,200,300];

let iter = numbers[Symbol.iterator]();

console.log(iter.next().done);

console.log(iter.next().done);

console.log(iter.next().done);

console.log(iter.next().done);

</script>

(6) we can use loop for prnting:

<script>

let numbers = [100,200,300];

let iter = numbers[Symbol.iterator]();

let result = iter.next();

while(!result.done)

{

console.log(result.value);

result = iter.next();

}

</script>

(7) We can print every character:

<script>

let numbers ="Welcome";

let iter = numbers[Symbol.iterator]();

let result = iter.next();

while(!result.done)

{

console.log(result.value);

result = iter.next();

}

</script>

1. **Read this code ?**

<script>

var arr=[1,2,3,4];

arr.forEach(alert); // output: 1 than 2 than 3 than 4 with in alert box.

arr.forEach(document.write("hello")); // output: hello i.e. it will execute only one time

</script>

1. **What is Web service ? what is the difference b/w web service and web application ?**

Web service is system of software that allows different machienes to connect with each other through network.

|  |  |
| --- | --- |
| **Web Service** | **Web Application** |
| It is used to transfer data b/w web applications. | It is an application that user can access over the internet. |
| It can be access from any language / platform. | It can be access from web browsers. |
| It is communication b/w two devices. | It is Human interaction. |
| There is no GUI. | There is one GUI. |

1. **What is API (Application Programming Interface) ?**

API is a code that enables two software programs to communicate with each other. API defines how a developer should request services from an operating system (OS) or application, and expose data within different contexts across multiple channels.

**API OR Rest API (Representational Application Programming Interface) OR Restful API:**