**RECURSION AND STACK:**

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

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

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

    <title>task1</title>

</head>

<body>

    <script>

        let num = Number(prompt("Enter the number",0));

        let fact = function(num){

            if(num == 1) return 1;

            return num\*fact(num-1);

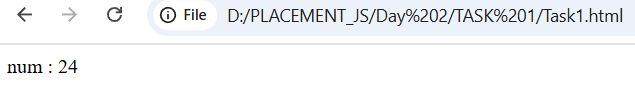
        }

        document.write(`num : ${fact(num)}`);

    </script>

</body>

</html>

****

**TASK 2: Write a recursive function to find the nth Fibonacci number.**

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

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

    <title>Task 1</title>

</head>

<body>

    <script>

        let num = Number(prompt("Enter a number : ",0));

        function fibo(num){

            if(num == 0) return 0;

            if(num == 1) return 1;

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

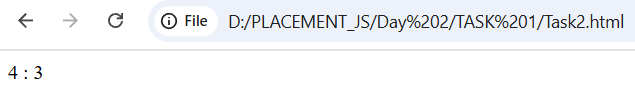
        }

        document.write(num+" : "+fibo(num));

    </script>

</body>

</html>

****

**TASK 3: 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.**

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

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

    <title>Task3</title>

</head>

<body>

    <script>

        let num = +(prompt("Enter steps:",0));

        document.write("No of possible ways : ",solve(num));

        function solve(num){

            if(num == 0) return 1;

            if(num < 0) return 0;

            let s1 = solve(num-1);

            let s2 = solve(num-2);

            let s3 = solve(num-3);

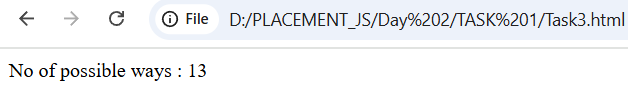
            return s1+s2+s3;

        }

    </script>

</body>

</html>

****

**TASK 4: Write a recursive function to flatten a nested array structure.**

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

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

    <title>Task 4</title>

</head>

<body>

    <script>

        let arr = [1,2,3,[5,[7,8,9],6,7],9,[10,22,56],65];

        //console.log(arr.flat(Infinity)); //nested arr to 1D array

        one\_D(arr,0);

        function one\_D(arr,l){

            if(l == arr.length) return;

            arr = arr.flat();

            one\_D(arr,l+1);

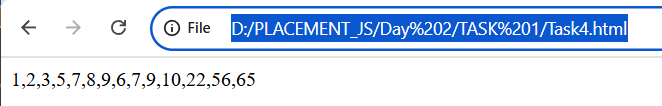
        }

        document.write(arr);

    </script>

</body>

</html>

****

**TASK 5: Implement the recursive Tower of Hanoi solution.**

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

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

    <title>Task5</title>

</head>

<body>

    <script>

        function tower(n,source,destination,helper){

        if(n == 0) return;

        tower(n-1,source,helper,destination);

        document.write(`Move ${n} disk from ${source} to ${destination} <br>`);

        tower(n-1,helper,destination,source);

    }

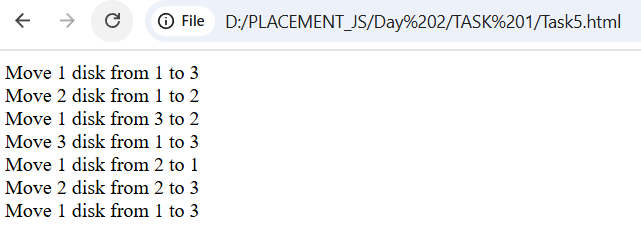
    let num = +(prompt("Enter the blocks:",0));

    tower(num,1,3,2);

    </script>

</body>

</html>

****

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

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

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

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

    <title>Task1</title>

</head>

<body>

    <script>

        var r\_sum = 0;

        function sum(num){

            r\_sum+=num;

            return r\_sum;

        }

        let n\_a = Number(prompt("Enter no of Arguments",0));

        for(let i = 0 ; i < n\_a ; i++){

            let p = +prompt("Enter the argument");

            res = sum(p);

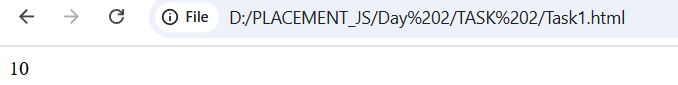
        }

        document.write(res);

    </script>

</body>

</html>

****

**TASK 2: Modify a function to accept an array of numbers and return their sum using the spread syntax.**

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

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

    <title>Task2</title>

</head>

<body>

    <script>

        let r\_sum=0;

        let sum = function(...num){

            for(let i of num)

            r\_sum +=i;

            return r\_sum;

        }

        let arr = Array();

        let arr\_size = +(prompt("Enter array size :",0));

        for(let i = 0; i < arr\_size ; i++){

            arr.push(+(prompt("Enter a number : ",0)));

        }

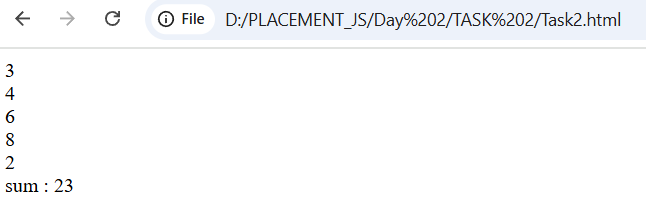
        let res = sum(...arr);

        document.write("sum : "+res);

    </script>

</body>

</html>

****

**TASK 3: Create a deep clone of an object using JSON methods.**

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

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

    <title>Task3</title>

</head>

<body>

    <script>

        let user = {

        name : "Kavi",

        age : 19,

        dept : "ece",

        section : 'b'

        }

        /\*let clone\_user = user;

        document.write(clone\_user.name);\*/

        let clone\_user = structuredClone(user);

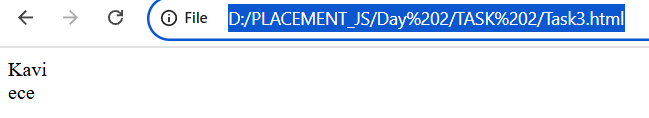
        document.write(clone\_user.name+"<br>");

        document.write(clone\_user.dept);

    </script>

</body>

</html>

****

**TASK 4: Write a function that returns a new object, merging two provided objects using the spread syntax.**

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

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

    <title>Task4</title>

</head>

<body>

    <script>

        function mergeObj(obj1,obj2){

            return {...obj1,...obj2};

        }

        let obj1 = {

            name:"Kavina",

            age:19

        };

        let obj2 = {

            college :"KCE",

            dept : "ECE"

        };

        let mergedObj = mergeObj(obj1,obj2);

        document.write(mergedObj.name +"<br>"+mergedObj.college);

    </script>

</body>

</html>



**TASK 5: Serialize a JavaScript object into a JSON string and then parse it back into an object.**

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

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

    <title>Task5</title>

</head>

<body>

    <script>

        let user = {

        name : "Kavi",

        age : 19,

        dept : "ece",

        section : 'b'

       }

       let j\_user = JSON.stringify(user);

       //document.write(user + " type : " + typeof user);

       document.write(j\_user + " , type : " + typeof j\_user+"<br>");

       let o\_user = JSON.parse(j\_user);

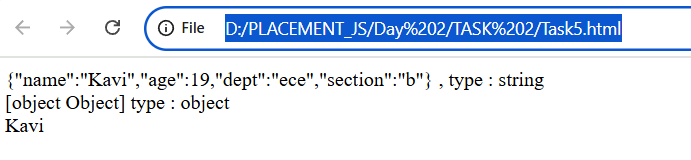
       document.write(o\_user + " type : " + typeof o\_user+"<br>");

       document.write(o\_user.name);

    </script>

</body>

</html>

****

**3. CLOSURE:**

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

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

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

    <title>Task1</title>

</head>

<body>

    <script>

        let give = function(n){

        let num = n-1;

        return reduced\_give(num);

    }

    function reduced\_give(numm){

        return numm-1;

    }

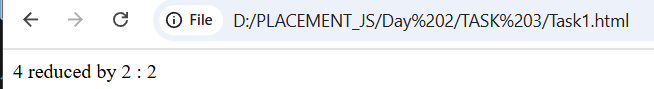
    let inpp = +(prompt("Enter number : ",0));

    document.write(inpp + " reduced by 2 : " +give(inpp));

    </script>

</body>

</html>

****

**TASK 2: Implement a basic counter function using closure, allowing incrementing and displaying the current count.**

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

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

    <title>Task2</title>

</head>

<body>

    <script>

        let start=+prompt("Enter start:");

        let end=+prompt("Enter end:");

        function counter(start,end){

            if(start > end) return;

            function display(){

                document.write(start+" ");

                counter(start+1,end);

            }

            display();

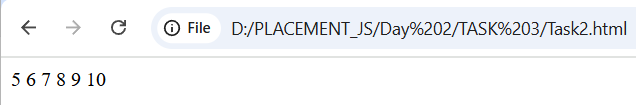
        }

        counter(start,end);

    </script>

</body>

</html>

****

**TASK 3: Write a function to create multiple counters, each with its own separate count.**

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

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

    <title>Task3</title>

</head>

<body>

    <script>

        //let count = 0;

        function counter(){

            let count = 0;

            return function(){

                count++;

                document.write(count+" ");

            };

        }

        let i1 = counter();

        let i2 = counter();

        i1();

        i1();

        i1();

        i1();

        i2();

        i2();

    </script>

</body>

</html>

****

**TASK 4: Use closures to create private variables within a function.**

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

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

    <title>Task4</title>

</head>

<body>

    <script>

        let fun = function(){

            let p\_v = 10;

            fun2();

            function fun2(){

                document.write("Private variable inside function2 closure : "+p\_v+"<br>");

            }

            return function(){

                document.write("Private variable inside return closure : "+p\_v+"<br>");

            }

        };

        //fun();

        let returned\_fun=fun();

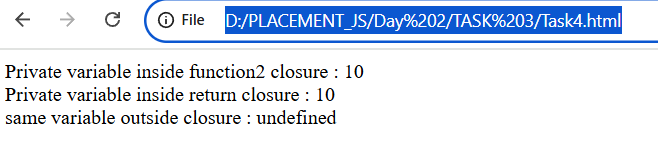
        returned\_fun();

        document.write("same variable outside closure : "+typeof p\_v+"<br>");

    </script>

</body>

</html>

****

**TASK 5: Build a function factory that generates functions based on some input using closures.**

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

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

    <title>Task5</title>

</head>

<body>

    <script>

        function fun(name){

            name+='a';

            return function(){

                document.write("name + a : "+name+"<br>");

            }

        }

        let r\_fun=fun("Kavin");

        r\_fun();

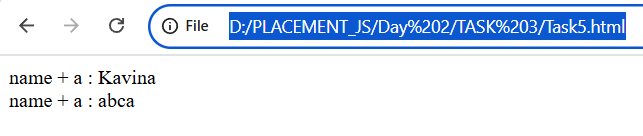
        r\_fun = fun("abc");

        r\_fun();

    </script>

</body>

</html>

****

**4. PROMISE, PROMISES CHAINING:**

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

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

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

    <title>Task1</title>

</head>

<body>

    <script>

        let pro = new Promise(function(resolve,reject){

            setTimeout(() => {

                document.write("Welcome...");

            }, 4000);

        });

        // let pro = new Promise(function(resolve,reject){

        //     setTimeout(() => {

        //         document.write(`Welcome Kavina !!!!`);

        //     }, 7000);

        //     document.write("print after 7s <br>");

        //     if(true) resolve("successfull");

        //     else reject("not printed");

        // });

        // pro.then((result) => document.write(result),(error) => document.write(error));

        // let name = prompt("Enter your name :");

        // let pro = new Promise(function(resolve,reject){

        //         setTimeout(() => {

        //             resolve(name);

        //         }, 4000);

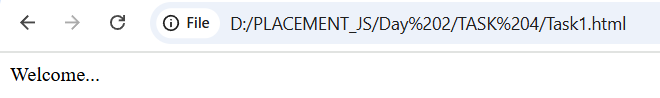
        //     });

        // pro.then((name) => {document.write(`Hello ${name}!!!`);});

    </script>

</body>

</html>

****

**TASK 2: Fetch data from an API using promises, and then chain another promise to process this data.**

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

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

    <title>Task2</title>

</head>

<body>

    <script>

       let pro = fetch("https://jsonplaceholder.typicode.com/todos/1");

       pro.then(Response => Response.json(),console.error()).then((data) => console.log(data));

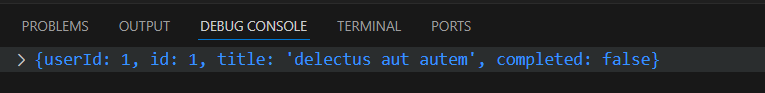
    //    let pro = fetch("https://jsonplaceholder.typicode.com/todos/1");

    //    pro.then(Response => Response.json(),console.error()).then((data) => document.write(JSON.stringify(data)));

    </script>

</body>

</html>

****

**TASK 3: Create a promise that either resolves or rejects based on a random number.**

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

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

    <title>Task3</title>

</head>

<body>

    <script>

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

        let pro = new Promise(function(resolve,reject){

            if(num % 2 == 0) resolve();

            else reject();

        });

        pro.then(() => document.write("resolved:Even number"),() => document.write("rejected:Odd number"));

        // new Promise(function(resolve,reject){

        //     if(num % 2 == 0) resolve();

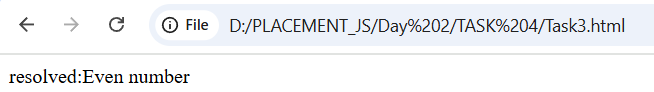
        //     else reject();

        // }).then(() => document.write("resolved:Even number"),() => document.write("rejected:Odd number"));

    </script>

</body>

</html>

****

**TASK 4: Use Promise.all to fetch multiple resources in parallel from an API.**

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

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

    <title>Task4</title>

</head>

<body>

    <script>

        const urls = [

    'https://jsonplaceholder.typicode.com/posts/1',

    'https://jsonplaceholder.typicode.com/posts/2',

    'https://jsonplaceholder.typicode.com/posts/3'

    ];

    let fetch\_url = urls.map((urls) => fetch(urls).then(Response =>{if(Response.ok){

        return Response.json();

        }

        else

        console.error();

    }));

    //Printing index by index in arr

    Promise.all(fetch\_url).then(data\_arr => {

        for(let v of data\_arr){

        document.write(JSON.stringify(v)+"<br>");

    }});

    // printing as whole arr

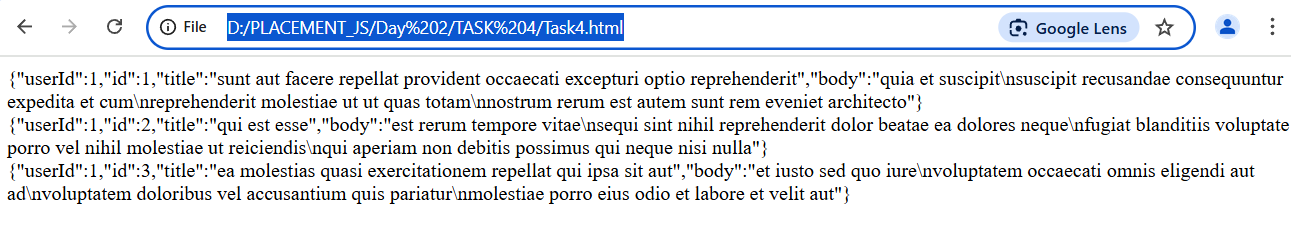
    // Promise.all(fetch\_url).then(data\_arr =>

    //     document.write(JSON.stringify(data\_arr)));

    </script>

</body>

</html>

****

**TASK 5: Chain multiple promises to perform a series of asynchronous actions in sequence.**

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

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

    <title>Task5</title>

</head>

<body>

    <script>

        function fun1(name){

            return new Promise(function(resolve,reject){

                document.write("function 1,name : "+name+"<br>");

                resolve(name+'a');

            });

        }

        function fun2(name){

            return new Promise(function(resolve,reject){

                document.write("function 2,name : "+name+"<br>");

                resolve(name+'a');

            });

        }

        function fun3(name){

            return new Promise(function(resolve,reject){

                document.write("function 3,name : "+name+"<br>");

                resolve(name+'a');

            });

        }

        let name = prompt("Enter string");

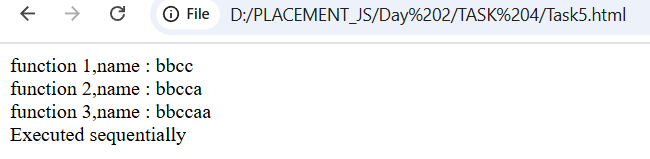
        let pro = fun1(name);

        pro.then((res) => fun2(res),console.error()).then((res) => fun3(res)).then((res) => document.write("Executed sequentially"));

    </script>

</body>

</html>

****

**5. ASYNC/AWAIT:**

**TASK 1: Rewrite a promise-based function using async/await.**

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

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

    <title>Task1</title>

</head>

<body>

    <script>

        let name = prompt("Enter your name : ");

        async function fun(){

            return await new Promise(function(resolve,reject){

                document.write("Hello "+name+"<br>");

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

                document.write(`Entered val : ${val}<br>`);

            });

        }

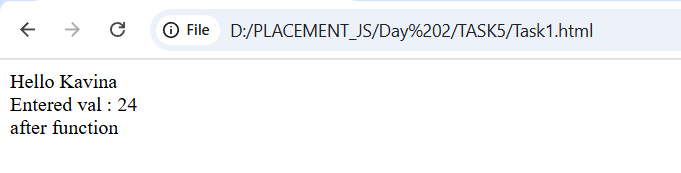
        fun().then();

        document.write("after function");

    </script>

</body>

</html>

****

**TASK 2: Create an async function that fetches data from an API and processes it.**

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

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

    <title>Task2</title>

</head>

<body>

    <script>

        async function f\_api(){

            return await fetch("https://jsonplaceholder.typicode.com/todos/1").then(Response => Response.json(),console.error()).then(data => console.log(data));

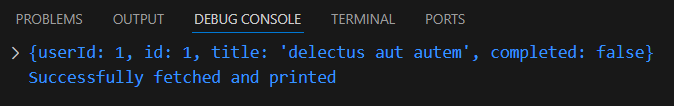
        }

        f\_api().then(() => console.log("Successfully fetched and printed"),console.error());

    </script>

</body>

</html>

****

**TASK 3: Implement error handling in an async function using try/catch.**

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

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

    <title>Task3</title>

</head>

<body>

    <script>

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

        async function sum(){

            return await new Promise(function(resolve,reject){

                try{

                    document.write(num/a);

                }

                catch(error){

                    document.write(error+"<br>");

                }

            });

        }

        // async function sum(){

        //     try{

        //         return await new Promise(function(resolve,reject){

        //         document.write(num/a);

        //     });

        //     }

        //     catch(error){

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

        //     }

        // }

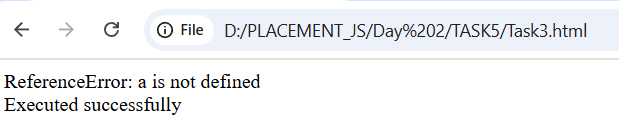
        sum().then();

        document.write("Executed successfully"+"<br>");

    </script>

</body>

</html>

****

**TASK 4: Use async/await in combination with Promise.all.**

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

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

    <title>Task4</title>

</head>

<body>

    <script>

        const urls = [

    'https://jsonplaceholder.typicode.com/posts/1',

    'https://jsonplaceholder.typicode.com/posts/2',

    'https://jsonplaceholder.typicode.com/posts/3'

    ];

    let fetch\_url = urls.map((urls) => fetch(urls).then(Response =>{

        if(Response.ok){

        return Response.json();

        }

        else

        console.error();

    }));

    async function print(){

        return await Promise.all(fetch\_url).then(data\_arr => {

        for(let v of data\_arr){

        document.write(JSON.stringify(v)+"<br>");

    }});

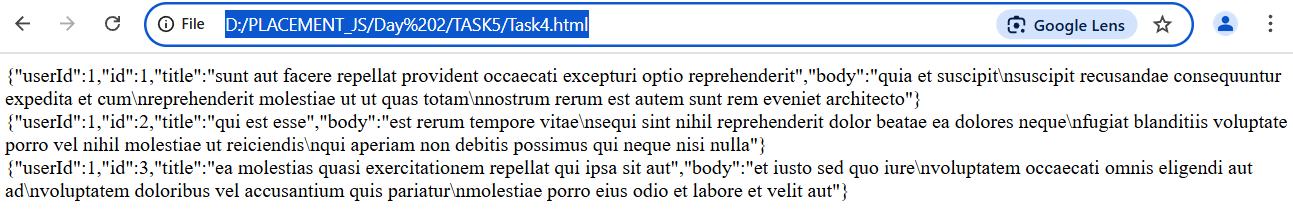
    }

    print().then();

    </script>

</body>

</html>

****

**TASK 5: Create an async function that waits for multiple asynchronous operations to complete before proceeding.**

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

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

    <title>Task5</title>

</head>

<body>

    <script>

        async function fun(){

            let pro1 = new Promise(function(resolve,reject){

                resolve();

            });

            let pro2 = new Promise(function(resolve,reject){

                document.write("Promise 2<br>");

            });

            let pro3 = new Promise(function(resolve,reject){

                reject();

            });

            pro1.then(() => document.write("resolve:Promise 1<br>"),() => document.write("reject:Promise 1<br>"));

            pro3.then(() => document.write("resolve:Promise 3<br>"),() => document.write("reject:Promise 3<br>"));

            try {

                await Promise.all([pro1,pro2,pro3]);

            }

            catch(error) {

                document.write("Error occured in Promise.all<br>");

            }

            return await new Promise(function(resolve,reject){

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

                resolve();

            });

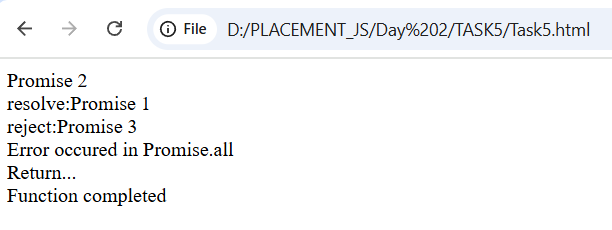
        }

        fun().then(() => document.write("Function completed"));

    </script>

</body>

</html>

****

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

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

export let num1 = 10;

export let num2 = 20;

export function sum(num1,num2){

    console.log(`sum : ${num1+num2}`);

}

export class Student{

constructor(name,id,age){

    this.name = name;

    this.id = id;

    this.age = age;

}

read() {

    console.log(`name : ${this.name} , id : ${this.id} , age : ${this.age}`);

}

}

**TASK 2: Import the module in another JavaScript file and use the exported entities.**

import { num1,num2,sum,Student } from "./TaskJ1.js";

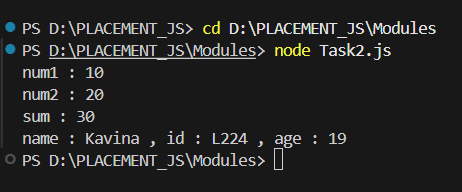
console.log(`num1 : ${num1}`);

console.log(`num2 : ${num2}`);

sum(num1,num2);

let s1 = new Student("Kavina","L224",19);

s1.read();

****

**TASK 3: Use named exports to export multiple functions from a module.**

export function sum(num1,num2){

    console.log(`sum : ${num1+num2}`);

}

export function mul(num1,num2){

    console.log(`mul : ${num1\*num2}`);

}

export function sub(num1,num2){

    console.log(`sub : ${num1-num2}`);

}

**TASK 4: Use named imports to import specific functions from a module.**

import {sum,mul,sub} from "./Task3.js";

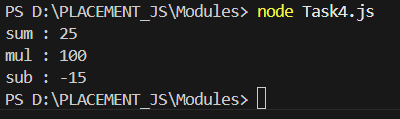
    let num1 =5;

    let num2 =20;

    sum(num1,num2);

    mul(num1,num2);

    sub(num1,num2);

****

**TASK 5: Use default export and import for a primary function of a module.**

export default function sum(num1,num2){

    console.log(`sum : ${num1+num2}`);

}

export function mul(num1,num2){

    console.log(`mul : ${num1\*num2}`);

}

export function sub(num1,num2){

    console.log(`sub : ${num1-num2}`);

}

import summ\_d,{mul,sub} from "./Task5.js"

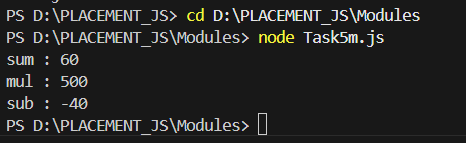
    let num1 = 10;

    let num2 = 50;

    summ\_d(num1,num2);

    mul(num1,num2);

    sub(num1,num2);

****

**7. BROWSER: DOM BASICS:**

**TASK 1: Select an HTML element by its ID and change its content using JavaScript.**

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

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

    <title>Task1</title>

</head>

<body>

    <p id="para">This is JavaScript</p>

    <button type="button" onclick="change()">click to change</button>

    <script>

        function change(){

            let p = document.getElementById("para");

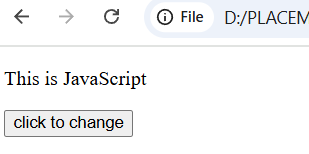
            p.textContent="This is Changed by Id";

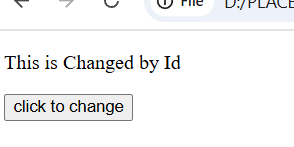
        }

    </script>

</body>

</html>

****

****

**TASK 2: Attach an event listener to a button, making it perform an action when clicked.**

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

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

    <title>Task2</title>

</head>

<body>

    <p id="para">This is JavaScript</p>

    <button type="button" id="clickkk">click</button>

    <script>

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

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

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

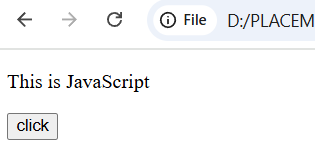
            paragraph.textContent="This is changed with addEventListener";

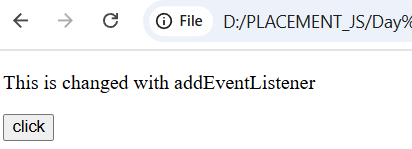
        });

    </script>

</body>

</html>

****

****

**TASK 3: Create a new HTML element and append it to the DOM.**

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

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

    <title>Task3</title>

</head>

<body>

    <p>this is created directly inside html</p>

    <button type="button" id="but" onclick="create()">click</button>

    <script>

        function create(){

            let h1 = document.createElement("h1");

            h1.id="heading1";

            h1.textContent = "this is created inside script";

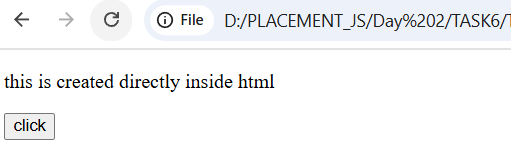
            document.body.append(h1);

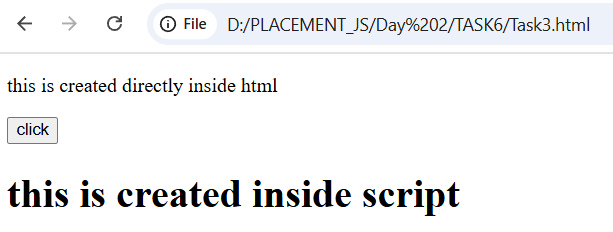
        }

    </script>

</body>

</html>

****

****

**TASK 4: Implement a function to toggle the visibility of an element.**

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

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

    <title>Task4</title>

</head>

<body>

    <p id="para">Hello JavaScript</p>

    <button id="but" type="button">Hide</button>

    <script>

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

        let count=0;

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

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

            if(count%2 == 0){

                paragraph.textContent="";

                button.textContent="Show";

            }

            else{

                paragraph.textContent="Hello JavaScript";

                button.textContent="Hide";

            }

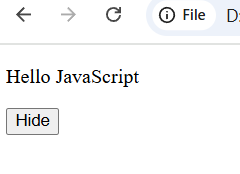
            count++;

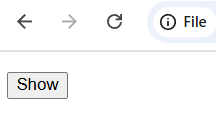
        });

    </script>

</body>

</html>

****

****

**TASK 5: Use the DOM API to retrieve and modify the attributes of an element.**

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

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

    <title>Task5</title>

</head>

<body>

    <img src="pexels-anjana-c-169994-674010.jpg" alt="Peacock Feature" id="image" height="250" width="250">

    <button type="button" id="but" onclick="change()">click</button>

    <script>

        function change(){

            let image = document.getElementById("image");

            image.height = "500"; //image.height = 500;

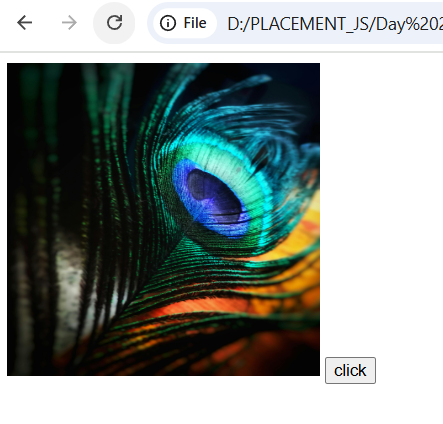
            image.width = "500";  //image.width = 500;

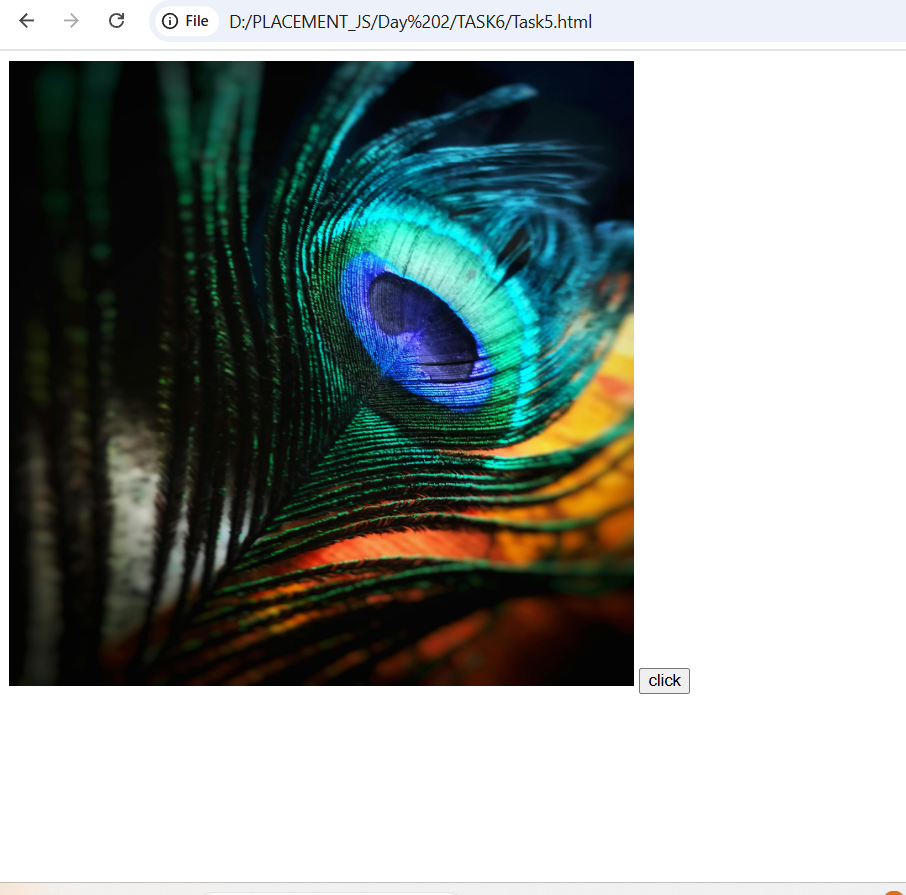
        }

    </script>

</body>

</html>

****

****