JEYSAN.V - 717823F225

Week 3 & 4:

1. Recursion and stack:

Task 1:

main.js

```
function factorial(n){
    if(n<=0){
        return 1;
    }
    else{
        return n*factorial(n-1);
    }
}

console.log(factorial(5))
console.log(factorial(9))
```

Output

```
120
362880

=== Code Execution Successful ===
```

Task 2:

main.js

```
function Fibonacci(n){
    if(n<=1){
        return n;
    }
    else{
        return Fibonacci(n-1)+Fibonacci(n-2);
    }
}
for(let i=0;i<10;i++)
console.log(Fibonacci(i))
```

Output

```
0
1
1
2
3
5
8
13
21
34

=== Code Execution Successful ===
```

Task 3:

main.js

```
function Fibonacci(n){
    if (n == 0) return 1;
    if (n < 0) return 0;
    return Fibonacci(n-1)+Fibonacci(n-2)+Fibonacci(n-3);
}
for(let i=0;i<10;i++)
console.log(Fibonacci(i))
```

Output

```
1
1
2
4
7
13
24
44
81
149

=== Code Execution Successful ===
```

Tassk 4:

main.js

Share Run

```
function flatten(arr){
    let result=[];
    arr.forEach((num)=>{
        if(Array.isArray(num)){
            result=result.concat(flatten(num));
        }else{
            result.push(num);
        }
    })
    return result;
}

let arr=[[1,2,3],[4,5,6],[7,8,9]];
console.log(flatten(arr));
```

Output

```
[
  1, 2, 3, 4, 5,
  6, 7, 8, 9
]

=== Code Execution Successful ===
```

Task 5:

main.js

Share Run

```
function towerOfHanoi(n, source, auxiliary, destination) {
    if (n === 1) {
        console.log(`Move disk 1 from ${source} to ${destination}`);
        return;
    }
    towerOfHanoi(n - 1, source, destination, auxiliary);
    console.log(`Move disk ${n} from ${source} to ${destination}`);
    towerOfHanoi(n - 1, auxiliary, source, destination);
}
towerOfHanoi(3, 'A', 'B', 'C');
```

Output

```
Move disk 1 from A to C
Move disk 2 from A to B
Move disk 1 from C to B
Move disk 3 from A to C
Move disk 1 from B to A
Move disk 2 from B to C
Move disk 1 from A to C

=== Code Execution Successful ===
```

2. JSON and variable length arguments/spread syntax:

Task 1:

main.js

```
function sum(...args){
    total=0;
    for(let n of args){
        total+=n;
    }
    return total;
}

console.log(sum(2 ,3 ,4));
console.log(sum(2 ,3 ,4 ,7 ,3 ,8));
```

Output

```
9
27

=== Code Execution Successful ===
```

Task 2:

main.js

```
function sumOfArr(...args){
    total=0;
    for(let arr of args){
        for(let n of arr)
            total+=n;
    }
    return total;
}

console.log(sumOfArr([2 ,3 ,4]));
console.log(sumOfArr([2 ,3 ,4] ,[7 ,3 ,8]));
```

Output

```
9
27

=== Code Execution Successful ===
```

Task 3:

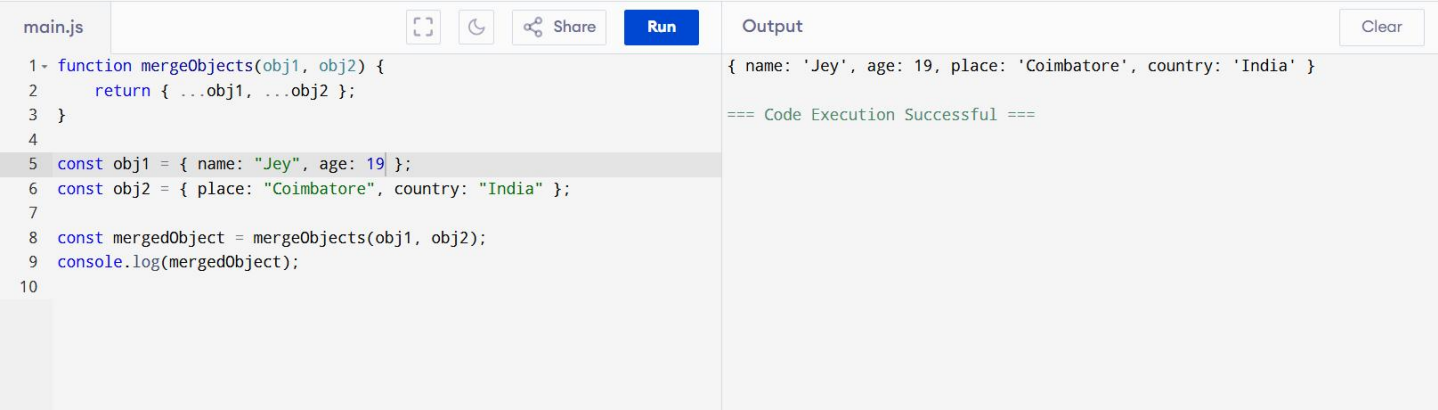
main.js

```
const originalObject = {
    name: "Jey",
    details: {
        age: 30,
        address: {
            city: "Coimbatore",
            country: "India"
        }
    }
};

const deepClone = JSON.parse(JSON.stringify(originalObject));
console.log(deepClone);
```

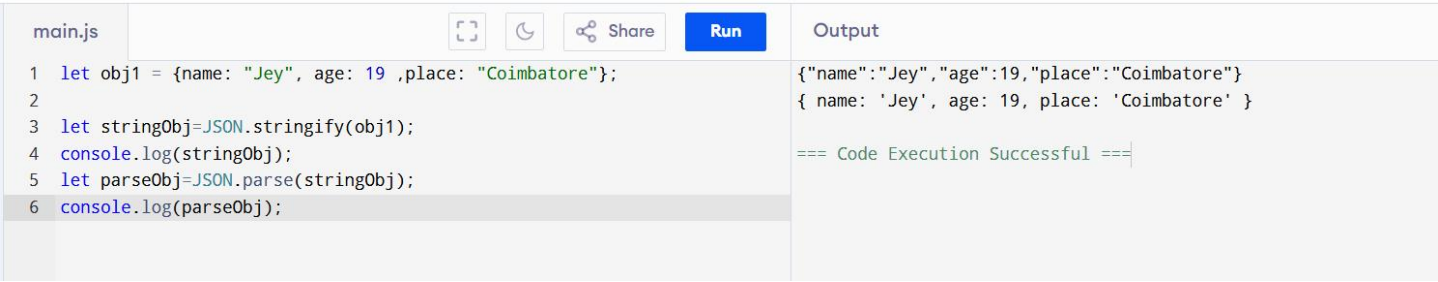
Output

```
{
  name: 'Jey',
  details: { age: 30, address: { city: 'Coimbatore', country: 'India' } }
}

=== Code Execution Successful ===
```

Task 4:

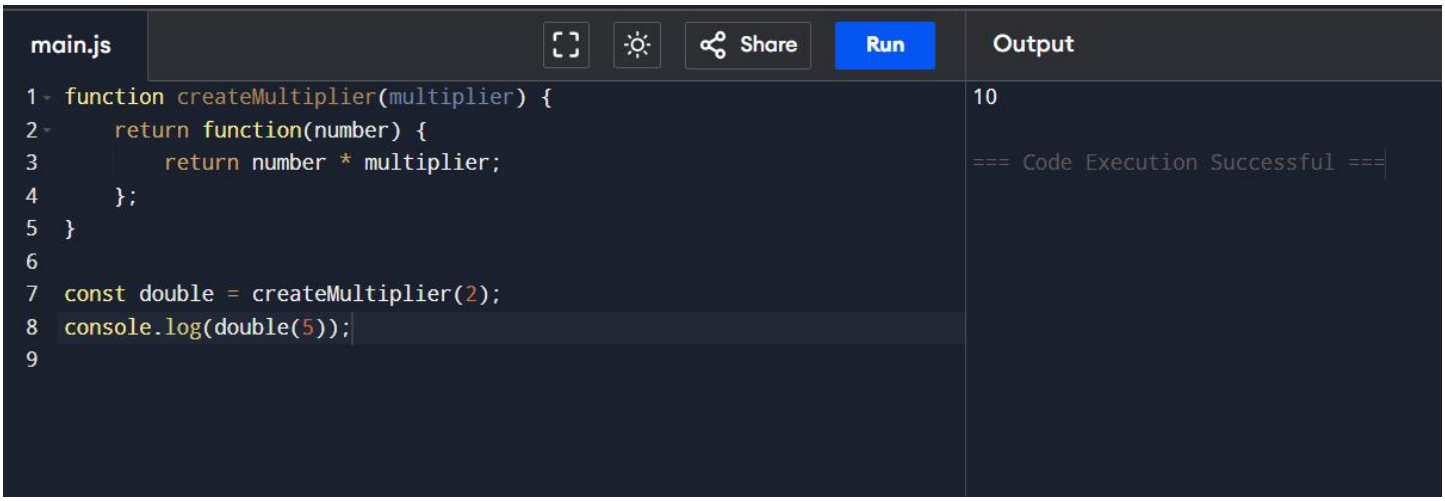


Task 5:



3. Closure:

Task 1:



Task 2:

main.js

```
function createCounter() {
    let count = 0;
    return {
        increment() {
            count++;
        },
        getCount() {
            return count;
        }
    };
}
const counter = createCounter();
console.log(counter.getCount());
counter.increment();
counter.increment();
counter.increment();
console.log(counter.getCount());
```

Output

```
0
3

=== Code Execution Successful ===
```

Task 3:

main.js

```
function createCounterFactory() {
    return function() {
        let count = 0;
        return {
            increment() {
                count++;
            },
            getCount() {
                return count;
            }
        };
    };
}

const newCounter = createCounterFactory();
const counter1 = newCounter();
const counter2 = newCounter();
counter1.increment();
console.log(counter1.getCount());
console.log(counter2.getCount());

```

Output

```
1
0

=== Code Execution Successful ===
```

Task 4:

main.js

```
function createPrivateCounter() {
    let privateCount = 0;
    return {
        increment() {
            privateCount++;
        },
        getCount() {
            return privateCount;
        }
    };
}

const privateCounter = createPrivateCounter();
privateCounter.increment();
console.log(privateCounter.getCount());

```

Output

```
1

=== Code Execution Successful ===
```

Task 5:

main.js

```
function createFunctionFactory(operation) {
    return function(a, b) {
        switch (operation) {
            case "add":
                return a + b;
            case "subtract":
                return a - b;
            default:
                return null;
        }
    };
}

const sub = createFunctionFactory("subtract");
console.log(sub(5, 3));

```

Output

```
2

=== Code Execution Successful ===
```

4. **Promise, Promises chaining**:

Task 1:

main.js

Share Run

Output

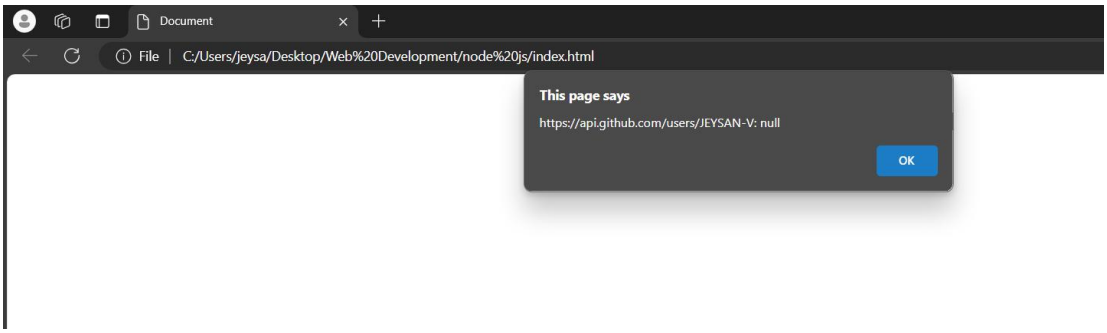
```
function delayedGreeting(seconds) {
    return new Promise((resolve) => {
        setTimeout(() => resolve("Greeting"), seconds * 1000);
    });
}

delayedGreeting(2).then(console.log);
```

```
Greeting

=== Code Execution Successful ===
```

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>

        let urls = [

            'https://api.github.com/users/JEYSAN-V'

        ];

        let requests = urls.map(url => fetch(url));

        Promise.all(requests)

            .then(responses =>

                Promise.all(responses.map(response => response.json()))

            )

            .then(data => data.forEach(user =>

                alert(`${user.url}: ${user.name}`)

            ));
```

```
        console.log(typeof request);

    </script>

</body>

</html>
```



Task 3:

main.js

```
function randomPromise() {
    return new Promise((resolve, reject) => {
        const rand = Math.random();
        rand > 0.5 ? resolve("Success!") : reject("Failure!");
    });
}

randomPromise()
    .then(console.log)
    .catch(console.error);

```

Share Run

Output

```
Success!

=== Code Execution Successful ===
```

Task 4:

index.html

C: > Users > student.AT-30 > index.html > html > body > script > urls

```
<!DOCTYPE html>
<html lang="en">
<head>
 <meta charset="UTF-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0"> <title>JavaScript</title> </head>
<body>
 <script>
 let urls = [
 'https://api.github.com/users/JEYSAN-V'
];
let requests = urls.map(url => fetch(url));
Promise.all(requests)
 .then(responses => responses.forEach(
 response =>console.log(`${response.url}: ${response.status}`) ));
 </script>
</body>
</html>
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

Filter (e.g. text, !exclude, \escape)

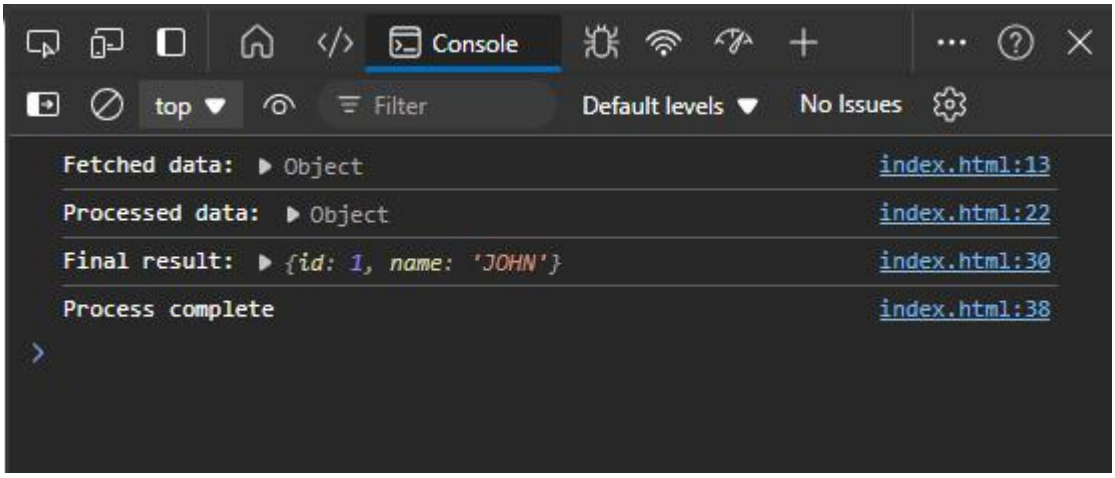
https://api.github.com/users/JEYSAN-V: 403

Task 5:

```
<!DOCTYPE html>
<html lang="en">
<head>
 <meta charset="UTF-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0"> <title>Promise Chain
Example</title>
</head>
<body>
 <script>
 function fetchData() {
 return new Promise((resolve) => {
 setTimeout(() => {
 const data = { id: 1, name: "John" };
 console.log('Fetched data:', data);
 resolve(data);
 }, 1000);
 });
 }
 function processData(data) {
 return new Promise((resolve) => {
 setTimeout(() => {
 data.name = data.name.toUpperCase();
 console.log('Processed data:', data);
 resolve(data);
 }, 1000);
 });
 }
 function logResult(data) {
```

```
return new Promise((resolve) => {
setTimeout(() => {
console.log('Final result:', data);
resolve('Process complete');
}, 1000);
});
}
fetchData()
.then(data => processData(data))
.then(processedData => logResult(processedData))
.then(result => console.log(result))
.catch(error => console.error('Error:', error));
</script>
</body>
</html>
```

Output:



5. Async/await:

Task 1:

main.js

```
async function delayedGreetingAsync(seconds) {
    await new Promise(resolve => setTimeout(resolve, seconds * 1000));
    console.log("Hello, world!");
}

delayedGreetingAsync(2);
```

Output

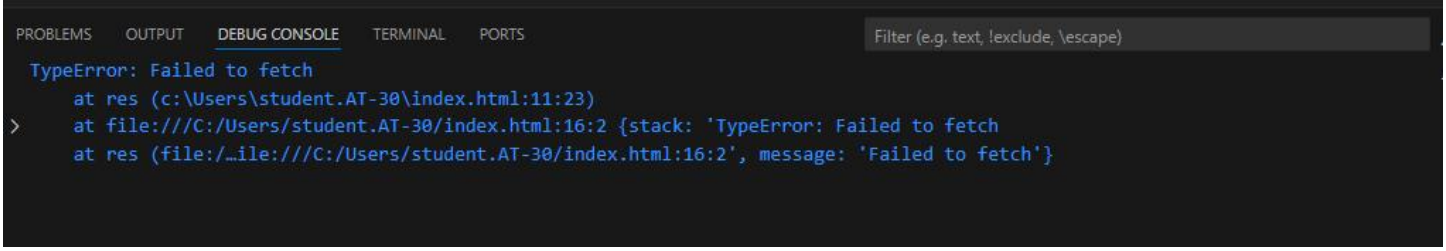
```
Hello, world!

=== Code Execution Successful ===
```

Task 2:

```
<!DOCTYPE html>
<html lang="en">
<head>
 <meta charset="UTF-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <title>Promise Chain Example</title>
</head>
<body>
 <script>
 data = async()=>{
 try{
 const url = await fetch('https://api.github.com/users/JEYSAN-V');
 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>
```

Task 3:

```
<!DOCTYPE html>
<html lang="en">
<head>
 <meta charset="UTF-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0"> <title>Promise Chain
Example</title>
</head>
<body>
 <script>
 async function res() {
 try {
 let response = await fetch('http://no-such-url');
 } catch (err) {
 console.log(err);
 }
 }
 res();
 </script>
</body>
</html>
```



Task 4:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Promise Chain Example</title>
</head>
<body>
  <script>
    async function getData() {
      let urls = [
        'https://api.github.com/users/JEYSAN-V'
      ];
      try {
        let requests = await Promise.all(urls.map(url => fetch(url)));

        let data = await Promise.all(requests.map(res => res.json()));

        return data;
```

```
    } catch (error) {
      console.error('Error:', error);
    }
  }
  getData().then(responses => {
    if (responses) {
      responses.forEach(response => {
        console.log(`${response.name}: ${response.login}`);
      });
    }
  }).catch(error => {
    console.error('Error:', error);
  });
 </script>
</body>
</html>
```

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

undefined: undefined

Task 5:

main.js Share Run Output

```
async function waitForAll() {
    const tasks = [
        new Promise(resolve => setTimeout(() => resolve("Task 1 done"
            ), 1000)),
        new Promise(resolve => setTimeout(() => resolve("Task 2 done"
            ), 2000)),
    ];
    const results = await Promise.all(tasks);
    console.log(results);
}

waitForAll();

```

```
[ 'Task 1 done', 'Task 2 done' ]

=== Code Execution Successful ===
```

6. Modules introduction, Export and Import:

Task 1:

main.js

```
export const variable = "Hello";
export function greet() {
    return "Hello, World!";
}
export class Greeter {
    sayHello() {
        return "Hi!";
    }
}
console.log("export successful");
```

Output

```
export successful

=== Code Execution Successful ===
```

Task 2:

Welcome | firstProgram.js | greet.js | samplegreet.js

samplegreet.js

```
import { greet, Greeter, variable } from './greet.js';

console.log(variable);
console.log(greet());
console.log(new Greeter().sayHello());

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS SEARCH ERROR SPELL CHECKER

```
[Running] node "c:\Users\jeysa\Desktop\Web Development\node js\samplegreet.js"
Hello
Hello, World!
Hi!
```

Task 3 & 4:

greet.js

```
export const variable = "Hello";
export function greet() {
    return "Hello, World!";
}
export class Greeter {
    sayHello() {
        return "Hi!";
    }
}
```

Welcome | firstProgram.js | greet.js | samplegreet.js

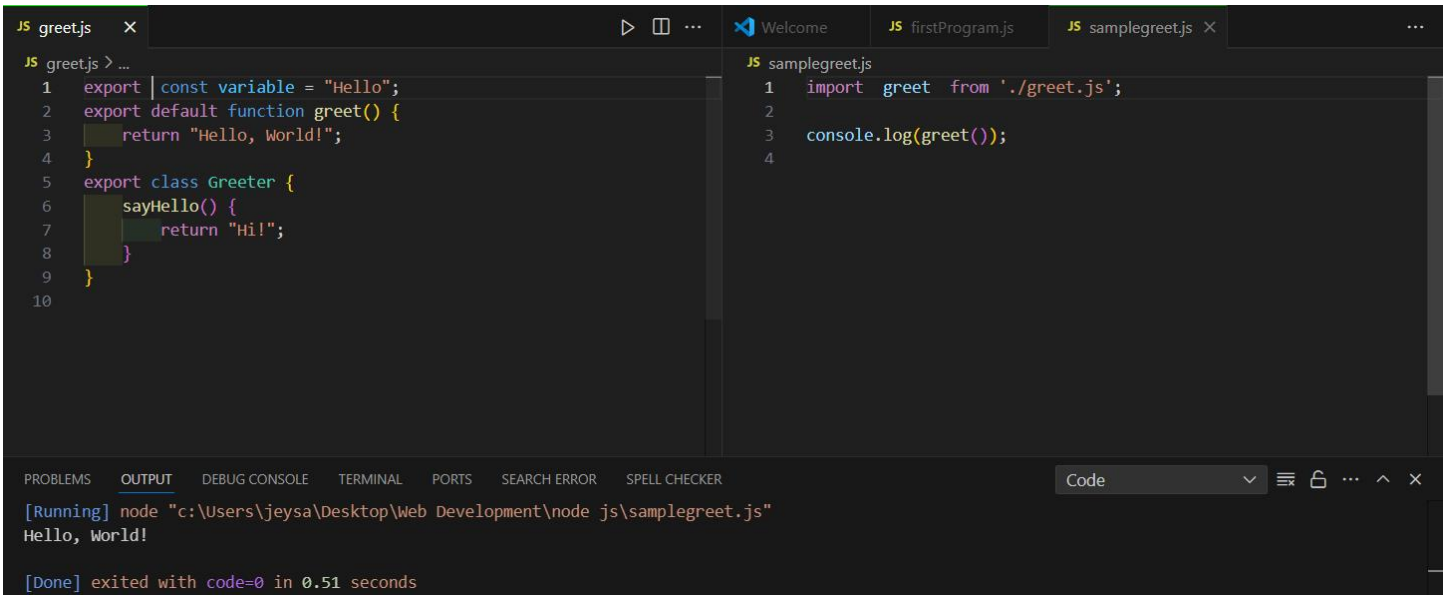
samplegreet.js

```
import { greet } from './greet.js';

console.log(greet());
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS SEARCH ERROR SPELL CHECKER

```
[Running] node "c:\Users\jeysa\Desktop\Web Development\node js\samplegreet.js"
Hello, World!

[Done] exited with code=0 in 0.51 seconds
```

Task 5:

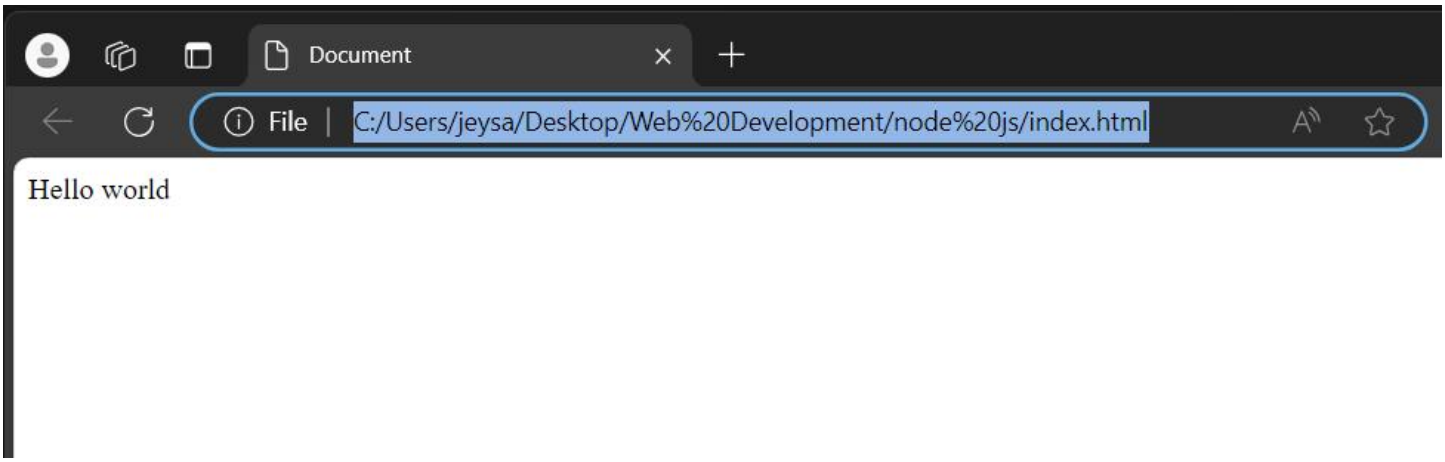


7. **Browser: DOM Basics**:

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>

    <div id="a"></div>

    <script>

        document.getElementById('a').textContent = "Hello world";



    </script>

</body>

</html>
```

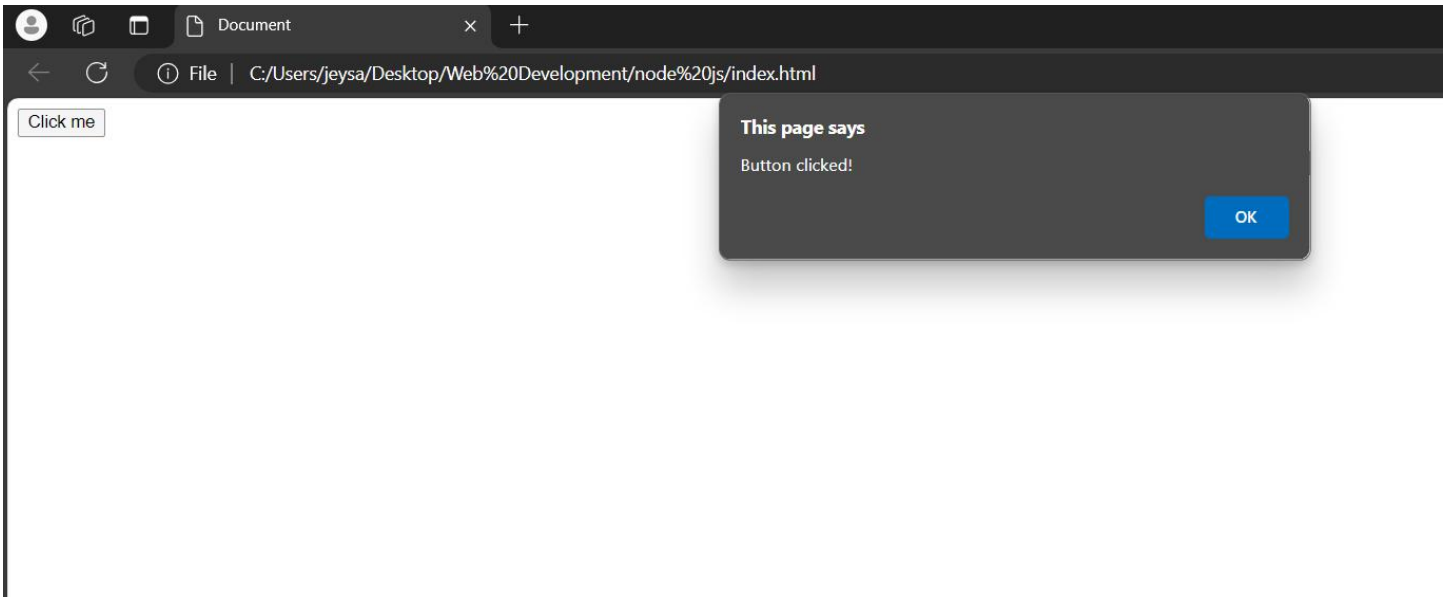
Output:



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>
    <button id="btn">Click me</button>


    </button>
    <script>
    document.getElementById('btn').addEventListener('click', () => {
        alert("Button clicked!");
    });


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

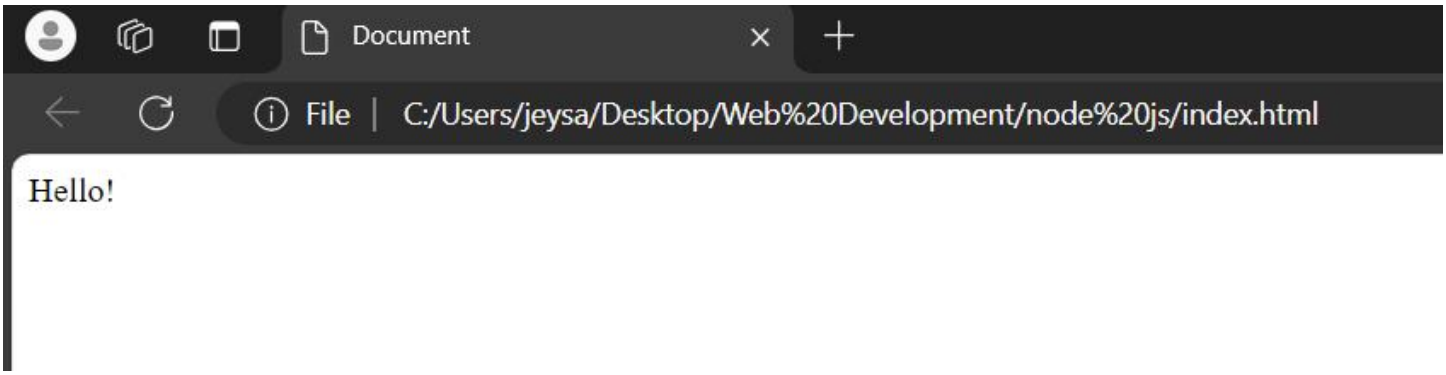
Output:



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>

        const newElement = document.createElement('div');

        newElement.textContent = "Hello!";

        document.body.appendChild(newElement);

    </script>

</body>

</html>
```

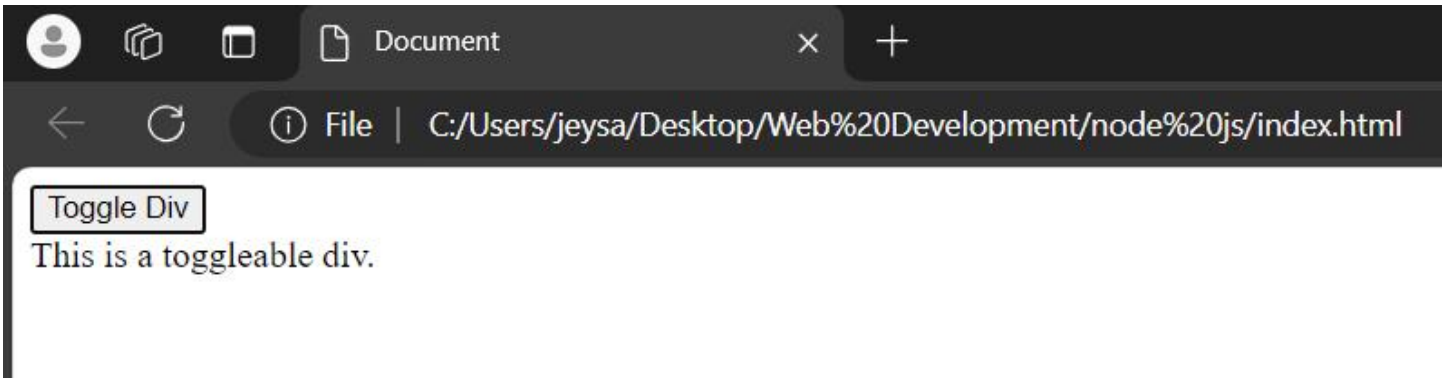
Output:



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>
    <button onclick="toggleVisibility('toggleDiv')">Toggle Div</button>
    <div id="toggleDiv" style="display: flex;">
        This is a toggleable div.
    </div>
    <script>
        function toggleVisibility(id) {
            const element = document.getElementById(id);
            element.style.display = element.style.display === "none" ? "block" : "none";
        }
    </script>
</body>
</html>
```

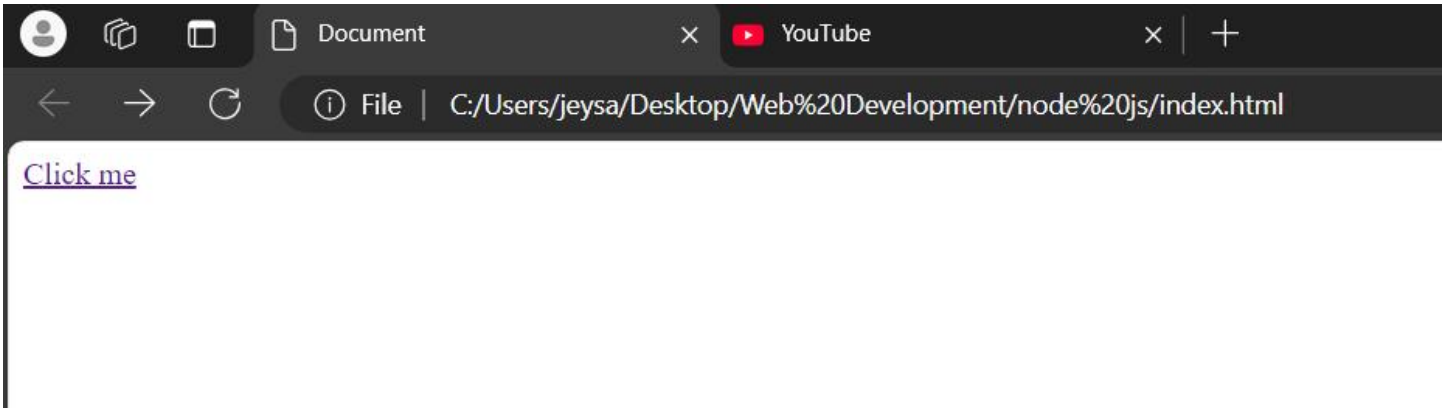
Output:



Task 5:

```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>

    <a id="linkId" href="google.com" target="_blank">Click me</a>

    <script>

        const element = document.getElementById('linkId');

        console.log(element.getAttribute('href'));

        element.setAttribute('href', 'https://www.youtube.com');


    </script>

</body>

</html>
```

Output:


Document
YouTube
File | C:/Users/jeysa/Desktop/Web%20Development/node%20js/index.html
Click me