**MERN Stack Training**

**Advanced JS Tasks**

# **1. Recursion and stack:**

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

<script>

    function fact(n){

        if(n==0 || n==1)

        return 1;

    else

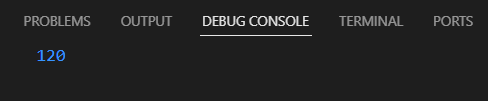
    return n\*fact(n-1);

    }

    console.log(fact(5));

</script>

**OUTPUT:**

****

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

<script>

    function fib(n){

        if(n<1)

        return 0;

    else if(n==1 || n==2)

    return 1;

    else

    return fib(n-1)+fib(n-2);

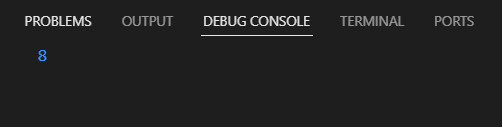
    }

    let v=prompt("Enter N:");

    console.log(fib(v));

</script>

**OUTPUT:**



**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.

<script>

    function climb(n){

        if(n==0)

        return 1;

    else if(n<0)

    return 0;

else

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

    }

    console.log(climb(4));

</script>

**OUTPUT:**

****

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

<script>

    function flattenArray(arr) {

    let result = [];

    arr.forEach(element => {

        if (Array.isArray(element)) {

            result = result.concat(flattenArray(element));

        } else {

            result.push(element);

        }

    });

    return result;

}

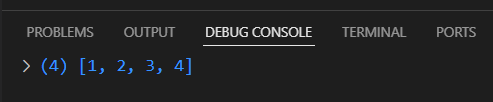
const nestedArray = [1, [2, [3, [4]]]];

let flattenedArray = flattenArray(nestedArray);

console.log(flattenedArray);

</script>

**OUTPUT:**



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

<script>

    function towerOfHanoi(n, from,  to,  aux)

    {

            if (n == 0)

            {

                return;

            }

            towerOfHanoi(n - 1, from, aux, to);

            console.log("Move disk " + n + " from rod " + from +

            " to rod " + to);

            towerOfHanoi(n - 1, aux, to, from);

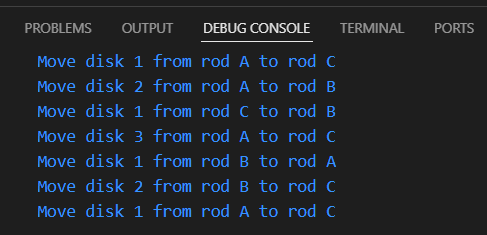
        }

        var N = 3;

        towerOfHanoi(N, 'A', 'C', 'B');

    </script>

**OUTPUT:**



# **2. JSON and variable length arguments/spread syntax:**

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

<script>

    function sum(...numbers){

        return numbers.reduce((total,num)=>num+total,0);

        let total=0;

        for(let num of numbers){

            total+=num;

        }

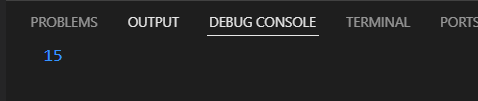
        return total;

    }

    console.log(sum(1,2,3,4,5));

</script>

**OUTPUT:**

****

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

<script>

    function sum(...numbers){

       let total=0;

       for(let num of numbers){

           total+=num;

       }

       return total;

   }

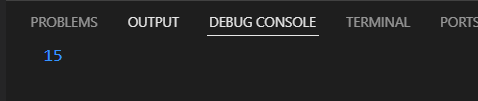
   let arr=[1,2,3,4,5];

   let res=sum(...arr);

   console.log(res);

</script>

**OUTPUT:**



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

<script>

    function clone(object){

        return JSON.parse(JSON.stringify(object));

    }

    const object={

        name:'Anu',

        age:20,

        address:{

            district:'Kallakurichi',

            village:'Siruvangur'

        },

        hobbies:['Drawing','Sleeping']

    };

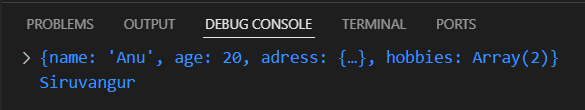
    const clonedObject=clone(object);

    console.log(clonedObject);

    console.log(clonedObject.adress.village);

</script>

**OUTPUT:**



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

<script>

    function merge(obj1,obj2){

        const mObj={...obj1,...obj2};

        return mObj;

    }

    const obj1={name:'Anu',age:20};

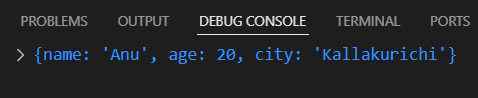
    const obj2={city:'Kallakurichi'};

    const mObj=merge(obj1,obj2);

    console.log(mObj);

</script>

**OUTPUT:**



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

<script>

    const object={name:'Anu',age:20};

    const jsonString=JSON.stringify(object);

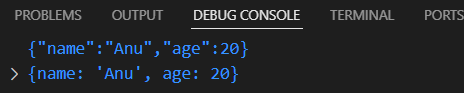
    console.log(jsonString);

    const parsedObj=JSON.parse(jsonString);

    console.log(parsedObj);

</script>

**OUTPUT:**



# **3. Closure:**

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

<script>

    function outerFun(x){

        return function innerFun(y){

            return x+y;

        }

    }

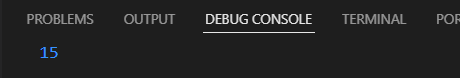
    const outer=outerFun(5);

    const res=outer(10);

    console.log(res);

</script>

**OUTPUT:**



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

<script>

    function createCounter(){

        let count=0;

        return {

            increment:function inc(){

                count++;

            },

            getCount:function get(){

                return count;

            }

        };

    }

    const c=createCounter();

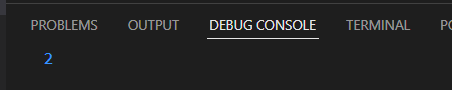
    c.increment();

    c.increment();

    console.log(c.getCount());

</script>

**OUTPUT:**



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

<script>

    function counterClosure(){

        let count=0;

        return {

            increment:function(){

                count++;

            },

            decrement:function(){

                count--;

            },

            getCount:function(){

                return count;

            }

        };

    }

    const counter1=counterClosure();

    const counter2=counterClosure();

    counter1.increment();

    counter1.increment();

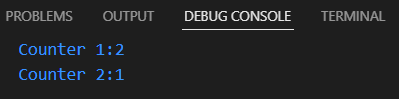
    counter2.increment();

    console.log("Counter 1:"+counter1.getCount());

    console.log("Counter 2:"+counter2.getCount());

</script>

**OUTPUT:**



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

<script>

function makeVarPrivate(){

        let count=0;

        return{

            increment:()=>{

                count++;

            },

            decrement:()=>{

                count--;

            },

            getCount:()=>{

                return count;

            }

        }

    }

    const fun=makeVarPrivate();

    fun.increment();

    console.log(fun.getCount());

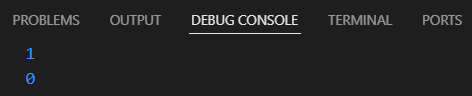
    fun.decrement();

    console.log(fun.getCount());

    //console.log(count);

</script>

**OUTPUT:**



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

<script>

    function factory(operation){

        switch(operation){

            case 'add':

                 return function(a,b){

                    return a+b;

                 };

            case 'subtract':

                return function(a,b){

                    return a-b;

                };

            case 'multiply':

                return function(a,b){

                    return a\*b;

                };

            case 'divide':

                return function(a,b){

                    if(b===0)

                    throw new Error("Can't divide by zero");

                    return a/b;

                };

            default:

                return function(){

                    throw new Error("Unsupported operation");

                }

        }

    }

    const add=factory('add');

    const sub=factory('subtract');

    const mul=factory('multiply');

    const div=factory('divide');

    console.log(add(5,6));

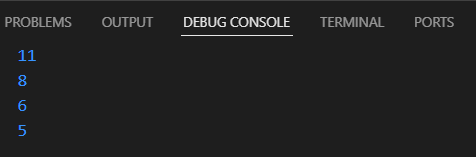
    console.log(sub(10,2));

    console.log(mul(2,3));

    console.log(div(10,2));

</script>

**OUTPUT:**

****

# **4. Promise, Promises chaining:**

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

<script>

    function greeting(second){

        return new Promise((resolve,reject)=>{

            if(second<=0)

            reject("Time must be greater than 0");

            else{

                setTimeout(()=>{

                    resolve(`Hi Waited for ${second} seconds`);

                },second\*1000);

            }

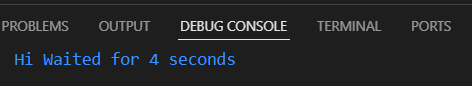
        });

    }

    greeting(4).then(greet=>console.log(greet)).catch(error=>console.log(error));

</script>

**OUTPUT:**



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

<script>

    function fetchData(){

        return fetch('https://jsonplaceholder.typicode.com/users/1').

        then((response)=>{

            if(!response.ok)

            throw new Error('NetWork issue');

           return response.json();

        });

    }

    function userData(user){

        return new Promise((resolve)=>{

            const userD=`Name:${user.name} Email:${user.email} Company:${user.company}`;

            resolve(userD);

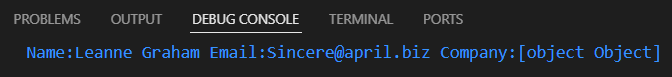
        });

    }

    fetchData().then(user=>{return userData(user)}).then(data=>console.log(data)).catch(error=>{console.log(error)});

</script>

**OUTPUT:**

****

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

<script>

    function check(){

        return new Promise((resolve,reject)=>{

            const random=Math.random();

            if(random<0.5)

            resolve(`Resolved The random number is ${random}`);

            else

            reject(`Rejected The Random number is ${random}`);

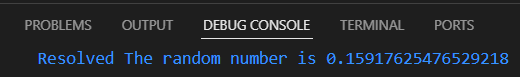
        });

    }

    check().then((response)=>console.log(response)).catch((error)=>console.log(error));

</script>

**OUTPUT:**

****

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

<script>

    function createPromisses(userId){

        return fetch(`https:jsonplaceholder.typicode.com/users/${userId}`).

        then((response)=>{

            if(!response.ok){

                throw new Error(`Network Error for Id ${userId}`);

            }else{

                return response.json();

            }

        });

    }

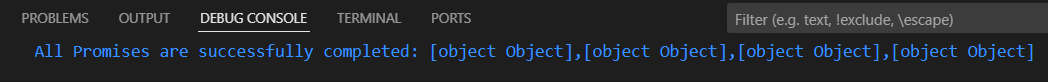
    const uId=[1,2,3,4];

    const pro=uId.map(id=>createPromisses(id));

    Promise.all(pro).then((response)=>console.log("All Promises are successfully completed: "+response)).catch((error)=>console.log("Error Occured :"+error))

</script>

**OUTPUT:**

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

<script>

    function action1(){

        return new Promise((resolve)=>

        setTimeout(()=>{

            console.log("Action 1 Function ");

            resolve("Action 1");

        },2000)

    );

    }

    function action2(content){

        return new Promise((resolve)=>

        setTimeout(()=>{

            console.log("Action 2 Function");

            console.log("Previous Chaning Function :"+content);

            resolve("Action 2");

        },2000)

        );

    }

    function action3(content){

        return new Promise((resolve)=>

        setTimeout(()=>{

            console.log("Action 3 Function");

            console.log("Previous Chaning Function :"+content);

            resolve("Action 3 Successfully completed");

        },2000)

    );

    }

    action1().then((response1)=>action2(response1)).then((response2)=>action3(response2)).catch((error)=>console.log(error));

</script>

**OUTPUT:**

****

# **5. Async/await:**

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

<script>

    function usingPromise(url){

        return fetch(url).then((response)=>{

            if(!response.ok)

            throw new Error("NetWork error");

            else{

                return response.json();

            }

        }).then((response)=>console.log(response)).catch((error)=>console.log(error));

    }

    //usingPromise('https://jsonplaceholder.typicode.com/users/1');

    async function usingAsyncAwait(url){

        try{

            const response=await fetch(url);

            if(!response.ok)

            throw new Error('Network Error');

           const data=await response.json();

           console.log(data)

        }catch(error){

            console.error("Error Occured:"+error);

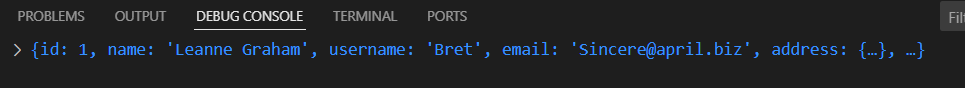
        }

    }

    usingAsyncAwait('https://jsonplaceholder.typicode.com/users/1')

</script>

**OUTPUT:**



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

<script>

    async function urlProccessing(url){

        try{

            const response=await fetch(url);

            if(!response.ok)

            throw new Error('Network Error');

            const data=await response.json();

            console.log(`Id:${data.id} Name:${data.username}`);

        }catch(error){

            console.error(error);

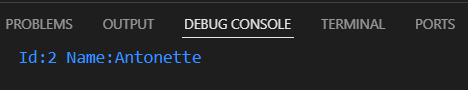
        }

    }

    urlProccessing('https://jsonplaceholder.typicode.com/users/2');

</script>

**OUTPUT:**



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

<script>

    async function errorHandling(url){

        try{

            const response=await fetch(url);

            if(!response.ok)

            throw new Error(`Error occured ${response.status} `);

            const data=await response.json();

            console.log(`Fetched Data ${data}`);

            return data;

        }catch(error){

            console.log(error);

            return null;

        }

    }

    errorHandling('https://jsonplaceholder.typicode.com/users/1').

    then((result)=>{

        if(result)

        console.log("Proccessed data "+result);

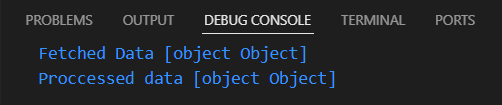
        else

        console.log("Np processed data");

    });

</script>

**OUTPUT:**



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

<script>

    async function fetchMultipleApi(urls){

        try{

            const promises=urls.map((url)=>fetch(url).then((response)=>response.json()));

            const result= await Promise.all(promises);

            console.log("Fetched result "+result);

            return result;

        }catch(error){

            console.error("Error Occured");

        }

    }

    const urls=[

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

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

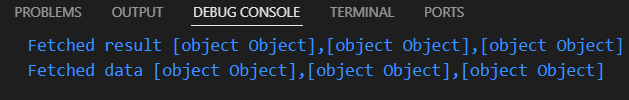
        'https://jsonplaceholder.typicode.com/users/3'

    ];

    fetchMultipleApi(urls).then((result)=>console.log("Fetched data "+result)).catch((error)=>console.log(error));

</script>

**OUTPUT:**



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

<script>

    async function mulipleApiAsynchronous(){

        try{

            const api1=fetch('https://jsonplaceholder.typicode.com/users/1').then((response)=>response.json());

            const api2=fetch('https://jsonplaceholder.typicode.com/users/2').then((response)=>response.json());

            const api3=fetch('https://jsonplaceholder.typicode.com/users/3').then((response)=>response.json());

            const results=await Promise.all([api1,api2,api3]);

            results.forEach((response)=>console.log(response));

            return results;

        }catch(error){

            console.error(error);

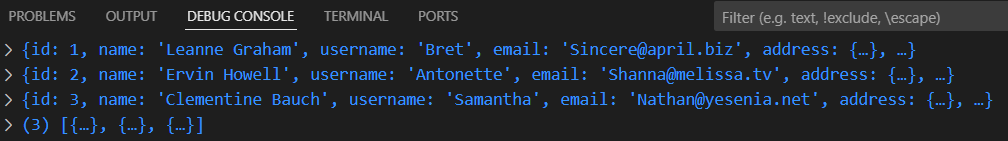
        }

    }

    mulipleApiAsynchronous().then((result)=>console.log(result)).catch((error)=>console.log(error));

</script>

**OUTPUT:**



# **6. Modules introduction, Export and Import:**

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

export function greet(name){

    console.log(`Hello ${name} welcome`);

}

export class Person{

    constructor(name,age) {

        this.name=name;

        this.age=age;

    }

   introduce(){

    return `Hi My name is ${this.name} and my age is ${this.age}`;

   }

}

export const name='Anu';

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

<!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 type="module">

        import {greet,Person,name} from './t26.js';

        greet('Anu');

        const person=new Person('Suba',16);

        console.log(person.introduce());

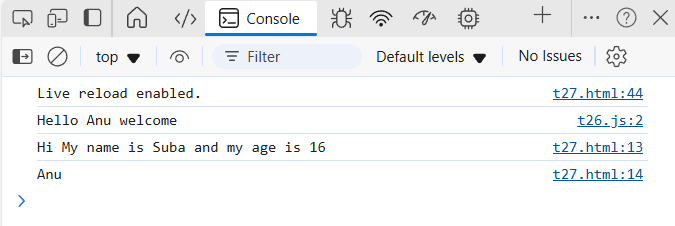
        console.log(name);

    </script>

</body>

</html>

**OUTPUT:**

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

<!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 type="module">

        import {add,subtract,mul,div} from './t28.js';

        console.log("Addition of 5+5 ="+add(5,5));

        console.log("Subtracction of 10-5 ="+subtract(10,5));

        console.log("Multiplication of 7\*5 ="+mul(7,5));

        console.log("Division of 25/5 ="+div(25,5));

    </script>

</body>

</html>

t28.js:

function add(a,b){

    return a+b;

}

function subtract(a,b){

    return a-b;

}

function mul(a,b){

    return a\*b;

}

function div(a,b){

    if(b===0)

        return "Cannot divide by 0";

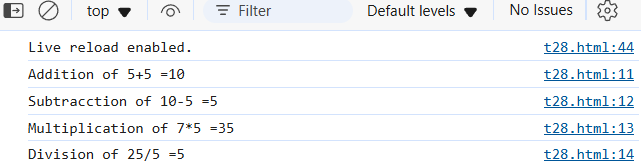
    else

    return a/b;

}

export {add,subtract,mul,div};

**OUTPUT:**

****

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

<!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 type="module">

        import \* as Task from './t28.js';

        console.log("Addition of 5+5 ="+Task.add(5,5));

        console.log("Subtracction of 10-5 ="+Task.subtract(10,5));

        console.log("Multiplication of 7\*5 ="+Task.mul(7,5));

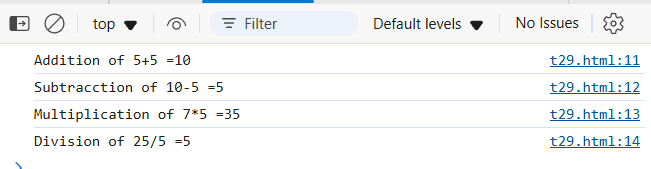
        console.log("Division of 25/5 ="+Task.div(25,5));

    </script>

</body>

</html>

**OUTPUT:**

****

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

<!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 type="module">

        import greet from './t30.js';

        console.log(greet('Anu'));

    </script>

</body>

</html>

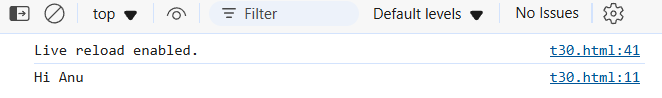
t30.js

export default function greet(name){

    return `Hi ${name}`;

}

**OUTPUT:**



# **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>Document</title>

</head>

<body>

    <h1 id="header">Hello World</h1>

    <script>

        const ref=document.getElementById('header');

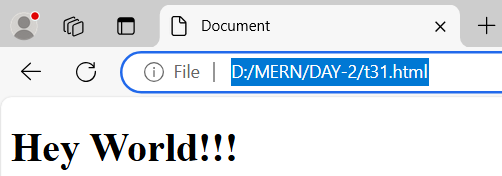
        ref.textContent="Hey World";

    </script>

</body>

</html>

**OUTPUT:**



**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>Document</title>

</head>

<body>

    <h1 id="head">Hello Anu</h1>

    <button id="Button">Click Me</button>

    <script>

        const b=document.getElementById('Button');

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

            const header=document.getElementById('head');

            header.textContent="Hey how are you?";

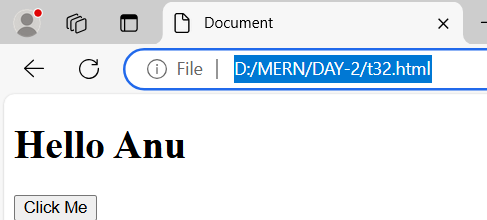
        })

    </script>

</body>

</html>

**OUTPUT:**



# 

**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>Document</title>

</head>

<body>

    <div id="container">

        <h1>Content Default Html</h1>

    </div>

    <script>

        const newElement=document.createElement("h2");

        newElement.textContent="I am inserted by using js";

        newElement.style.color='blue';

        const existing=document.getElementById('container');

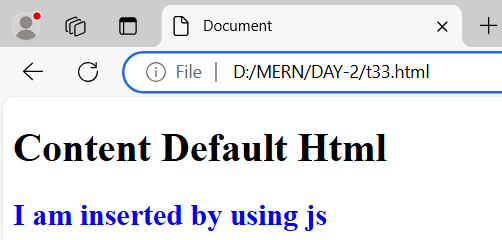
        existing.appendChild(newElement);

    </script>

</body>

</html>

**OUTPUT:**



**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>Document</title>

</head>

<body>

    <button id="togButton">Toggle Content</button>

    <div id="togElement">I am Visible if you want me to unvisible click the above button (:</div>

    <script>

        function toggle(togId){

            const ref=document.getElementById(togId);

            if(ref.style.display==='none'){

                ref.style.display='block';

            }else{

                ref.style.display='none';

            }

        }

        const bu=document.getElementById('togButton');

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

            toggle('togElement');

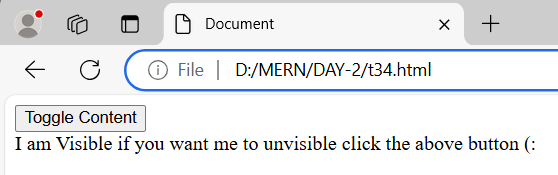
        });

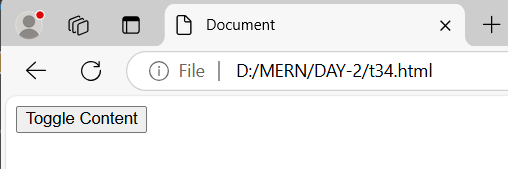
    </script>

</body>

</html>

**OUTPUT:**





**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>Document</title>

</head>

<body>

    <a id="myLink" href="www.google.com" target="\_blank">Google</a>

    <button id="buttonId">Modify Attributes</button>

    <script>

        function modify(modId){

            const ref=document.getElementById(modId);

            const existingLink=ref.getAttribute('href');

            console.log("Existing href link "+existingLink);

            ref.setAttribute('href','www.yahoo.com');

            console.log("Modified href link "+ref.getAttribute('href'));

            console.log("Existing target style "+ref.getAttribute('target'));

            ref.setAttribute('target','\_self');

            console.log("Existing content "+ref.getAttribute('title'));

            ref.setAttribute('title','Yahoo');

            console.log("Modified Title "+ref.getAttribute('title'));

            ref.textContent="Yahoo";

        }

        const bu=document.getElementById('buttonId');

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

            modify('myLink');

        })

    </script>

</body>

</html>

**OUTPUT:**

