1) **Recursion and stack:**

Task 1:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <title>Document</title>
</head>
<body>
    <script>
        function factorial(n){

            if(n<=1){
                return 1;
            }
            else{

                return n*factorial(n-1);
            }
        }
        console.log(factorial(6));
    </script>
</body>
</html>
```

Output:

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    PORTS    SPELL CHECKER

720
```

Task 2:

```
<!DOCTYPE html>
<html lang="en">
<head>
<title>fibonicci</title>
</head>
<body>
<script>
function fibonicci(n){
 if(n==0){
```

```
    return 0;
   }
  if(n==1){
   return 1;
  }
  else
{
   return fibonicci(n-1)+fibonicci(n-2);
}

}
console.log(fibonicci(5));
</script>
</body>
</html>
```

Output:

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    PORTS    SPELL CHECKER  5
  5
```

Task 3:

```
<!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 stares(n){

         if(n==0){
            return 1;

         }
```

```
        if(n<0){
            return 0;
        }
        else{
            return stares(n-1)+stares(n-2)+stares(n-3);
        }

    }
    console.log(stares(4));

  </script>
</body>
</html>
```

Output:

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    PORTS    SPELL CHECKER
  7
```

Task 4:

```
<!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 flatter(n){

        let final=[];
        function fun(ele){
            if(Array.isArray(ele)){
                for(let i=0;i<ele.length;i++){

                fun(ele[i]);
                }
            }
```

```
            else{
                final.push(ele);
            }
        }
        fun(n);
        return final;

    }
    let arr=[2,3,[4,6,5],7,8,[1,2]];
    let result=flatter(arr);
    console.log(result);

  </script>
</body>
</html>
```

Output:

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    PORTS    SPELL CHECKER
> (9) [2, 3, 4, 6, 5, 7, 8, 1, 2]
```

Task 5:

```
<!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 TowerofHanoi(n,source,destination,auxilary){
        if(n===1){
            console.log(`Move disk 1 from ${source} To ${destination}`);
```

```
            return 1;
        }
        TowerofHanoi(n-1,source,auxilary,destination);
        console.log(`Move disk ${n} from ${source} To ${destination}`);
        TowerofHanoi(n-1,auxilary,destination,source);
    }

    TowerofHanoi(3,'A','C','B');
  </script>
</body>
</html>
```

Output:

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    PORTS    TERMINAL
  Move disk 1 from A To C
  Move disk 2from A To B
  Move disk 1 from C To B
  Move disk 3from A To C
  Move disk 1 from B To A
  Move disk 2from B To C
  Move disk 1 from A To C
```

2) **JSON and variable length arguments/spread syntax:**

Task 1:

```
<!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 arbitary(a,b){

            return a+b;
        }
        console.log(arbitary(0.1456,-9999));
        console.log(arbitary(-5.348, 2.1));
        console.log(arbitary(9999999999.999991,-100000.0000001));

    </script>
</body>
</html>
```

Output:

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    PORTS    TERMINAL
  -9998.8544
  -3.2479999999999998
  9999899999.99999
```

Task 2:

```
<!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 number(...n){
        var sum=0;
        for(const num of n){
```

```
        sum+=num

    }
    return sum;
    }
    var arr=[1,2,3,4,5];
    console.log(number(...arr));
  </script>
</body>
</html>
```

Output:

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    PORTS    TERMINAL
 15
```

Task 3:

```
<!DOCTYPE html>
<html lang="en">
<head>

  <title>Deepclone</title>
</head>
<body>
  <script>

    function deepclone(obj){

     return JSON.parse(JSON.stringify(obj));
    }
```

```
    const object={

     name:'john',
     age:21,
     skills:['python','js','java'],
     address :{

         city:'chennai',
         pincode:600001
     }
    };
    var clonedObject=(deepclone(object));

    console.log(clonedObject);
   </script>
</body>
</html>
```

Output:

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    PORTS    TERMINAL
> {name: 'john', age: 21, skills: Array(3), address: {…}}
```

Task 4:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <title>Adding two object</title>
</head>
<body>
    <script>
        function merging(obj1,obj2){

            let newobj={...obj1,...obj2};
            let result=JSON.stringify(newobj);
            return result;
        }
```

```
        let obj1 = {

            name:'john',
            age: 34,
            company:'ibm',
            skills:'python'
        };
        let obj2={

            name:'kennedy',
            age:54,
            college:'kce',
            dept:'cd'
        };
        document.writeln(merging(obj1,obj2));


    </script>
</body>
</html>
```

Output:

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS    Filter (e.g. text, !exclude, \escape)
  {"name":"kennedy","age":54,"company":"ibm","skills":"python","college":"kce","dept":"cd"}
```

Task 5:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <title>Mern Stack</title>
</head>
<body>
    <script>

        let obj={

            name:'kennedy',
            age:54,
            college:'kce',
            dept:'cd'
        };
        let result=JSON.stringify(obj);
        console.log(result);
```

```
        let last=JSON.parse(result);
        console.log(last);


    </script>
</body>
</html>
```

Output:

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS    Filter (e.g. text, !exclude, \esc

  {"name":"kennedy","age":54,"college":"kce","dept":"cd"}
> {name: 'kennedy', age: 54, college: 'kce', dept: 'cd'}
```

3) **Closure:**

Task 1:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <title>Mern Stack</title>
</head>
<body>
    <script>

        function fun1(){

            var n=10;
             function number(){

              console.log(n);
            }
            number();
        }
        fun1();

    </script>
</body>
</html>
```

Output:

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS    Filter (e.g.

  10
```

Task 2:

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <title>Mern Stack</title>
</head>
<body>
    <script>

        function fun1(){

            var count=0;
            return function(){

                count++;
                return count;
            }
        }
        const counter = fun1();
        console.log(counter());
        console.log(counter());
        console.log(counter());
    </script>
</body>
</html>
```

Output:

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS    Filt

  1
  2
  3
```

Task 3:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <title>Mern Stack</title>
</head>
<body>
    <script>

        function counter(){
            return function(){
                var count=0;
                return function() {
                    count++;
                    return count;

                };
            };

        }
        const counter1 = counter()();
const counter2 = counter()();

console.log(counter1());
console.log(counter1());
console.log(counter2());
console.log(counter2());
console.log(counter1());
    </script>
</body>
</html>
```

Output:

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS

1
2
1
2
3
```

Task 4:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <title>Mern Stack</title>
</head>
<body>
    <script>

        function counter(){

            let count=0;

            return {
                increment(){
                    count++;
                    return count;
                },
                decrement(){

                    count--;
                    return count;
                },
                print(){

                    return count;
                }


            }
        };
        const counter1 = counter();


console.log(counter1.increment());
console.log(counter1.increment());
console.log(counter1.decrement());
console.log(counter1.print());




    </script>
</body>
</html>
```

Output:

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS

  1
  2
  1
  1
```

Task 5:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <title>Mern Stack</title>
</head>
<body>
    <script>

        function Createmultiply(factor){

          return function(num){
            return num*factor;
          }

        };
        const counter1 = Createmultiply(5);
        const counter2 = Createmultiply(6);
        const counter3 = Createmultiply(3);

console.log(counter1(8));
console.log(counter2(4));
console.log(counter3(9));

    </script>
</body>
</html>
```

Output:

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS

40
24
27
```

4) **Promise, Promises chaining:**

Task 1:

```
<!DOCTYPE html>
<html lang="en">
<head>

    <title>mern stack</title>
</head>
<body>
    <script>

       function promis(){

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

        setTimeout(function(){
           resolve("greeting");
        },3000);
    });
       }
      promis().then((result)=>{
        console.log(result);
      }).catch((error)=>{console.log(errror)})
```

```
    </script>
</body>
</html>
```

Output:

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    PORTS    TERMINAL
greeting
```

Task 2:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>MERN Stack</title>
</head>
<body>
    <script >
            function fetchData() {
        return fetch("https://jsonplaceholder.typicode.com/posts")
            .then((response) => {
                if (!response.ok) {
                    throw new Error("Failed to fetch data.");
                }
                return response.json(); // Parse JSON data
            });
    }
    function processData(data) {
        return new Promise((resolve) => {
            const processedData = data.slice(0, 5);
            resolve(processedData);
        });
    }
    fetchData()
        .then((data) => {
            console.log("Fetched Data (First 5):", data.slice(0, 5));
            return processData(data);
        })
        .then((processedData) => {
            console.log("Processed Data:", processedData);
```

```
        })


    </script>
</body>
</html>
```

Output:

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS

> Fetched Data (First 5): (5) [{…}, {…}, {…}, {…}, {…}]
> Processed Data: (5) [{…}, {…}, {…}, {…}, {…}]
```

Task 3:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>MERN Stack</title>
</head>
<body>
    <script >
           function randomPromise() {
    return new Promise((resolve, reject) => {
        const randomNumber = Math.random();
        console.log("The random number is "+randomNumber);
        if (randomNumber > 0.5) {
            resolve("Success: The random number is greater than 0.5");
        } else {
            reject("Error: The random number is less than or equal to 0.5");
        }
    });
}
    randomPromise()
    .then((message) => {
        console.log(message);
```

```
    })
    .catch((error) => {
        console.error(error);
    })
    .finally(() => {
        console.log("The promise has settled.");
    });


    </script>
</body>
</html>
```

Output:

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS

The random number is 0.2188037918790835
Error: The random number is less than or equal to 0.5
The promise has settled.
```

Task 4:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>MERN Stack</title>
</head>
<body>
    <script >
        function fetchData(url) {
    return fetch(url)
        .then((response) => {
            if (!response.ok) {
                throw new Error(`Failed to fetch from ${url}`);
            }
            return response.json();
        });
```

```
}
const url1 = "https://jsonplaceholder.typicode.com/posts";
const url2 = "https://jsonplaceholder.typicode.com/comments";
const url3 = "https://jsonplaceholder.typicode.com/albums";
Promise.all([fetchData(url1), fetchData(url2), fetchData(url3)])
    .then((results) => {
        const posts = results;
        console.log("Posts:", posts);
    })
    .catch((error) => {
        console.error("Error:", error);
    })
    .finally(() => {
        console.log("All API calls have completed.");
    });


    </script>
</body>
</html>
```

Output:

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS

> Posts: (3) [Array(100), Array(500), Array(100)]
  All API calls have completed.
```

Task 5:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>MERN Stack</title>
</head>
<body>
    <script >
       function asyncTask(taskName, delay) {
    return new Promise((resolve, reject) => {
        setTimeout(() => {
            console.log(`${taskName} completed`);
```

```
            resolve(taskName)
        }, delay);
    });
}
asyncTask("Task 1", 1000)
    .then((result) => {
        console.log(`${result} has finished. Proceeding to next task.`);
        return asyncTask("Task 2", 2000);
    })
    .then((result) => {
        console.log(`${result} has finished. Proceeding to next task.`);
        return asyncTask("Task 3", 1500);
    })
    .then((result) => {
        console.log(`${result} has finished. All tasks completed.`);
    })
    .catch((error) => {
        console.error("An error occurred:", error);
    })
    .finally(() => {
        console.log("All tasks are complete.");
    });

    </script>
</body>
</html>
```

Output:

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS

Task 1 completed
Task 1 has finished. Proceeding to next task.
Task 2 completed
Task 2 has finished. Proceeding to next task.
Task 3 completed
Task 3 has finished. All tasks completed.
All tasks are complete.
```

5) **Async/await:**

Task 1:

```html
<!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>
      async function fetchData() {
    const success = true;
    return new Promise((resolve, reject) => {
        setTimeout(() => {
            if (success) {
                resolve("Data fetched successfully!");
            } else {
                reject("Failed to fetch data.");
            }
        }, 1000);
    });
}

async function execute() {
    try {
        const data = await fetchData();
        console.log(data);
    } catch (error) {
        console.error(error);
    }
}

execute();

        </script>
</body>
</html>
```

Output:

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS

  Data fetched successfully!
```

Task 2:

```html
<!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>
    data = async()=>{
 try{
      const url = await fetch('https://api.github.com/users/iliakan');
      if(!url.ok)
    throw new Error("Can't able to fetch the data");
    console.log("Data Fetched...");
    const pro = await url.json()
    console.log("Fetched data: ",pro);
    const pdata = await pro.name.toUpperCase()
        return pdata;
 }
    catch(error){
    console.error(error)
 }
 }
 data().then((res)=>{
 console.log("Fetched Response",res);
 })

        </script>
</body>
</html>
```

Output:

```
PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL   PORTS                                   Filter (e.g. text, !exclude, \escape)
 Data Fetched...                                                                                                    index.html:16
> Fetched data:  {login: 'iliakan', id: 349336, node_id: 'MDQ6VXNlcjM0OTMzNg==', avatar_url: 'https://avatars.githubusercontent.com/u/349336?v=4', gravatar_id: '', …}   index.html:18
 Fetched Response ILYA KANTOR                                                                                       index.html:27
```

Task 3:

```html
<!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>

    async function res() {

        try{

            var api=await fetch('https://no-url');

        }catch(error){
            console.log(error);
        }

    }
    res()
        </script>
</body>
</html>
```

Output:

```
   Live reload enabled.                                                    index.html:54
⊗ Failed to load resource: net::ERR_NAME_NOT_RESOLVED                      no-url/:1
   TypeError: Failed to fetch                                              index.html:19
       at res (index.html:16:27)
       at index.html:23:5
>
```

Task 4:

```
<!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>

    async function name(){
 let urls = [
 'https://api.github.com/users/iliakan',
 'https://api.github.com/users/remy',
 'https://api.github.com/users/jeresig'
 ];
 let requests = await Promise.all(urls.map(url => fetch(url))) ;
  let obj = await
Promise.all(requests.map((res)=>res.json())) ;
 return obj;
 }
 name().then(responses => responses.forEach(
 response => console.log(`${response.name}: ${response.login}`)  )).catch(error => {
 console.error('Error:', error);
 });
        </script>
</body>
</html>
```

Output:

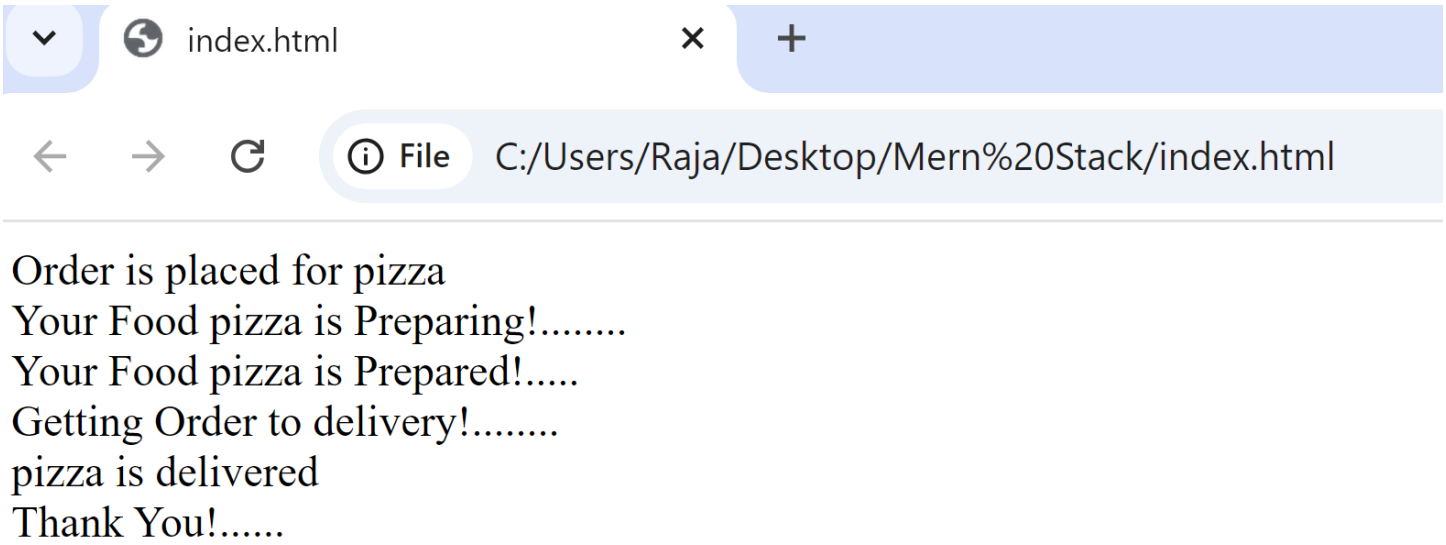
```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS

Ilya Kantor: iliakan
Remy Sharp: remy
John Resig: jeresig
```

Task 5:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>MERN Stack</title>
</head>
<body>
    <script >
            orderfood = (order) => {
        document.write(`Getting Order!........`)
        document.write("<br>")
        return new Promise((resolve)=>{
        setTimeout(() => {
        document.write(`Order is placed for ${order}`);
          document.write("<br>")
        resolve(order)
        }, 1000);
        })
        }
        preparefood = (order) => {
        document.write(`Your Food ${order} is Preparing!........`) ;
         document.write("<br>")
        return new Promise((resolve)=>{
        setTimeout(() => {
        document.write(`Your Food ${order} is Prepared!.....`) ;
         document.write("<br>")
        resolve(order)
        }, 1000);
        })
        }
        deliverfood = (order) => {
        document.write(`Getting Order to delivery!........`);
          document.write("<br>")
        return new Promise((resolve)=>{
        setTimeout(() => {
        document.write(`${order} is delivered`);
         document.write("<br>")
        resolve(order)
        }, 1500);
        })
        }
        food = async(value) => {
        const order = await orderfood(value)
        const Prepare = await preparefood(order)
        const deliver = await deliverfood(Prepare)
        if(deliver == order){
        document.write("Thank You!......");
```

```
        }else{
        document.write("Please Wait!...") ;
        }
        }
        value = prompt("Enter the Dish you desire: ");
          food(value)
    </script>
</body>
</html>
```

Output:



6) **Modules introduction, Export and Import:**

Task 1:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>MERN Stack</title>
</head>
<body>
    <script type="module" src="./main.js"></script>
</body>
</html>
```

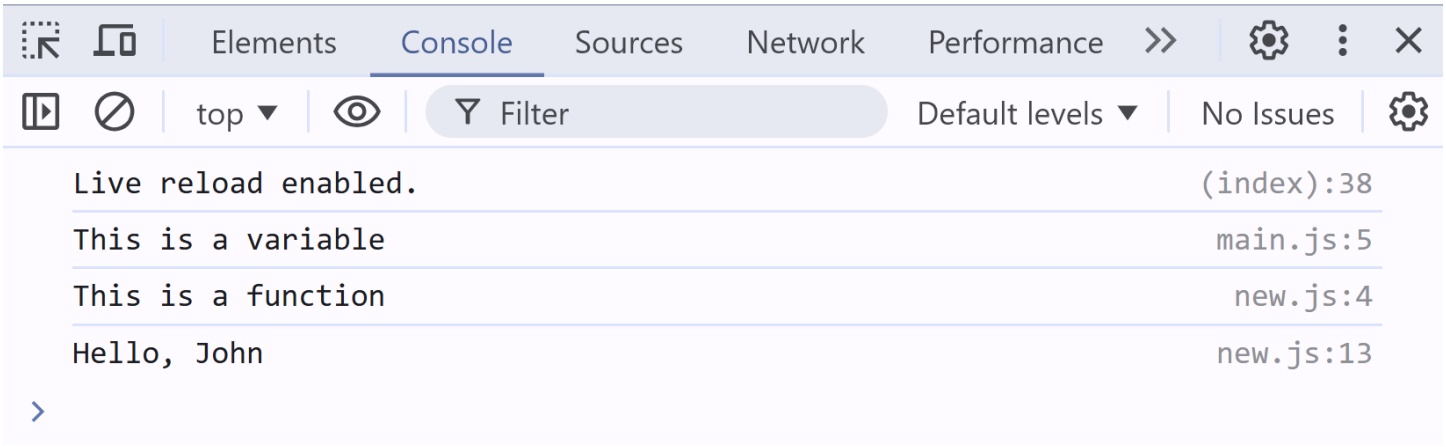
Main.js:

```
import { variable, vari, names } from './new.js';


console.log(variable);
vari();

const obj = new names("John");
obj.greet();
```

new.js:

```
export const variable = "This is a variable";

export function vari() {
    console.log("This is a function");
}

export class names {
    constructor(name) {
        this.name = name;
    }

    greet() {
        console.log(`Hello, ${this.name}`);
    }
}
```

Output:



Task 2:

```html
<!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 } from "./new.js";

console.log("Addition:", add(5, 3));
console.log("Subtraction:", subtract(5, 3));
        </script>
</body>
</html>
```

New.js:

```js
export function add(a, b) {
    return a + b;
}

export function subtract(a, b) {
    return a - b;
}
```

Output:

```
Live reload enabled.                                    index.html:45
Addition: 8                                             index.html:14
Subtraction: 2                                          index.html:15
>
```

Task 3:

```html
<!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">
        export function sayHi(user) {
        console.log(`Hello, ${user}!`);
        }
        export function sayBye(user) {
        console.log(`Bye, ${user}!`);
        }
        sayHi("Irfan")
        </script>
</body>
</html>
```

Output:

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS

Hello, Irfan!
```

Task 4:

```html
<!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 { sayHi,sayBye } from './new.js';
 sayHi('John');
 sayBye('vickey')

        </script>
</body>
</html>
```

New.js

```
 export function sayHi(user) {
    console.log(`Hello, ${user}!`);
    }
    export function sayBye(user) {
    console.log(`Bye, ${user}!`);
    }
```

Output:

Task 5:

```
<!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 sayHi from './new.js';

new sayHi('hendry');

        </script>
</body>
</html>
```

New.js:

```
export default function sayHi(user) {
   console.log(`Hello, ${user}!`);
   }
   export default function sayBye(user) {
   console.log(`Bye, ${user}!`);
   }
```

Output:

```
   Live reload enabled.                                    index.html:45
   Hello, hendry!                                                new.js:4
>
```

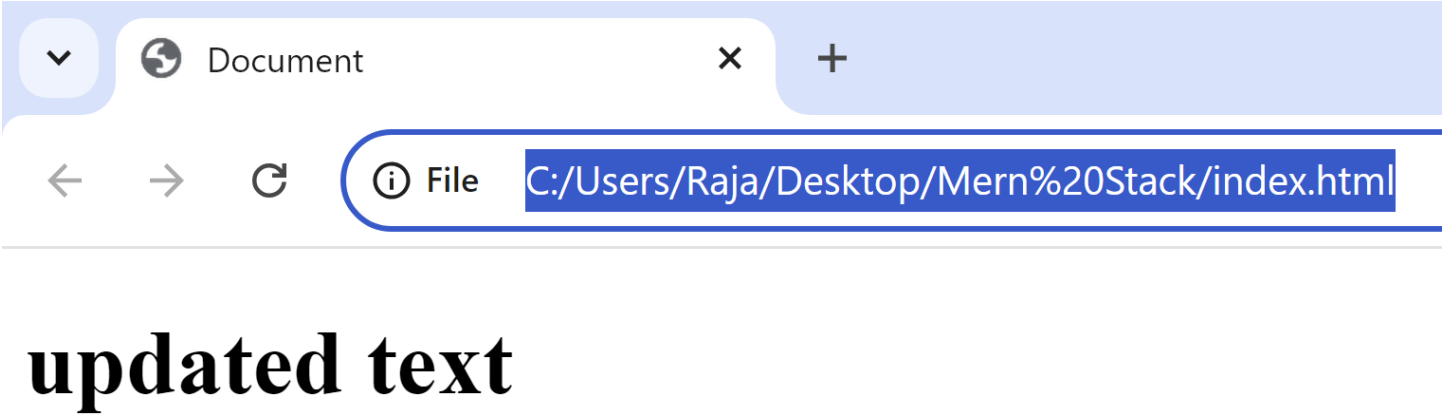
7) **Browser: DOM Basics:**

Task 1:

```
<!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">First Text</h1>
    <script>
        var head=document.getElementById("head");
```

```
        setTimeout(()=>{head.textContent="updated text"},1000);
    </script>
</body>
</html>
```

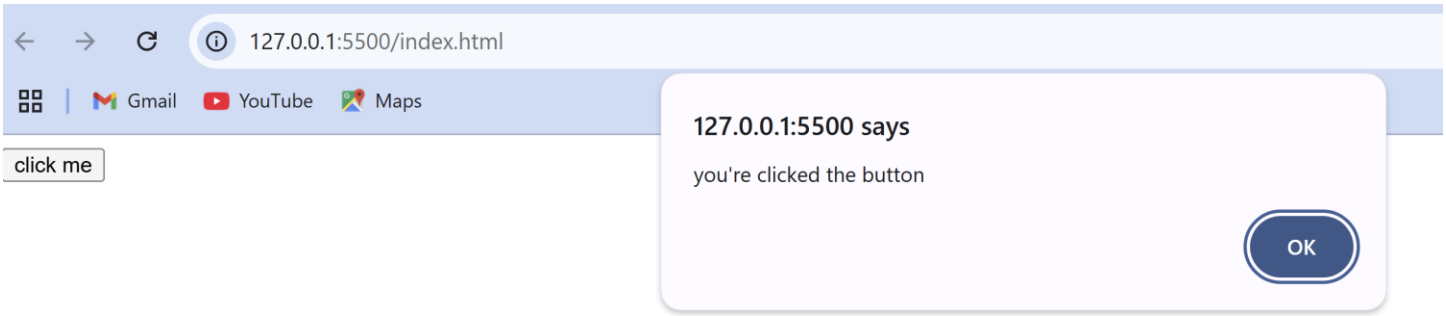
Output:



Task 2:

```
<!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>
    <input type="button" value="click me" id="unique">
    <script>
        var head=document.getElementById("unique");
        head.addEventListener("click",()=>{alert("you're clicked the button")});
    </script>
</body>
</html>
```

Output:



Task 3:

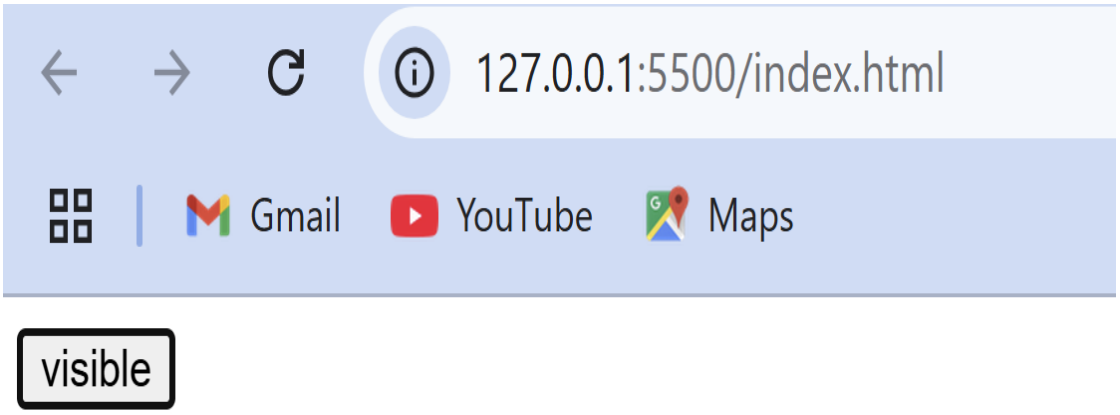
```
<!DOCTYPE html>

<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>
         var h=document.createElement("h1");
        h.textContent="created tag";
        document.body.appendChild(h);
        </script>
</body>
</html>
```

Output:

Task 4:

```html
<!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="con" style="height:50px; width:50px;background-color: blue;"></div>
    <input type="button" value="click me" id="unique" onclick="toggle()" value="hide">

    <script>
        var element=document.getElementById("con");
        var btn=document.getElementById("unique");
        function toggle(){

            if(element.style.display=="block"){

                element.style.display="none";
                btn.value="visible";
            }
            else{
                element.style.display="block";
                btn.value="hide";
            }
        }
        </script>
</body>
</html>
```

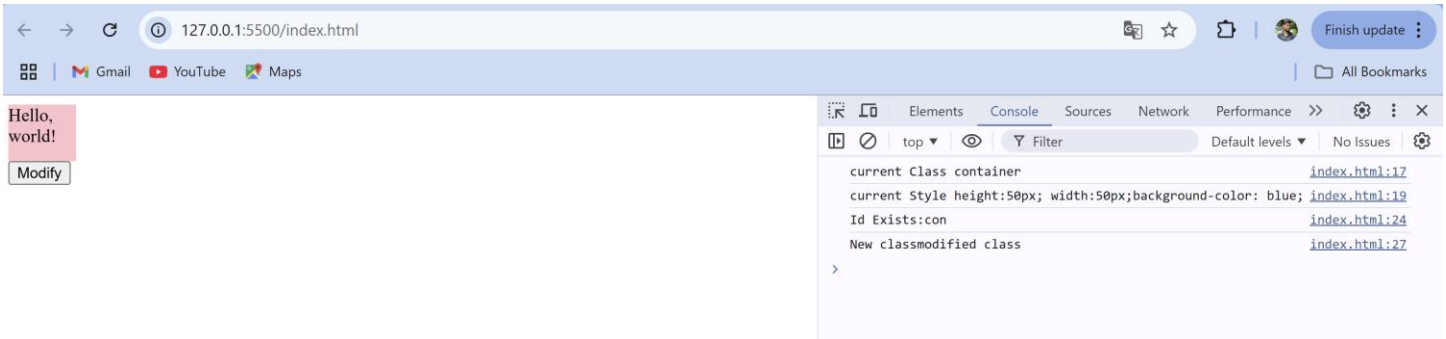
Output:

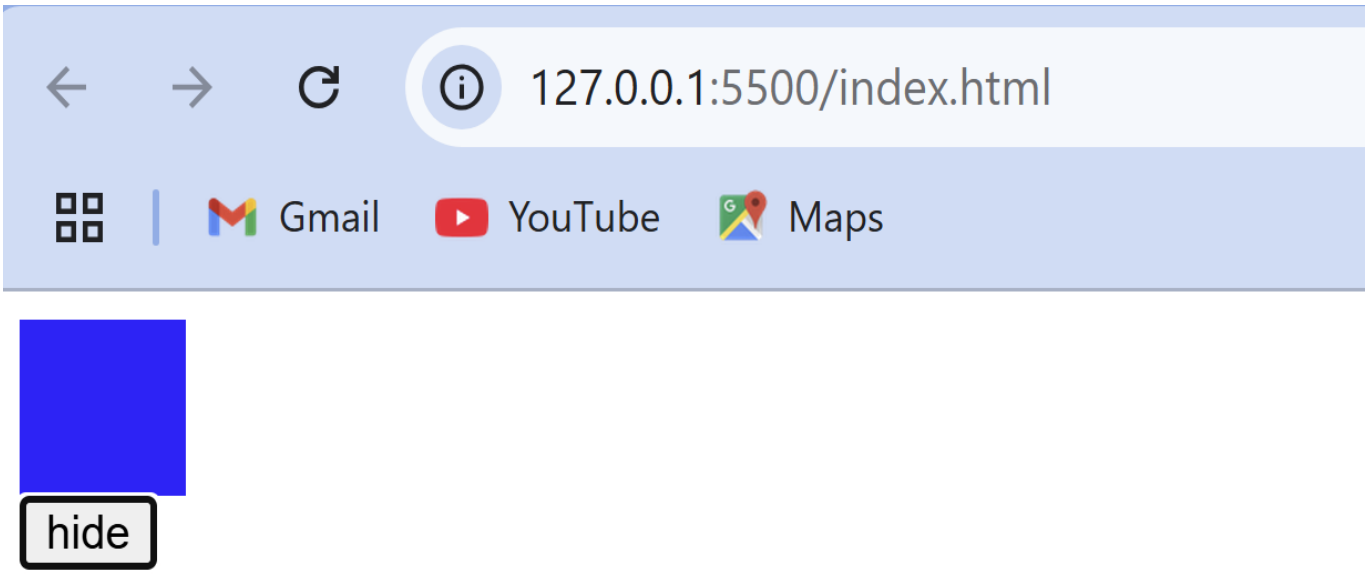


Task 5:

```
<!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="con" style="height:50px; width:50px;background-color: blue;" class="container">
Hello, world!</div>
    <input type="button" id="unique" onclick="Modify()"value="Modify">

    <script>

        function Modify(){

            var element=document.getElementById("con");
        console.log("current Class",element.getAttribute("class"));
        element.setAttribute("class","modified class");
        console.log("current Style",element.getAttribute("style"));
        element.setAttribute("style","height:50px; width:60px;background-color:pink");

            if(element.hasAttribute("id")){

               console.log("Id Exists:"+element.getAttribute("id"));
            }
           element.removeAttribute("container");
           console.log("New class"+element.getAttribute("class"));
        }
        </script>
</body>
</html>
```

Output:




127.0.0.1:5500/index.html
Gmail
YouTube
Maps
hide