**Javascript**

It is an object-based scripting language that is lightweight and cross-platform

1)Internal JavaScript

2)External JavaScript

1)Internal Javascript:-

a)Between body tag of Html:-

<!DOCTYPE html>

<html>

<body>

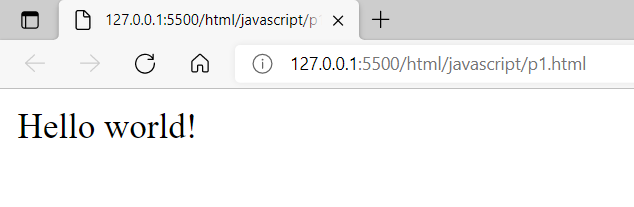
    <script type="text/javascript">

        document.write('Hello world!');

    </script>

</body>

</html>



b)in head tag:-

<!DOCTYPE html>

<html lang="en">

<head>

    <script type="text/javascript">

        function msg(){

            alert('Hello world');

        }

    </script>

</head>

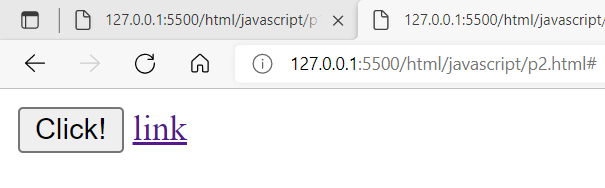
<body>

    <input type="button" value="Click!" onClick="msg();"/>

    <a href="#" onClick="msg();">link</a>

</body>

</html>



2)External JS:-

Step1:- create external js file “message.js”

function msg(){

    alert('Hello world');

}

Step2:-create html file and link js file

<!DOCTYPE html>

<html lang="en">

<head>

    <script type="text/javascript" src="message.js"></script>

</head>

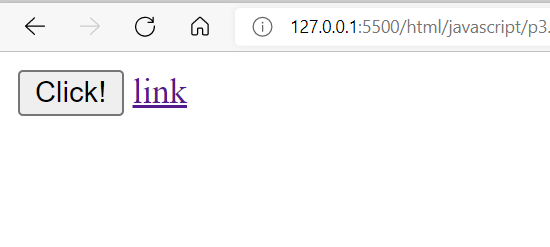
<body>

    <input type="button" value="Click!" onClick="msg();"/>

    <a href="#" onClick="msg();">link</a>

</body>

</html>



**Variables In JS**

* It is a container for storing data value
* Reserved words for variables

Var,let and const

* Identifiers are case-sensitive.

**Var keyword:-**

Ex:-

<!--var keyword-->

<!DOCTYPE html>

<html lang="en">

<head>

</head>

<body>

    <script>

        var val='Hello JavaScript';

      console.log(val);

        //it can be updated

        val="Hello LWC";

        console.log(val);

        val=100;

        console.log(val);

        val=true;

        console.log(val);

//it can be redeclared

        var val=1000;

        console.log(val);

        //scope

        //its supports 2 type of scope

       // 1.global

       var subject="LWC";

       console.log("subject="+subject);

        //2.functional

          function abc(){

              var x=10;

              console.log("X="+x);

          }

          abc();

         // console.log(x);//error=> x is not defined

        //its not support to block level scope

       if(10==10){

            var y=20;

        }

        console.log('Y='+y);

    </script>

</body>

</html>

**Let keyword:-**

<!-- let keyword -->

<!DOCTYPE html>

<html lang="en">

<head>

    <script>

       let x=10;

       console.log('X='+x);

       //it can be updated

       x=20;

       console.log('X='+x);

       //can't be redeclared

     //  let x=30;//error=> x has already been declared

     //scopes

     //its supports global,fuctional and block scope

     //1.global scope

     let y=10;

       console.log('Y='+y);

       //2.functional scope

       function scope(){

           let z=20;

           console.log('Z='+z);

       }

       scope();

      // console.log('Z='+z);//error=> Z is not defined

      //3.block level scope

      if(10==10){

          let val='i am in block level scope';

          console.log('val='+val);

      }

    //  console.log('val='+val);//error=> val is not defined

      var temp1='i am in temp1';

      let temp2='i am in temp2';

      console.log('temp1='+temp1);

      console.log(window.temp1);//output

      console.log(window.temp2);//undefined

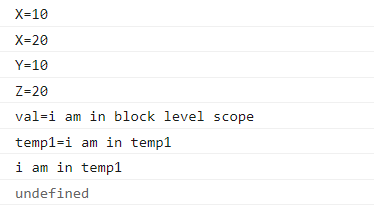
    </script>

</head>

<body>

</body>

</html>

****

**const keyword:-**

<!-- const keyword -->

<!DOCTYPE html>

<html lang="en">

<body>

    <script>

        const PI=3.14;

        console.log(PI);

        //it can not be updated

         //PI=6.14;//error

         //it can not be redeclared

        // const PI=4;//error=> PI has already declared

         //scope

         //it support global,functional and block level scope

    </script>

</body>

</html>

****

**Datatypes**

1.Number

2.String

3.boolean

4.bigint

5.undefined

6.null

7.object

Note:- Rest all the types are objects i.e.

Array,Date,Math,string etc…..

Ex:-

<!-- Data types in JS -->

<!DOCTYPE html>

<html lang="en">

<head>

    <script type="text/javascript">

        //number

        var x=10.5;

        console.log(x);

        console.log(typeof x);//number

        //String

        var str='silvercloud';

        console.log(typeof str);//string

        //boolean

        var isActive=true;

        console.log(typeof isActive);//boolean

        //bigint

        var y=12345n;

        console.log(typeof y);//bigint

        //undefined

        var z;

        console.log(typeof z);//undefined

        //null

        var w=null;

        console.log(typeof w);//object

        //objects

        var obj={name:"abc",age:23,"DOB":"01/06/1989"};

        console.log(obj.age);

        console.log(typeof obj);//object

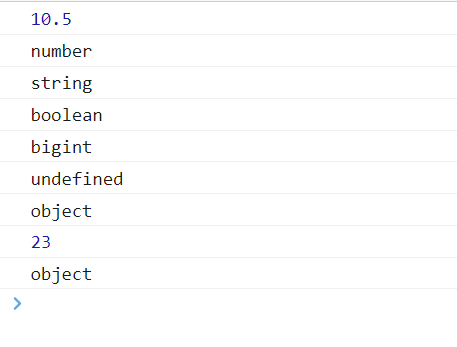
    </script>

</head>

<body>

</body>

</html>



Diff bet Null and undefined:-

Null:-

It’s a special data type which is used to represent nothing or empty value

Its defined explicitly

Its use with typeof then it returns object

Undefined:-

If a variable is declared but not initialized or assigned any value then js automatically initializes its value with undefined

Its use with typeof then returns undefined

Q=>Diff bet == and ===

<!-- Difference between == and === -->

<!DOCTYPE html>

<html lang="en">

<head>

    <script>

        //==

        console.log(3==3);//true

        console.log(3=="3");//true

        //===

        console.log('===');

        console.log(3===3);//true

        console.log(3==="3");//false

    </script>

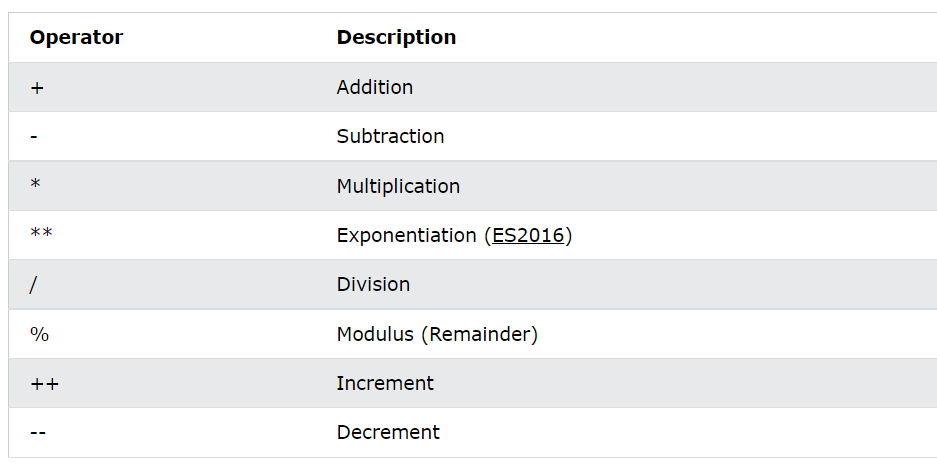
</head>

<body>

</body>

</html>

Operators in JS:-



**Function in JS**

1.No Argument Function:-

<!-- no argument function  -->

<!DOCTYPE html>

<html lang="en">

<head>

    <script>

        function msg(){

            alert('This is alert message');

        }

    </script>

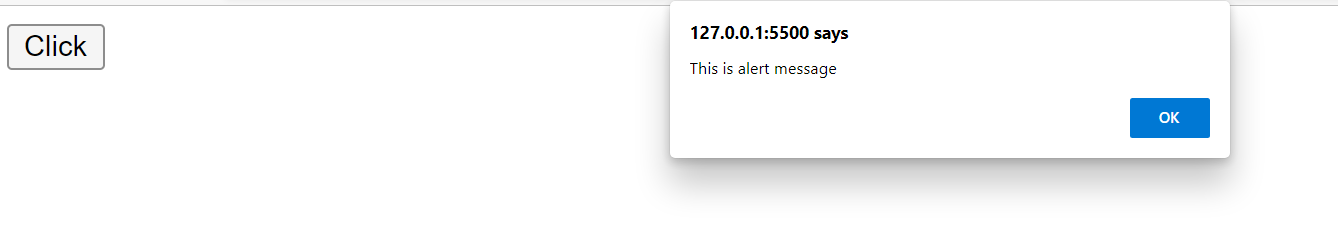
</head>

<body>

   <button onClick="msg();">Click</button>

</body>

</html>



2.Argument Function:-

a.myfun.js

function getCube(number){

    alert('Cube='+(number\*number\*number));

}

b.p11.html

<!DOCTYPE html>

<html lang="en">

<head>

    <script src="myfun.js"></script>

</head>

<body>

    <button onClick="getCube(2);">GET CUBE</button>

</body>

</html>

**JAVASCRIPT OBJECT**

* JS is an object-based language
* Everything is an object in js
* JS is template based not class-based here we don’t create a class to get the object
* But we can directly create an object

There are 3 ways to create an object

1.By Object Literal

2.By creating instance of object directly using new keyword

3.By using an object constructor

1.By Object Literal:-

Syntax:-

objName={property1:value1,property2:value2,.......,property-N:value-N}

<!-- By obejct literal -->

<!DOCTYPE html>

<html lang="en">

<head>

</head>

<body>

    <script>

        studObj={id:101,name:'abc',branch:'IT',age:23};

        document.write("Id:"+studObj.id+" Name:"+studObj.name+" Branch:"+studObj.branch+" Age:"+studObj.age);

    </script>

</body>

</html>



2.By creating Instance:-

<!-- By creating instance -->

<!DOCTYPE html>

<html lang="en">

<body>

   <script>

       var emp=new Object();

       emp.id=101;

       emp.name='XYZ';

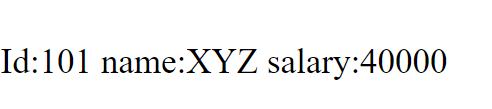
       emp.salary=40000;

       document.write("<br>Id:"+emp.id+" name:"+emp.name+" salary:"+emp.salary);

   </script>

</body>

</html>



Ex2:-

<!-- By creating instance -->

<!DOCTYPE html>

<html lang="en">

<body>

   <script>

       var emp=new Object();

       emp.id=101;

       emp.name='XYZ';

       emp.salary=40000;

       document.write("<br>Id:"+emp.id+" name:"+emp.name+" salary:"+emp.salary);

       var emp2=new Object();

       emp2.id=102;

       emp2.name='Chetan';

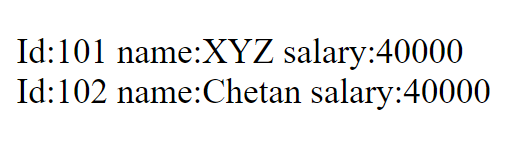
       emp2.salary=40000;

       document.write("<br>Id:"+emp2.id+" name:"+emp2.name+" salary:"+emp2.salary);

   </script>

</body>

</html>



3)By using an object constructor:-

<!-- By using an object constructor -->

<!DOCTYPE html>

<html lang="en">

<head>

    <script>

        function Employee(id,name,salary){

                   this.id=id;

                   this.name=name;

                   this.salary=salary;

        }

    </script>

</head>

<body>

    <script>

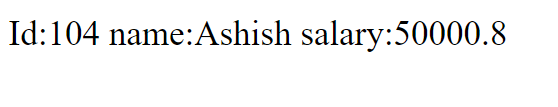
        e=new Employee(104,"Ashish",50000.80);

        document.write("Id:"+e.id+" name:"+e.name+" salary:"+e.salary);

    </script>

</body>

</html>



**Arrays In JS**

Type 1:

Syntax:

Let variableName=[ val1,val2,….val-N];

Ex1:

<!-- Arrays in JS -->

<!DOCTYPE html>

<html lang="en">

<body>

    <script>

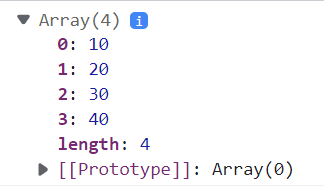
        let arr=[10,20,30,40];

        console.log(arr);

    </script>

</body>

</html>



Ex2:-

<!-- Arrays in JS -->

<!DOCTYPE html>

<html lang="en">

<body>

    <script>

        let arr=[10,20,30,40];

        for (let i=0; i<arr.length; i++){

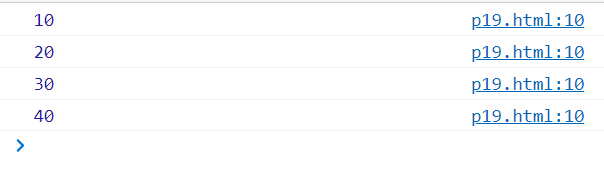
            console.log(arr[i]);

        }

    </script>

</body>

</html>



Ex3:-

<!-- Arrays in JS -->

<!DOCTYPE html>

<html lang="en">

<body>

    <script>

        let arr=[10,20,30,40];

        console.log(arr);

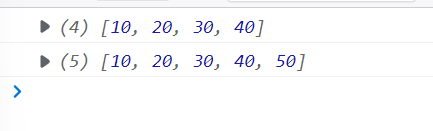
        arr[4]=50;

        console.log(arr);

    </script>

</body>

</html>



Ex4:-

<!-- Arrays in JS -->

<!DOCTYPE html>

<html lang="en">

<body>

    <script>

        let arr=[];

        console.log(arr);

        arr[0]=10;

        arr[1]=20;

        arr[2]=30;

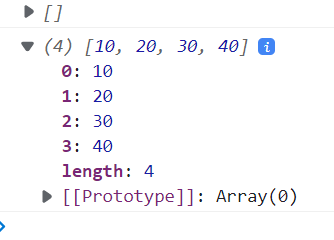
        arr[3]=40;

        console.log(arr);

    </script>

</body>

</html>



Type 2:-

Syntax:-

Let variableName=new array(val1,val2,….val-N);

Ex1:-

<!-- Arrays in JS -->

<!DOCTYPE html>

<html lang="en">

<body>

    <script>

        let arr=new Array(10,20,30,40);

        console.log(arr);

        arr[4]=50;

        console.log(arr);

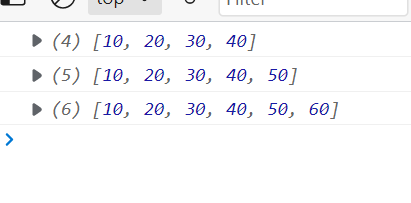
        arr.push(60);

        console.log(arr);

    </script>

</body>

</html>



Ex2:-

<!-- Arrays in JS -->

<!DOCTYPE html>

<html lang="en">

<body>

    <script>

        let arr=new Array(1,2,3,4,5,6,7,8,9,10);

        let newArr=[];

        for(let i=0;i< arr.length;i++){

            if((arr[i]%2)==0){

               newArr.push(arr[i]);

            }

        }

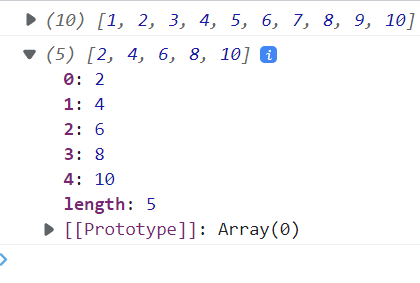
     console.log(arr);

     console.log(newArr);

    </script>

</body>

</html>



Ex3:-

<!DOCTYPE html>

<html lang="en">

<body>

<script>

    let studArr=[

    {name:'Abc',age:23,branch:'IT'},

    {name:'Xyz',age:23,branch:'ETC'},

    {name:'WXY',age:24,branch:'ME'}

    ];

    console.log(studArr);

    for (let i=0;i<studArr.length;i++) {

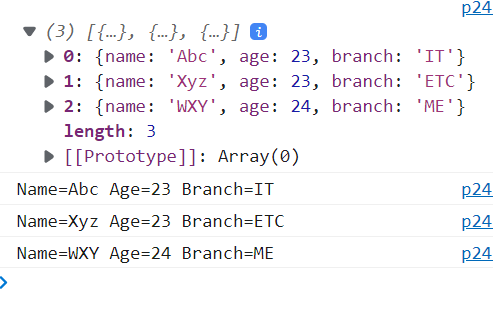
        console.log('Name='+studArr[i].name+' Age='+studArr[i].age+' Branch='+studArr[i].branch);

    }

</script>

</body>

</html>



**Spread Operator**

1. Expanding string:-

convert string into a list of array

<!-- spread operator

  1.    Expanding string

-->

<!DOCTYPE html>

<html lang="en">

<body>

    <script>

        let str='Hello Javascript';

        let charArr=[...str];

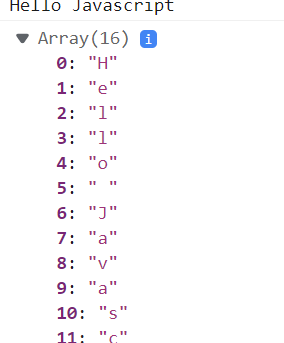
        console.log(str);

        console.log(charArr);

    </script>

</body>

</html>



1. Combining Arrays:-

Combine array or values to array

Ex1:-combining Array

<!-- spread operator

  2.    Combining Arrays:-

-->

<!DOCTYPE html>

<html lang="en">

<body>

    <script>

        let stud1=['Amit','Chetan','Mayur'];

        let stud2=['Pranjal','Kunal'];

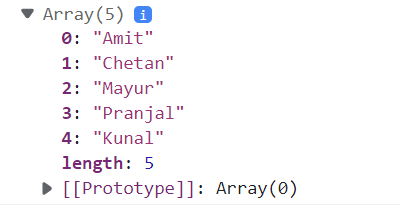
        let finalList=[...stud1,...stud2];

        console.log(finalList);

    </script>

</body>

</html>



Ex2:-Combine array with value

<!-- spread operator

  2.    Combining Array with value:-

-->

<!DOCTYPE html>

<html lang="en">

<body>

    <script>

        let stud1=['Amit','Chetan','Mayur'];

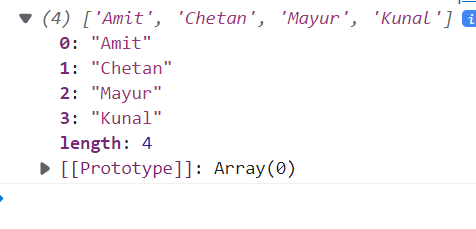
        let finalList=[...stud1,'Kunal'];

        console.log(finalList);

    </script>

</body>

</html>



3.Combining object:-

Combine object or add value to object

Ex:-

<!-- spread operator

  3.    Combining objects:-

-->

<!DOCTYPE html>

<html lang="en">

<body>

    <script>

        let stud1={name:'Abc',age:23,DOB:'01/02/1988'};

        let stud2={name:'xyz',age:25};

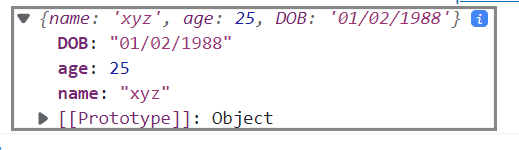
        let finalList={...stud1,...stud2};

        console.log(finalList);

    </script>

</body>

</html>



4.Shallow copy:-

Create new shallow copy of array and object

<!-- spread operator

  4. Shallow Copy:-

-->

<!DOCTYPE html>

<html lang="en">

<body>

    <script>

        console.log('\*\*\*\*\*\*\*\*\*\*\*\*\*\*without shallow copy\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*')

        let arr1= ['X','Y','Z'];

        let arr2=arr1;

        arr2.push('a');

        console.log(arr1);

        console.log(arr2);

        console.log('\*\*\*\*\*\*\*\*\*\*\*\*\*\*with shallow copy\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*')

        let arr3= ['X','Y','Z'];

        let arr4=[...arr3];

        arr4.push('a');

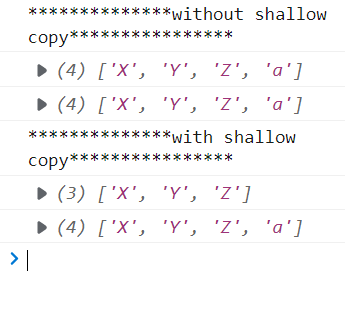
        console.log(arr3);

        console.log(arr4);

    </script>

</body>

</html>



Ex2:-

<!-- spread operator

  4. Shallow Copy:-

-->

<!DOCTYPE html>

<html lang="en">

<body>

    <script>

        console.log('\*\*\*\*\*\*\*\*\*\*\*\*\*\*without shallow copy\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*')

        let arr1= {name:'Abc',age:23};

        let arr2=arr1;

        arr2.age=25;

        console.log(arr1);

        console.log(arr2);

        console.log('\*\*\*\*\*\*\*\*\*\*\*\*\*\*with shallow copy\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*')

        let arr3= {name:'Abc',age:23};

        let arr4={...arr3};

        arr4.age=25;

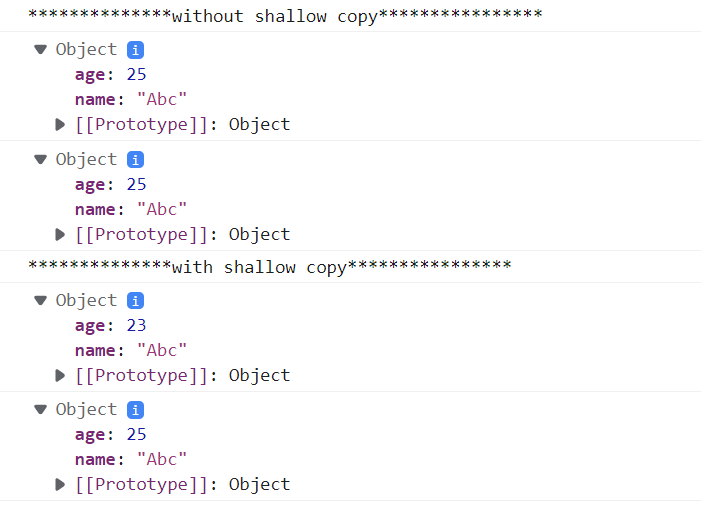
        console.log(arr3);

        console.log(arr4);

    </script>

</body>

</html>



**Destructuring**

It is a special type of syntax that allows us to unpackage arrays and object into a bunch of variables.

=>Array destructuring

=>object destructuring

1)Array Destructuring:-

<!DOCTYPE html>

<html lang="en">

<head>

</head>

<body>

    <script>

        let arr=["Amit","Mayur","Pranjal"];

        console.log("without using destructuring");

        let val1=arr[0];

        let val2=arr[1];

        console.log(val1);

        console.log(val2);

        console.log("with destructuring");

        let[stud1,stud2,stud3]=arr;

        console.log(stud1);

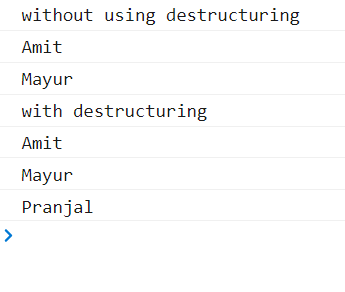
        console.log(stud2);

        console.log(stud3);

    </script>

</body>

</html>



2) object destructuring:-

<!DOCTYPE html>

<html lang="en">

<body>

    <script>

        let student={name:"ABC",age:25,branch:"IT"};

        console.log("Without using Object destructuring");

        let n=student.name;

        let a=student.age;

        let b=student.branch;

        console.log("Name="+n+" and Age="+a+" and Branch="+b);

        console.log("using Object destructuring");

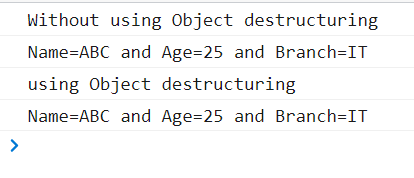
        let{name,age,branch}=student;

        console.log("Name="+name+" and Age="+age+" and Branch="+branch);

    </script>

</body>

</html>



**String Interpolation**

* String interpolation allows us to embed expression in the string
* Template string use back-ticks(``) rather than single or double quotes

Ex1:-

<!-- String interpolation -->

<!DOCTYPE html>

<html lang="en">

<head>

</head>

<body>

    <script>

        let name='ABC';

        let age=25;

        let phone='1234567890';

        console.log(`Name=${name} Age=${age} Phone=${phone}`);

    </script>

</body>

</html>



Ex2:-

<!DOCTYPE html>

<html lang="en">

<head>

    <script>

        function getMenuList(){

            let options=["Apple","Bannana","Mango","Pomgranate","ice cream"];

            let content=`<h4>Menu List!!!!!!!!!!!!!!!!</h4><ul>`;

            for(let i=0;i<options.length;i++){

                content += `<li>${options[i]}</li>`;

            }

            content +=`</ul>`;

            document.getElementById("menuCard").innerHTML=content;

        }

    </script>

</head>

<body>

    <h1><marquee>Welcome to My Restro</marquee></h1>

        <div id="menuCard"></div>

        <button type="button" onclick="getMenuList();">Show Menu Card!</button>

</body>

</html>

