**1. RECURSION AND STACK:**

1. Implement a function to calculate the factorial of a number using recursion

**CODE:**

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=<device-width>, initial-scale=1.0">

    <title>Document</title>

</head>

<body>

    <script>

        let num=+prompt("Enter a number");

        let res=function(num){

            if(num==1)

            return 1;

            else

            return num\*res(num-1);

        }

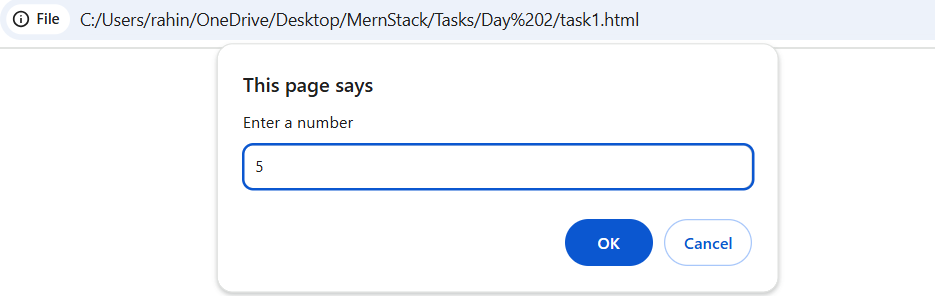
        document.write(res(num));

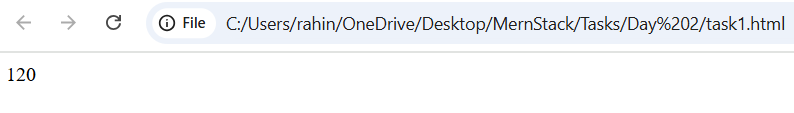
    </script>

</body>

</html>

**OUTPUT:**

****

****

1. Write a recursive function to find the nth Fibonacci number.

**CODE:**

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Document</title>

</head>

<body>

    <script>

        let num=Number(prompt("Enter a number"));

        let fibo=function(num){

            if(num==0)

            return 0;

            else if(num==1)

            return 1;

            else

            return fibo(num-1)+fibo(num-2);

        }

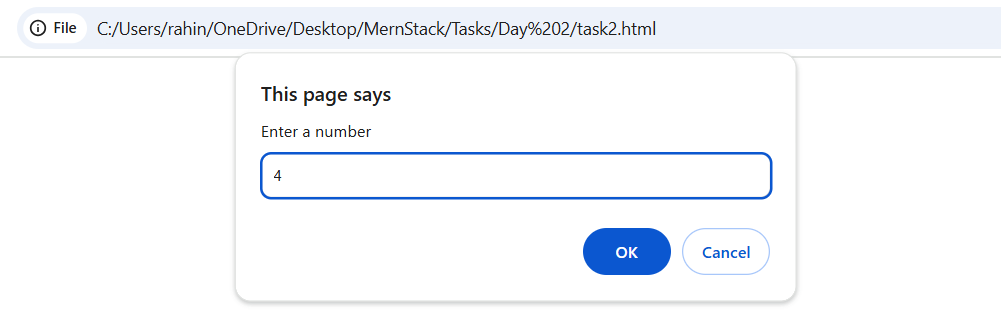
        document.write(fibo(num));

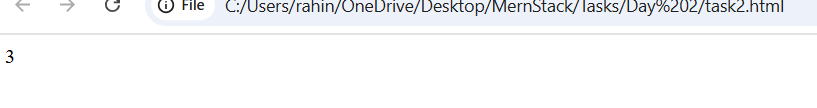
    </script>

</body>

</html>

**OUTPUT:**

****

****

1. Create a function to determine the total number of ways one can climb a staircase with 1, 2, or 3 steps at a time using recursion.

**CODE:**

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Document</title>

</head>

<body>

    <script>

        let num=Number(prompt("Enter a number"));

        let climb=function(num){

            if(num==0 || num==1)

            return 1;

            else if(num==2)

            return 2;

            else

            return climb(num-1)+climb(num-2)+climb(num-3);

        }

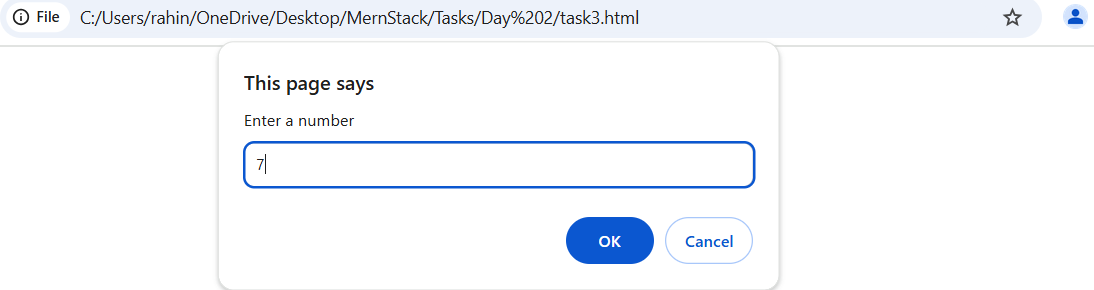
        document.write(climb(num));

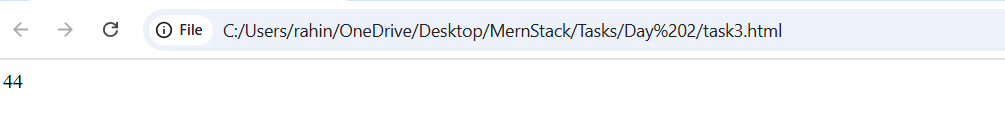
    </script>

</body>

</html>

**OUTPUT:**

****

****

1. Write a recursive function to flatten a nested array structure.

**CODE:**

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Document</title>

</head>

<body>

    <script>

        function helper(arr,n)

        {

            if(n==0)

            return arr;

            arr=arr.flat();

            return helper(arr,n-1);

        }

        let arr=[1,2,[3,4],[5,6],7];

        let n=arr.length;

        let arr1=helper(arr,n);

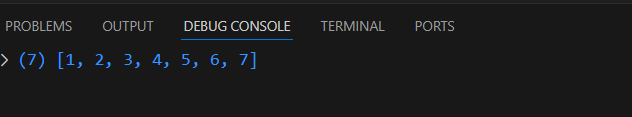
        console.log(arr1);

    </script>

</body>

</html>

**OUTPUT:**

****

1. Implement the recursive Tower of Hanoi solution.

**CODE:**

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Document</title>

</head>

<body>

    <script>

        let num=Number(prompt("Enter a number"));

        let tower=function(num,source,desti,helper){

            if(num==0)

            return 0;

            tower(num-1,source,helper,desti);

            document.write("Move disk no "+num+" from "+source+" to "+desti+"<br>");

            tower(num-1,helper,desti,source);

        }

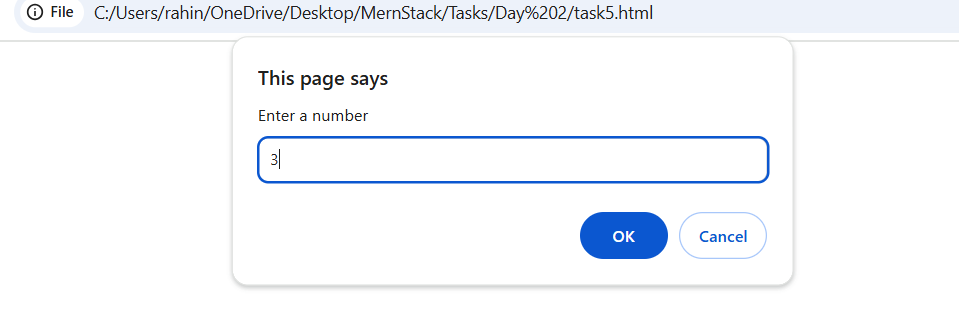
        tower(num,1,3,2);

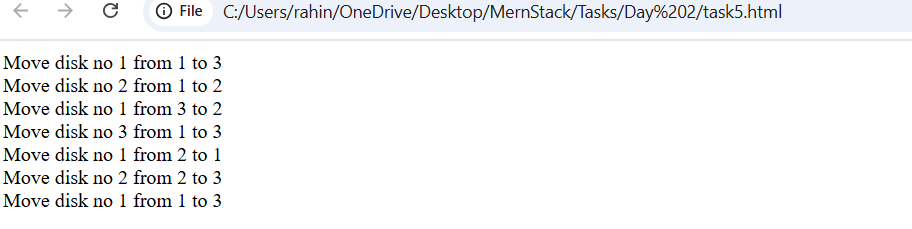
    </script>

</body>

</html>

**OUTPUT:**

****

****

**2. JSON AND VARIABLE LENGTH ARGUMENTS/SPREAD SYNTAX:**

1. Write a function that takes an arbitrary number of arguments and returns their sum.

**CODE:**

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Document</title>

</head>

<body>

    <script>

        let sum=0;

        let num=+prompt("Execution of For loop");

        function Exe(val)

        {

            sum+=val;

            return sum;

        }

        for(let i=0;i<num;i++)

        {

            let val=+prompt("Enter a number");

            var out=Exe(val);

        }

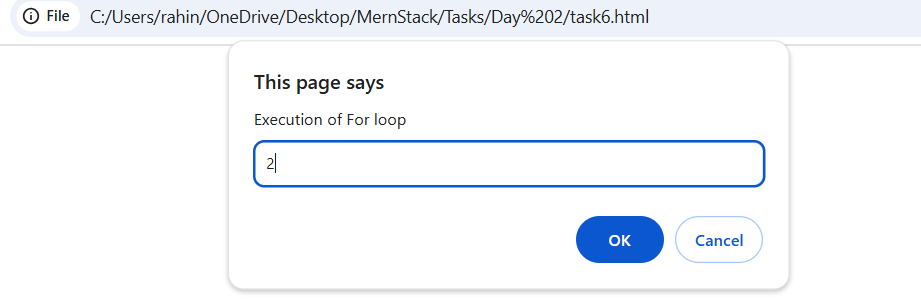
        document.write("The sum is:"+out);

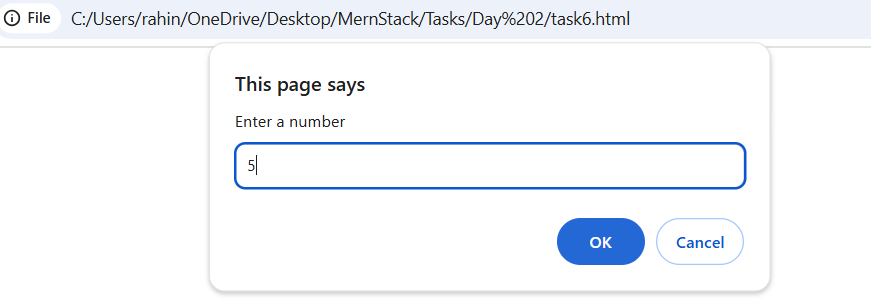
    </script>

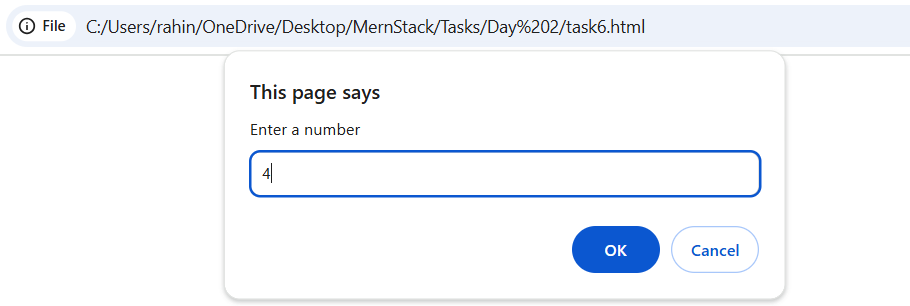
</body>

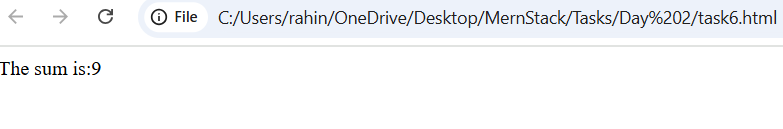
</html>

**OUTPUT:**

****

****

****

****

1. Modify a function to accept an array of numbers and return their sum using the spread syntax.

**CODE:**

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Document</title>

</head>

<body>

    <script>

        let sum=0;

        let num=+prompt("Execution of For loop");

        function Exe(val)

        {

            sum+=val;

            return sum;

        }

        var arr = Array();

        for(let i=0;i<num;i++)

        {

            arr.push(Number(prompt("Enter a number")));

        }

        for(let i=0;i<num;i++)

        {

            var out=Exe(arr[i]);

        }

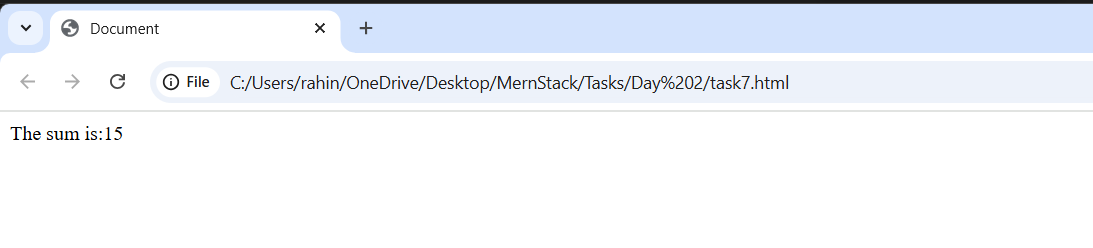
        document.write("The sum is:"+out);

    </script>

</body>

</html>

**OUTPUT:**

****

1. Create a deep clone of an object using JSON methods.

**CODE:**

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Document</title>

</head>

<body>

    <script>

        let obj={

            Name:"John",

            Age:19,

            Department:"ECE"

        }

        let clone=JSON.parse(JSON.stringify(obj));

        clone.Name="Smith";

        console.log(obj);

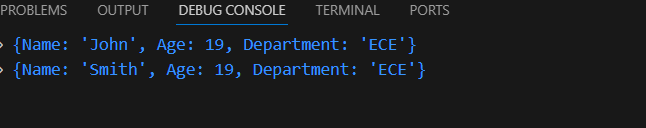
        console.log(clone);

    </script>

</body>

</html>

**OUTPUT:**

****

1. Write a function that returns a new object, merging two provided objects using the spread syntax.

**CODE:**

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Document</title>

</head>

<body>

    <script>

        function merge(object1,object2)

        {

            return {...object1,...object2};

        }

        let object1={

            Name:"John",

            Age:19

        };

        let object2={

            Dept:"ECE"

        };

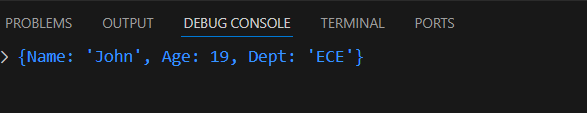
        console.log(merge(object1,object2));

    </script>

</body>

</html>

**OUTPUT:**

****

1. Serialize a JavaScript object into a JSON string and then parse it back into an object.

**CODE:**

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Document</title>

</head>

<body>

    <script>

        let object1={

            Name:"John",

            Age:10

        };

        let str=JSON.stringify(object1);

        console.log(str);

        console.log(typeof(str));

        let obj=JSON.parse(str);

        console.log(obj);

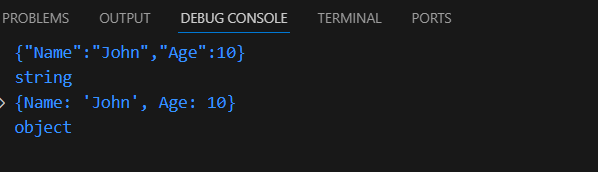
        console.log(typeof(obj));

    </script>

</body>

</html>

**OUTPUT:**

****

**3. CLOSURE:**

1. Create a function that returns another function, capturing a local variable.

**CODE:**

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Document</title>

</head>

<body>

    <script>

        function display(){

            let message="Hi";

            document.write("Display Function <br>");

            function sub(message){

                document.write("Hello! and "+message+" to everyone.");

            }

            sub(message);

        }

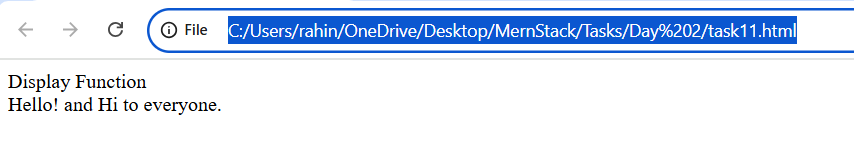
        display();

    </script>

</body>

</html>

**OUTPUT:**

****

1. Implement a basic counter function using closure, allowing incrementing and displaying the current count.

**CODE:**

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Document</title>

</head>

<body>

    <script>

        let count=0;

        function display(count)

        {

            if(count==num)

            return;

            count++;

            function res(count)

            {

                document.write(count+" ");

                display(count);

            }

            res(count);

        }

        let num=+prompt("Enter a number");

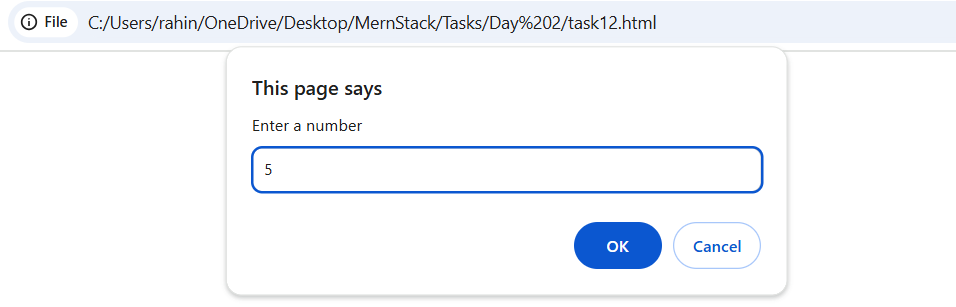
        display(count);

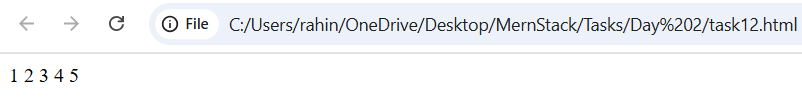
    </script>

</body>

</html>

**OUTPUT:**

****

****

1. Write a function to create multiple counters, each with its own separate count.

**CODE:**

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Document</title>

</head>

<body>

    <script>

        function CreateCounter(){

            let count=0;

            return function()

            {

                count++;

                document.write(count+" ");

            }

        }

            const counter1 = CreateCounter();

            const counter2 = CreateCounter();

            counter1();

            counter1();

            document.write("<br>");

            counter2();

            counter2();

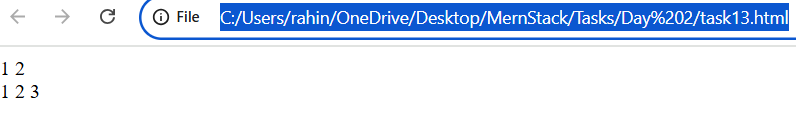
            counter2();

    </script>

</body>

</html>

**OUTPUT:**

****

1. Use closures to create private variables within a function.

**CODE:**

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Document</title>

</head>

<body>

    <script>

        function display(){

            let message="Hi";

            function res(){

                document.write(message+" to everyone.");

            }

            res();

        }

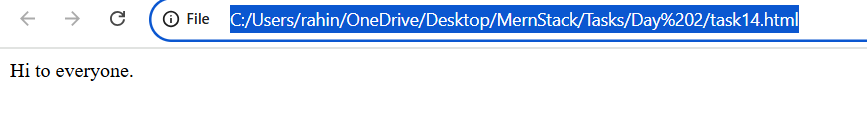
        display();

    </script>

</body>

</html>

**OUTPUT:**

****

1. Build a function factory that generates functions based on some input using closures.

**CODE:**

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Document</title>

</head>

<body>

    <script>

        function createMathOperation(operation) {

             return function(a, b) {

             if(operation=='add') {

                return a + b;

             }

             if(operation=='sub')

             {

                return a-b;

             }

            }

        }

        const add=createMathOperation('add');

        console.log(add(10,4));

        const sub=createMathOperation('sub');

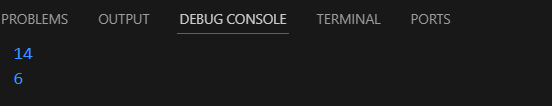
        console.log(sub(10,4));

    </script>

</body>

</html>

**OUTPUT:**

****

**4. PROMISE, PROMISES CHAINING:**

1. Create a new promise that resolves after a set number of seconds and returns a greeting.

**CODE:**

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Document</title>

</head>

<body>

    <script>

        let promise=new Promise(function(resolve)

        {

            setTimeout(()=>resolve("Hello!"),1000);

        });

        promise.then(

            result=>alert(result)

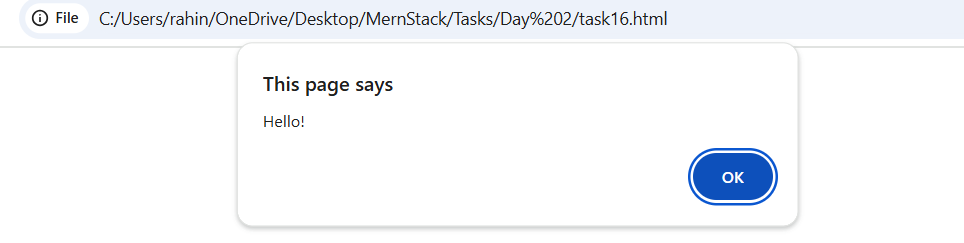
        );

    </script>

</body>

</html>

**OUTPUT:**

****

1. Fetch data from an API using promises, and then chain another promise to process this data.

**CODE:**

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Document</title>

</head>

<body>

    <script>

        let url='https://jsonplaceholder.typicode.com/users/1';

        async function  res(url){

        let response=await fetch(url);

        if(response.ok){

            let jso=await response.text();

            document.write(jso);

        }

        else{

            alert("HTTP -Error:"+response.status);

        }

       };

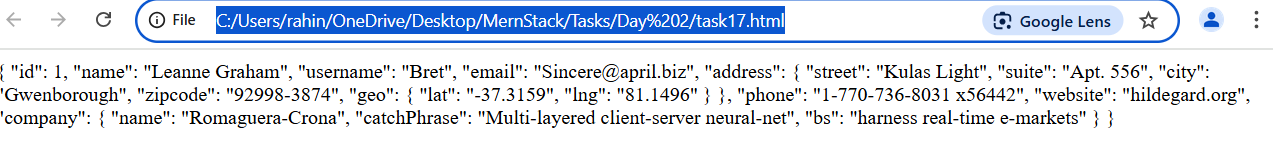
       res(url);

    </script>

</body>

</html>

**OUTPUT:**

****

1. Create a promise that either resolves or rejects based on a random number.

**CODE:**

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Document</title>

</head>

<body>

    <script>

        let num=+prompt("Enter a nuber");

         let promise=new Promise(function(resolve,reject)

        {

            if(num>5)

            setTimeout(()=>resolve("Hello"),1000);

            else

            document.write("Error");

        });

        promise.then(

            result=>alert(result),

            resolve=>alert(resolve)

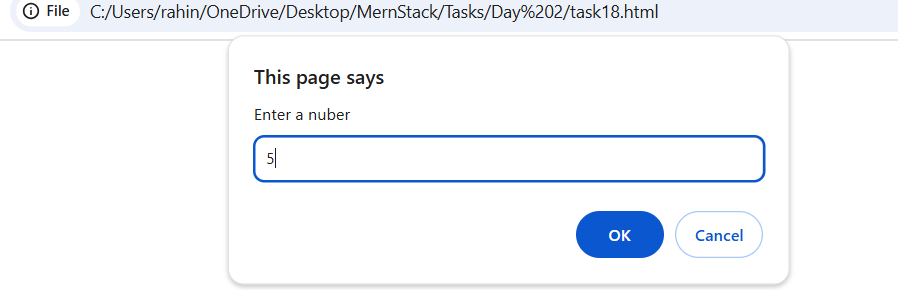
        );

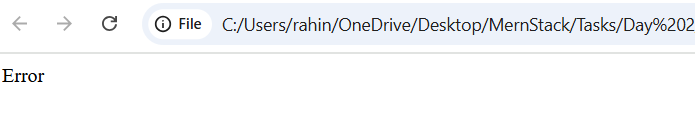
    </script>

</body>

</html>

**OUTPUT:**

****

****

**5.MODULES INTRODUCTION, EXPORT AND IMPORT:**

1. Create a module that exports a function, a class, and a variable.

**CODE:**

export function sayHi(){

    let res="Hi!";

    console.log(res);

}

export class Details

    {

        constructor(name)

        {

            this.name=name;

        }

        det()

        {

            console.log("Name: "+ this.name);

        }

    }

import {sayHi,Details} from './task26.js';

sayHi();

let D=new Details("Kavina");

D.det();

**OUTPUT:**

****

1. Import the module in another JavaScript file and use the exported entities.

**CODE:**

export function sayHi(){

    let res="Hi!";

    console.log(res);

}

export class Details

    {

        constructor(name)

        {

            this.name=name;

        }

        det()

        {

            console.log("Name: "+ this.name);

        }

    }

import {sayHi,Details} from './task26.js';

sayHi();

let D=new Details("Kavina");

D.det();

**OUTPUT:**

****

1. Use named exports to export multiple functions from a module.

**CODE:**

export function fun(){

    let Name="Rahini";

    return Name;

}

export function fun1(){

    let Age=20;

    return Age;

}

export function fun2(){

    let City="Coimbatore";

    return City;

}

import {fun,fun1,fun2} from './task28.js';

console.log(fun());

console.log(fun1());

console.log(fun2());

**OUTPUT:**

****

1. Use named imports to import specific functions from a module.

**CODE:**

export function fun(){

    let Name="Rahini";

    return Name;

}

export function fun1(){

    let Age=20;

    return Age;

}

export function fun2(){

    let City="Coimbatore";

    return City;

}

import {fun,fun1,fun2} from './task28.js';

console.log(fun());

console.log(fun1());

console.log(fun2());

**OUTPUT:**

****

1. Use default export and import for a primary function of a module.

**CODE:**

export default function detail(){

    let name="Rahini";

    return name;

}

import detai from './task30.js';

console.log(detai());

**OUTPUT:**

****

1. **BROWSER: DOM BASICS:**
2. Select an HTML element by its ID and change its content using JavaScript.

**CODE:**

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Document</title>

</head>

<body>

    <p id="myparagraph">Original Text</p>

    <button onclick="modify()">Onclick</button>

    <script>

        function modify(){

            document.getElementById("myparagraph").innerHTML="Changed Text";

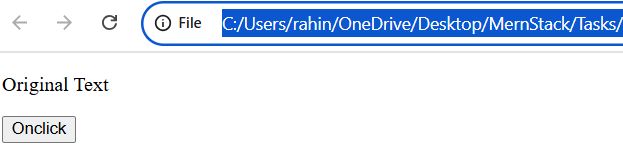
        }

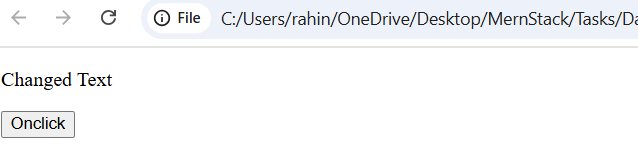
    </script>

</body>

</html>

**OUTPUT:**

****

****

1. Attach an event listener to a button, making it perform an action when clicked.

**CODE:**

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Document</title>

</head>

<body>

    <p id="myparagraph">Original Text</p>

    <button onclick="modify()">Onclick</button>

    <script>

        function modify(){

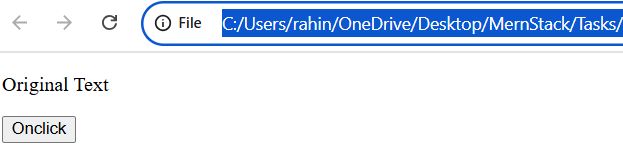
            document.getElementById("myparagraph").innerHTML="Changed Text";

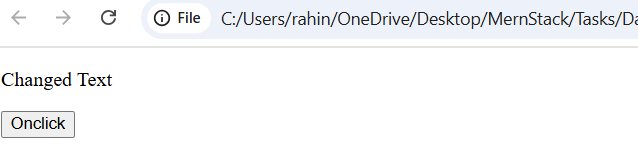
        }

    </script>

</body>

**OUTPUT:**

****

****

1. Create a new HTML element and append it to the DOM.

**CODE:**

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Document</title>

</head>

<body>

    <button id="addElement">Add Element</button>

    <div id="container"></div>

    <script>

        const button = document.getElementById("addElement");

        const container = document.getElementById("container");

        button.addEventListener('click',function(){

            const newEle=document.createElement("p");

            newEle.textContent="New Element Received";

            container.appendChild(newEle);

        }

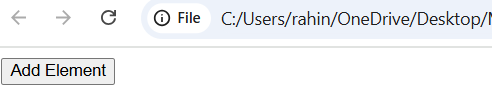
    );

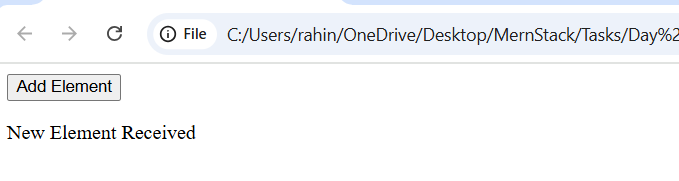
    </script>

</body>

</html>

**OUTPUT:**

****

****

1. Implement a function to toggle the visibility of an element.

**CODE:**

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Document</title>

</head>

<body>

    <p id="paragraph">Hello</p>

    <button id="con" >Hide</button>

    <script>

        let count=0;

        let button=document.getElementById("con");

        button.addEventListener('click',function(){

        let res=document.getElementById("paragraph");

        if(count%2==0)

        {

            res.textContent="";

        }

        else

        {

            res.textContent="Hello";

        }

        count++;

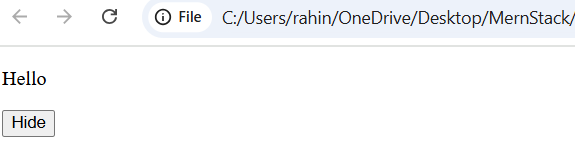
    });

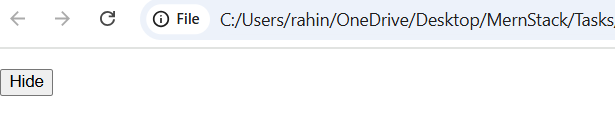
    </script>

</body>

</html>

**OUTPUT:**

****

****

1. Use the DOM API to retrieve and modify the attributes of an element.

**CODE:**

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Document</title>

</head>

<body>

    <img id="myImage" src="old-photo.jpg" alt="Old Image" width="300">

    <button onclick="modifyAttributes()">Change Image</button>

    <script>

        function modifyAttributes() {

            const img = document.getElementById("myImage");

            //img.src="new-photo.jpg";

            img.setAttribute("src", "new-photo.jpg");

            img.setAttribute("alt", "New Image");

            img.setAttribute("width", "500");

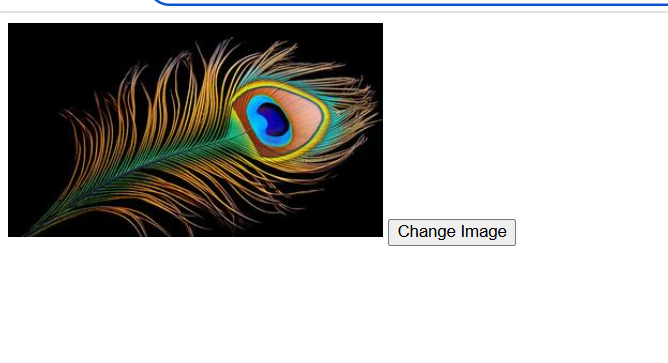
        }

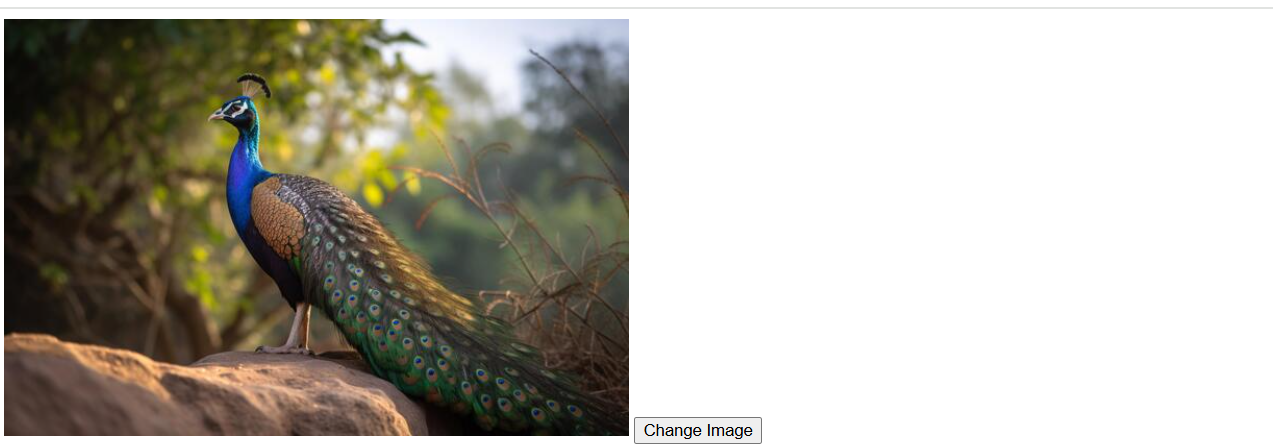
    </script>

</body>

</html>

**OUTPUT:**

****

****