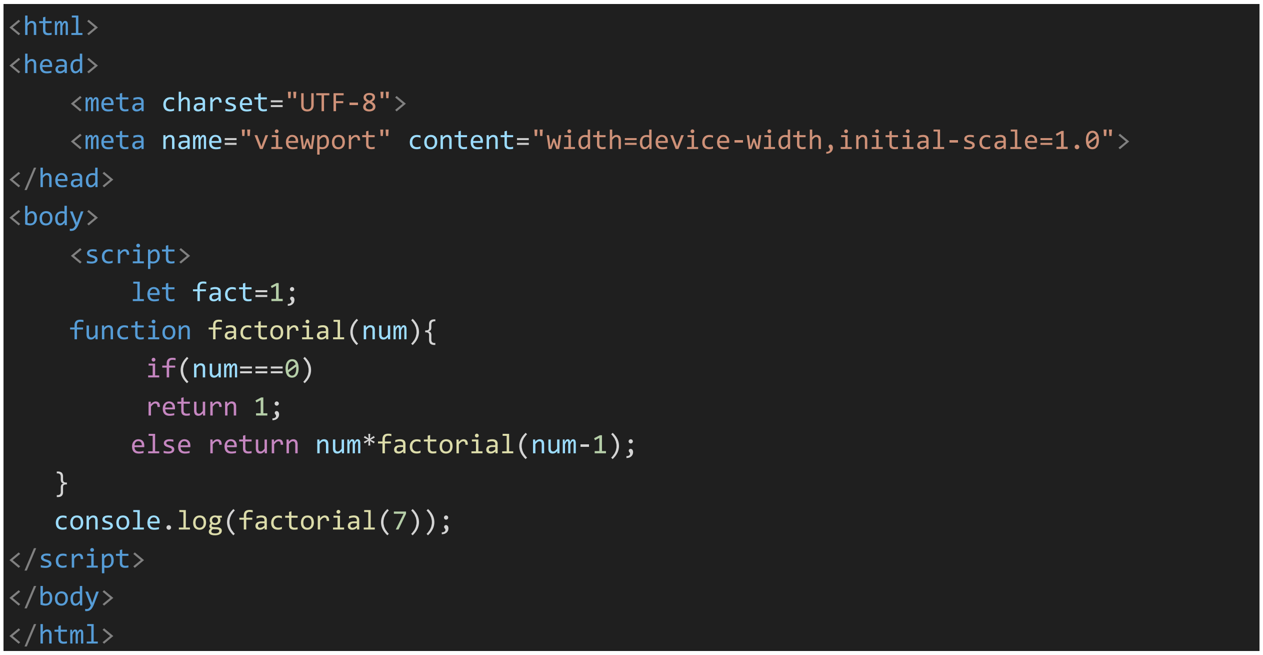
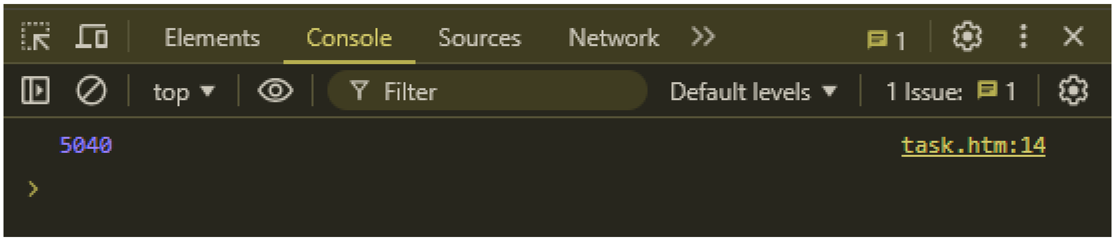
1. Recursion and stack:

TASK 1



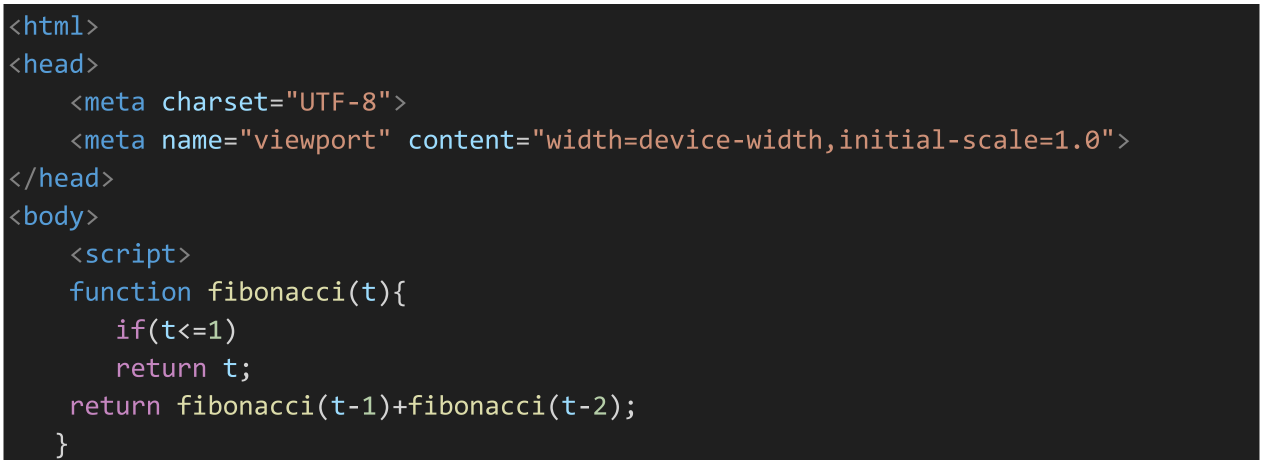
```
<html>
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width,initial-scale=1.0">
</head>
<body>
    <script>
        let fact=1;
    function factorial(num){
         if(num===0)
         return 1;
        else return num*factorial(num-1);
   }
   console.log(factorial(7));
</script>
</body>
</html>
```

OUTPUT:

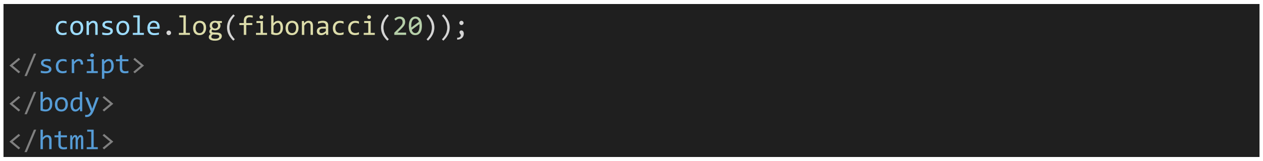


```
5040                                             task.htm:14
>
```

TASK 2

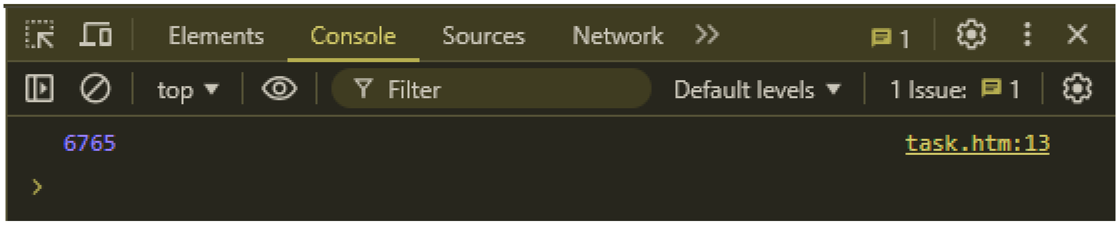


```
<html>
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width,initial-scale=1.0">
</head>
<body>
    <script>
    function fibonacci(t){
       if(t<=1)
       return t;
    return fibonacci(t-1)+fibonacci(t-2);
   }
```



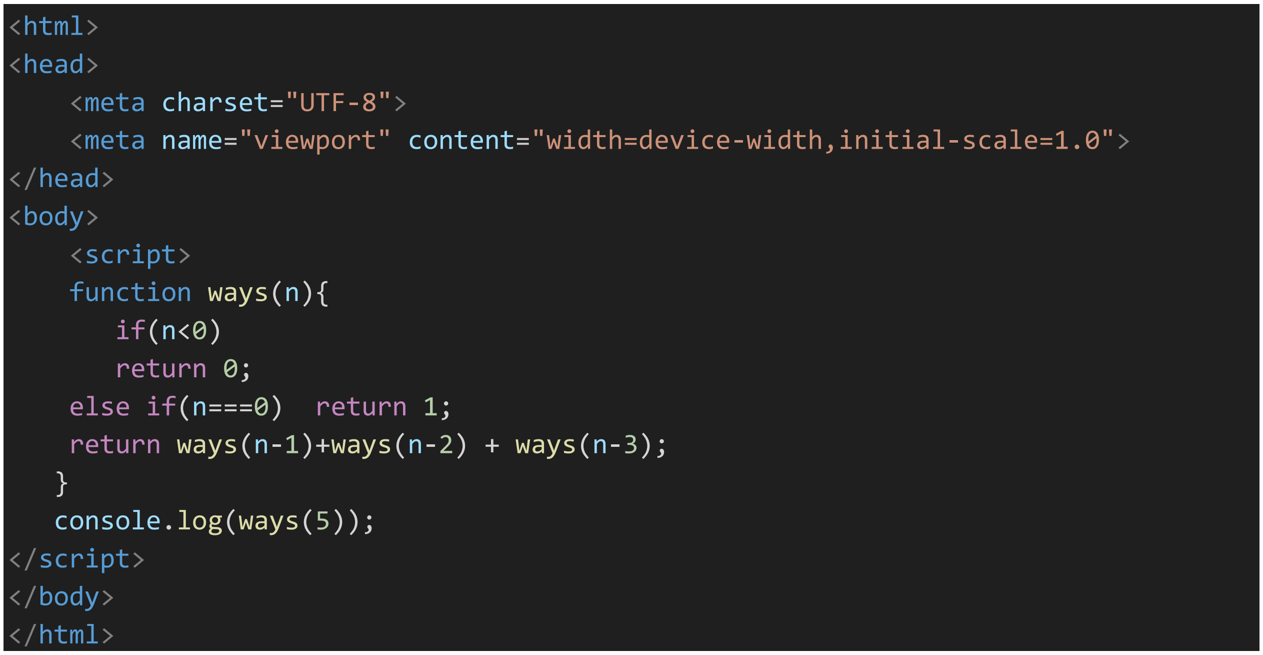
```
    console.log(fibonacci(20));
</script>
</body>
</html>
```

OUTPUT:



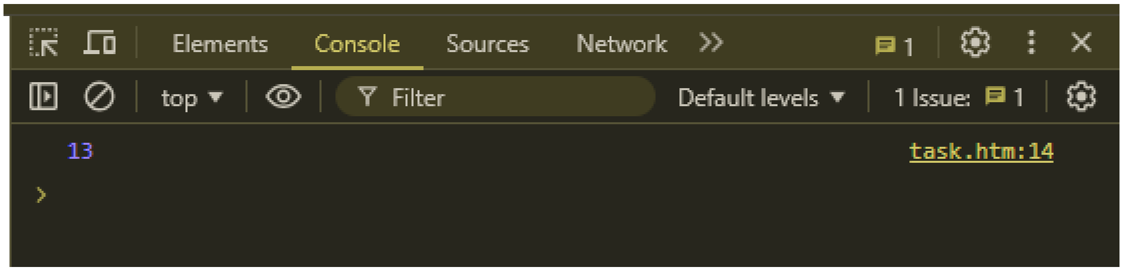
```
6765                                                        task.htm:13
>
```

TASK 3



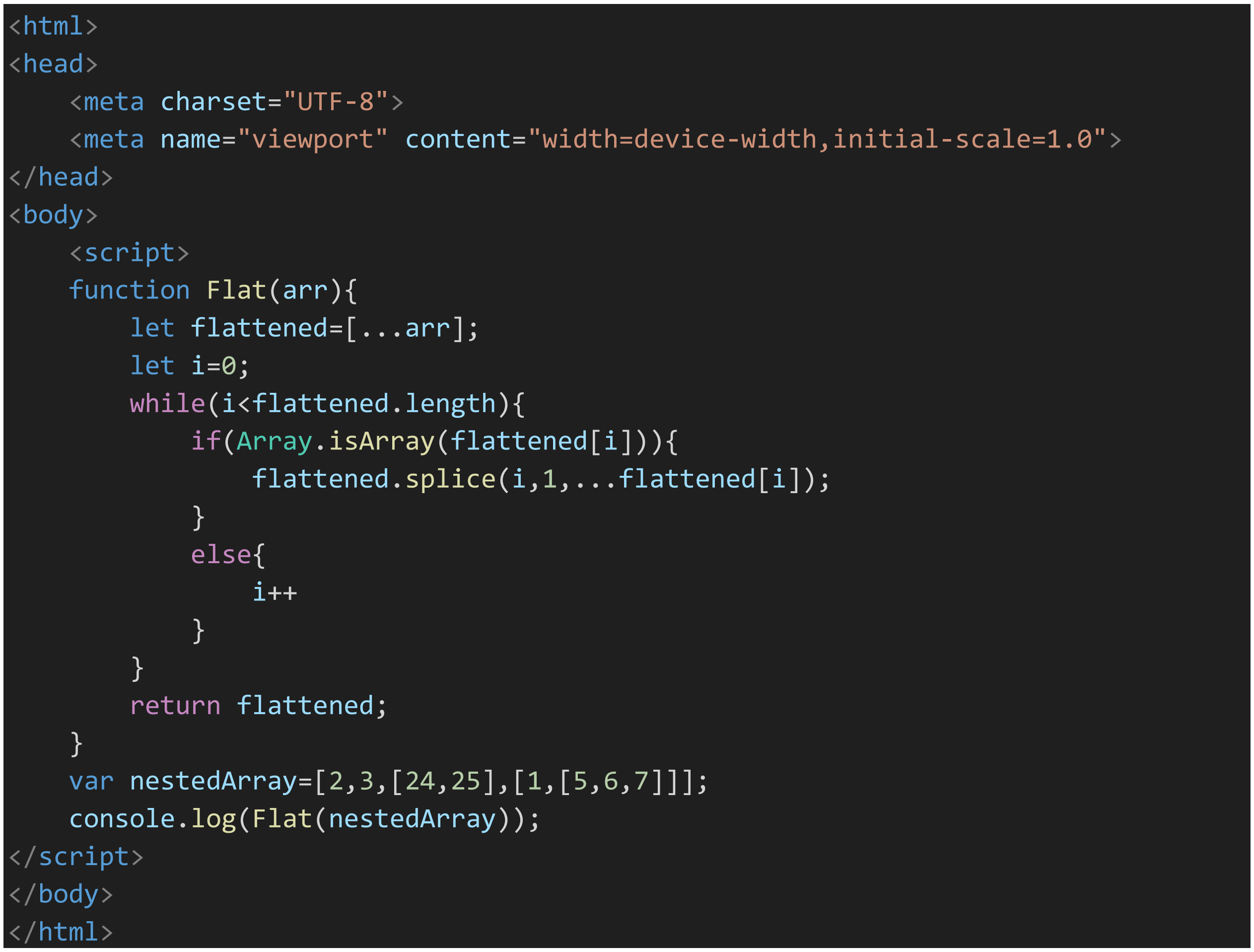
```
<html>
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width,initial-scale=1.0">
</head>
<body>
    <script>
    function ways(n){
       if(n<0)
       return 0;
    else if(n===0)  return 1;
    return ways(n-1)+ways(n-2) + ways(n-3);
   }
   console.log(ways(5));
</script>
</body>
</html>
```

OUTPUT:



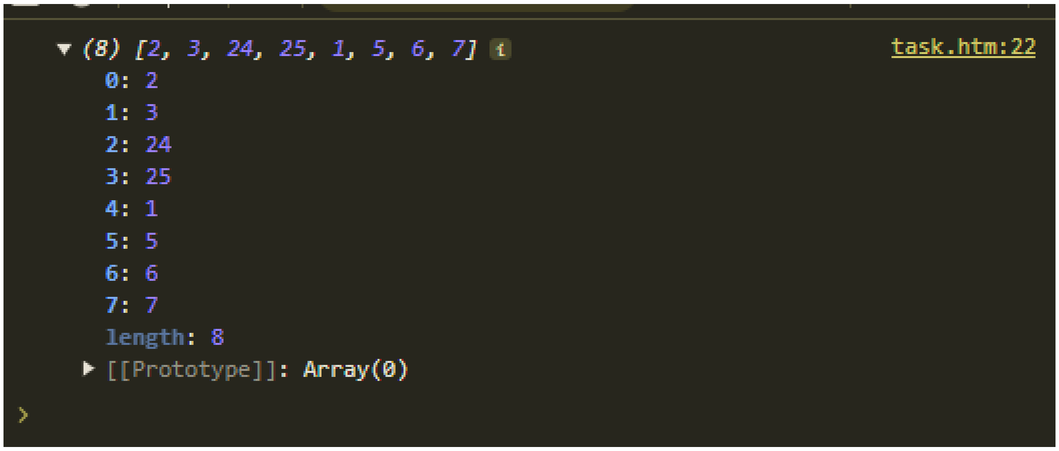
```
13                                                          task.htm:14
>
```

TASK 4



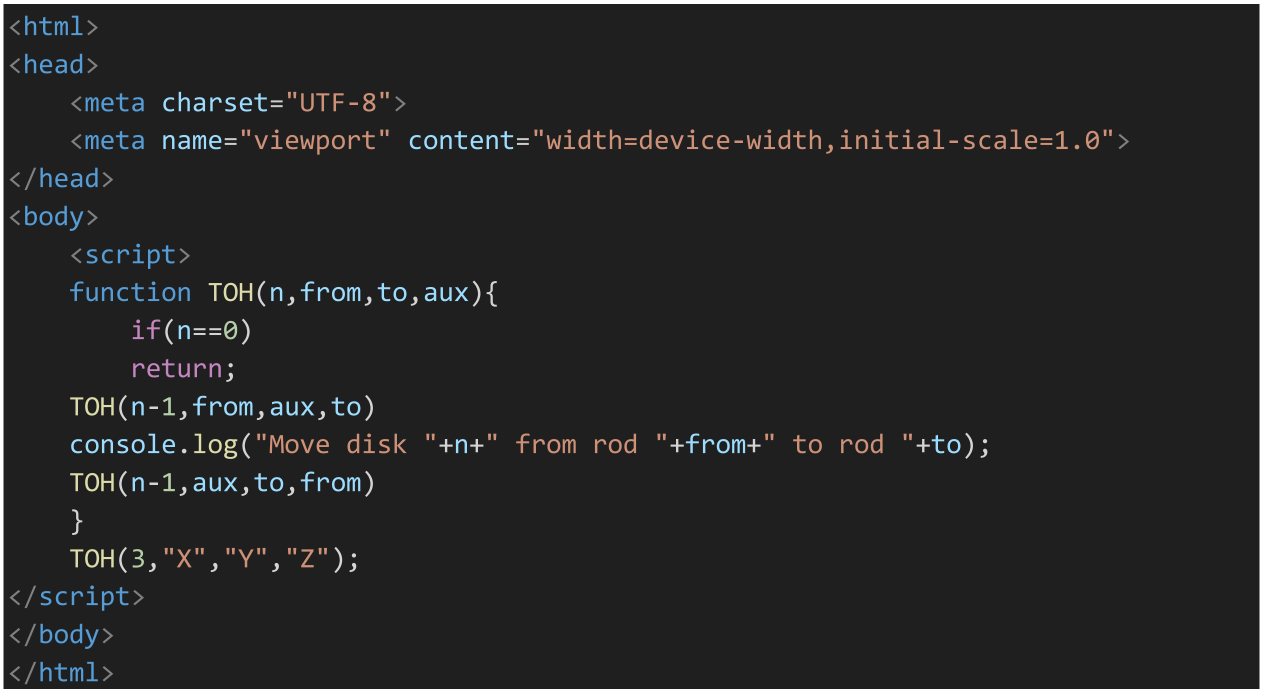
```
<html>
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width,initial-scale=1.0">
</head>
<body>
    <script>
    function Flat(arr){
        let flattened=[...arr];
        let i=0;
        while(i<flattened.length){
            if(Array.isArray(flattened[i])){
                flattened.splice(i,1,...flattened[i]);
            }
            else{
                i++
            }
        }
        return flattened;
    }
    var nestedArray=[2,3,[24,25],[1,[5,6,7]]];
    console.log(Flat(nestedArray));
</script>
</body>
</html>
```

OUTPUT:



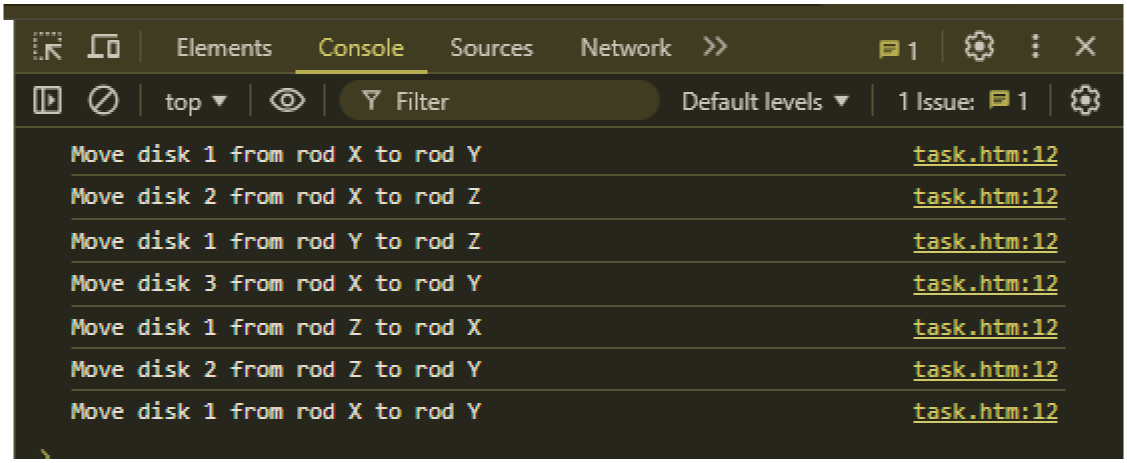
```
  ▼ (8) [2, 3, 24, 25, 1, 5, 6, 7]                              task.htm:22
      0: 2
      1: 3
      2: 24
      3: 25
      4: 1
      5: 5
      6: 6
      7: 7
      length: 8
    ▶ [[Prototype]]: Array(0)
>
```

TASK 5



```
<html>
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width,initial-scale=1.0">
</head>
<body>
    <script>
    function TOH(n,from,to,aux){
        if(n==0)
        return;
    TOH(n-1,from,aux,to)
    console.log("Move disk "+n+" from rod "+from+" to rod "+to);
    TOH(n-1,aux,to,from)
    }
    TOH(3,"X","Y","Z");
</script>
</body>
</html>
```

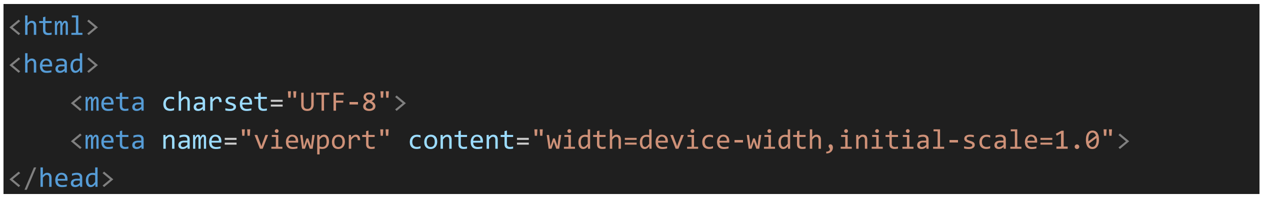
OUTPUT:



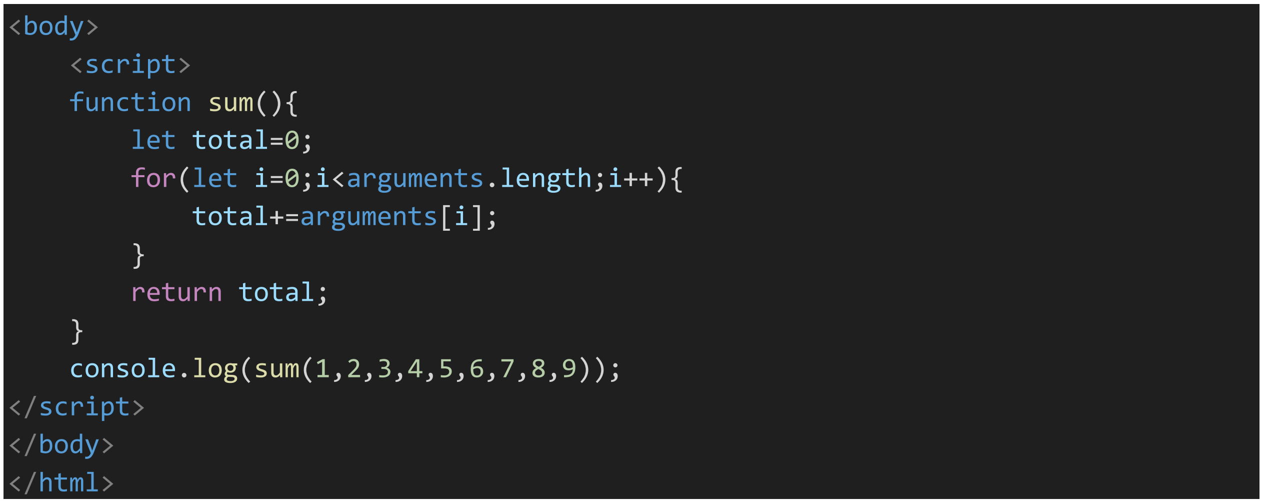
```
Move disk 1 from rod X to rod Y                          task.htm:12
Move disk 2 from rod X to rod Z                          task.htm:12
Move disk 1 from rod Y to rod Z                          task.htm:12
Move disk 3 from rod X to rod Y                          task.htm:12
Move disk 1 from rod Z to rod X                          task.htm:12
Move disk 2 from rod Z to rod Y                          task.htm:12
Move disk 1 from rod X to rod Y                          task.htm:12
```

2. JSON and variable length arguments/spread syntax:

TASK 1

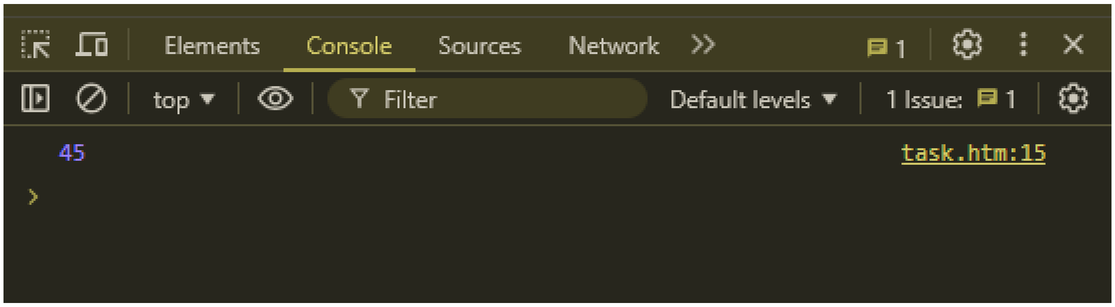


```
<html>
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width,initial-scale=1.0">
</head>
```



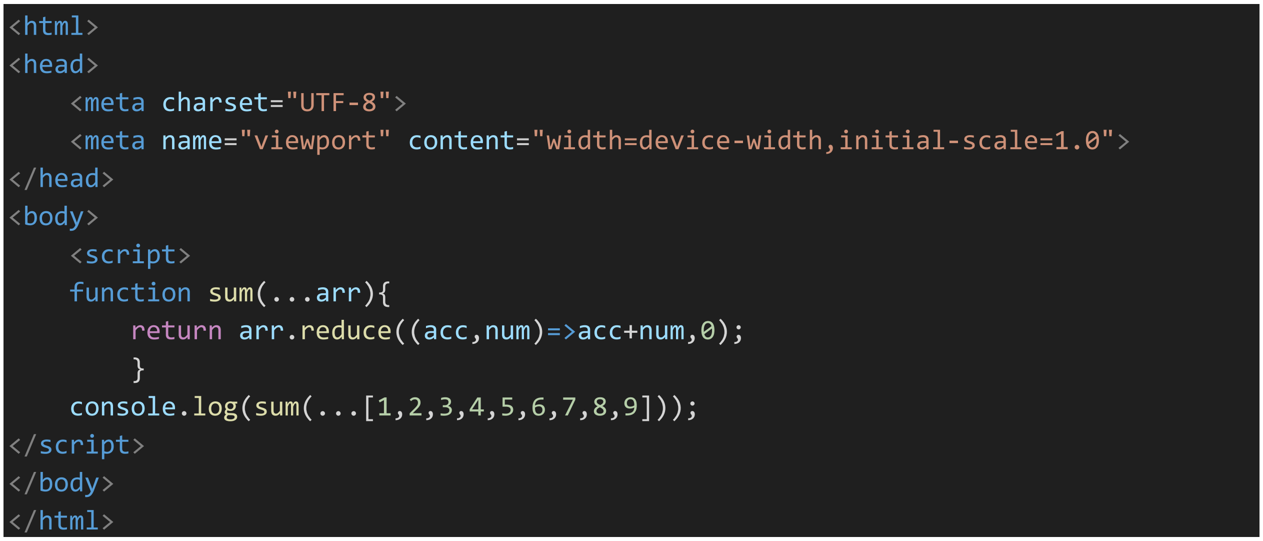
```
<body>
    <script>
    function sum(){
        let total=0;
        for(let i=0;i<arguments.length;i++){
            total+=arguments[i];
        }
        return total;
    }
    console.log(sum(1,2,3,4,5,6,7,8,9));
</script>
</body>
</html>
```

OUTPUT:



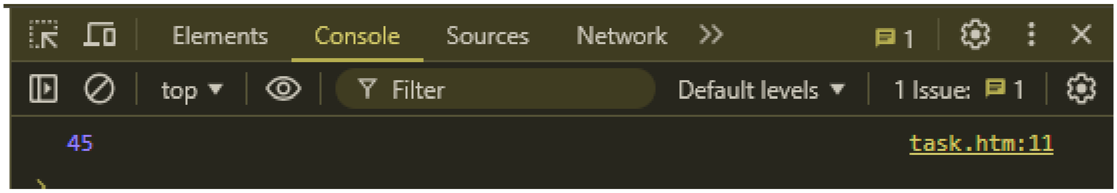
```
45                                                    task.htm:15
>
```

TASK 2



```
<html>
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width,initial-scale=1.0">
</head>
<body>
    <script>
    function sum(...arr){
        return arr.reduce((acc,num)=>acc+num,0);
        }
    console.log(sum(...[1,2,3,4,5,6,7,8,9]));
</script>
</body>
</html>
```

OUTPUT:

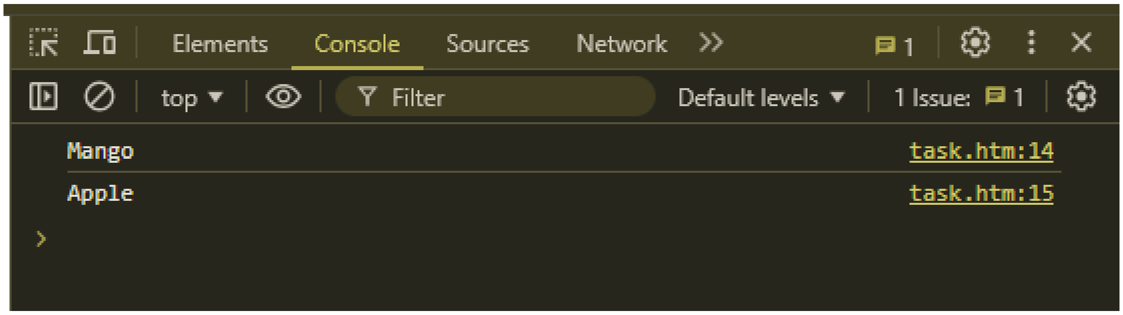


```
45                                                    task.htm:11
```

TASK 3

```
<html>
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width,initial-scale=1.0">
</head>
<body>
    <script>
    let fruit1={
       name:"Apple",
       Price:"52.0"
    }
   let fruit2=JSON.parse(JSON.stringify(fruit1));

    fruit1.name="Mango";

    console.log(fruit1.name);
    console.log(fruit2.name);
</script>
</body>
</html>
```

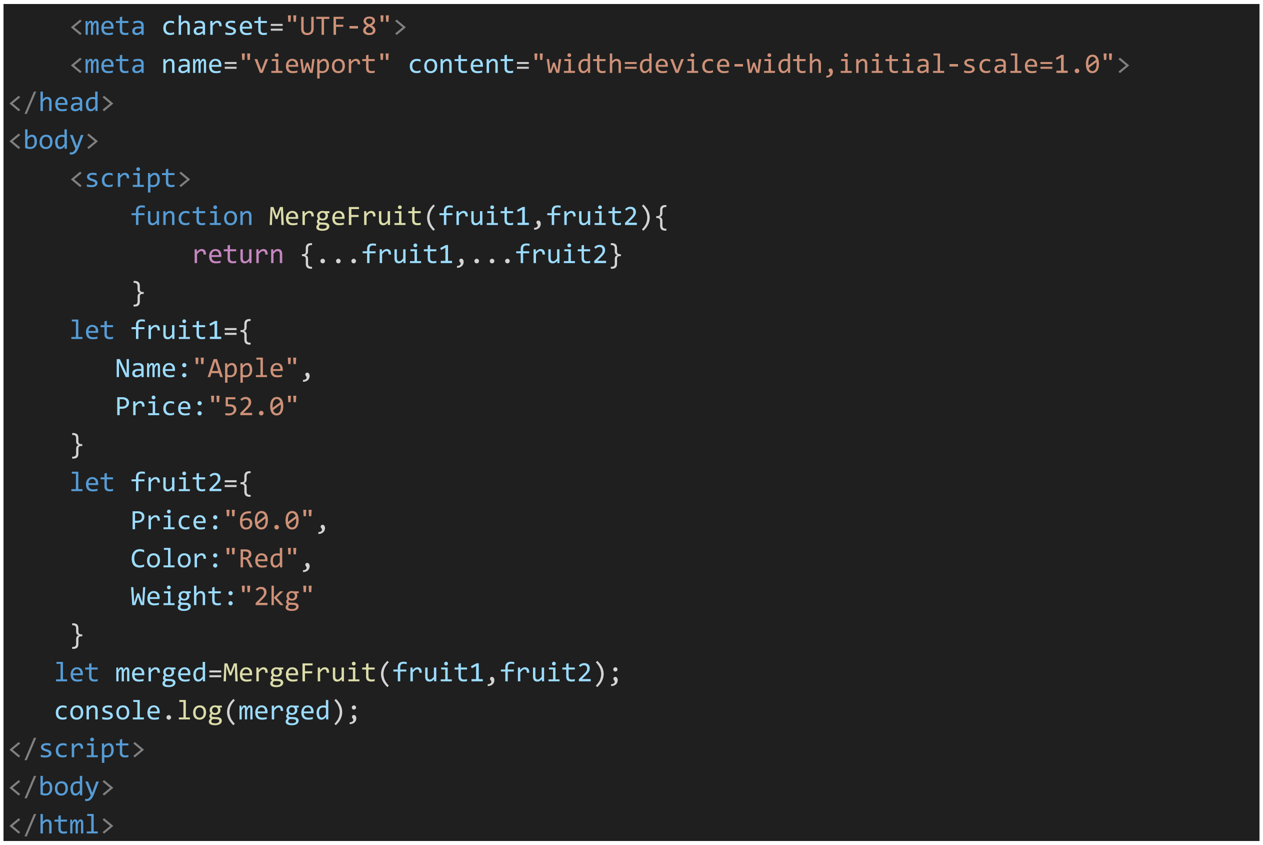
OUTPUT:



```
Mango                                                 task.htm:14
Apple                                                 task.htm:15
```

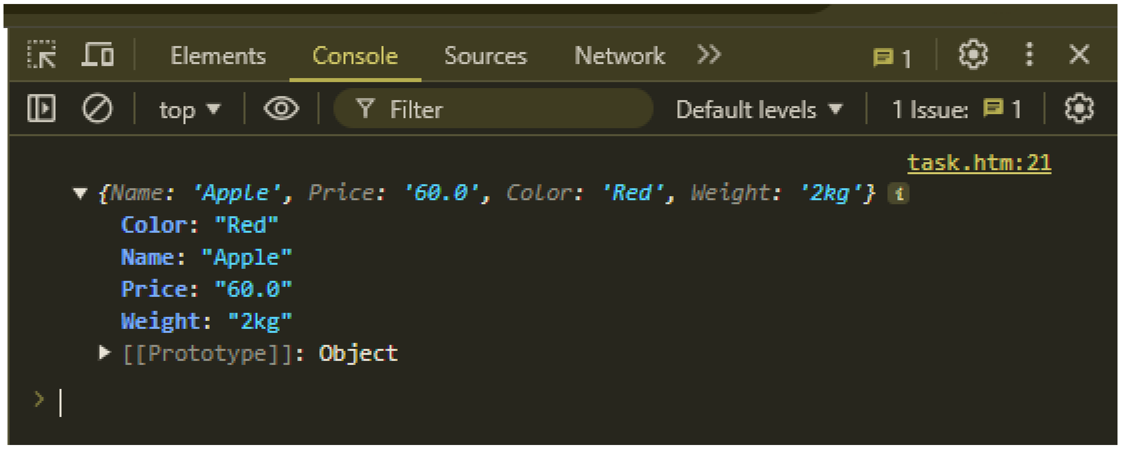
TASK 4

```
<html>
<head>
```



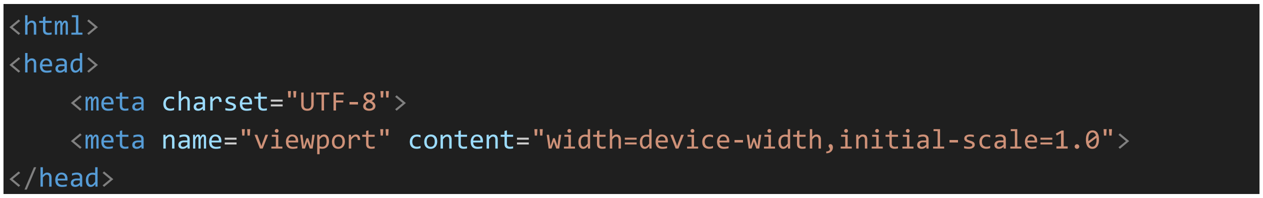
```
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width,initial-scale=1.0">
</head>
<body>
    <script>
        function MergeFruit(fruit1,fruit2){
            return {...fruit1,...fruit2}
        }
    let fruit1={
       Name:"Apple",
       Price:"52.0"
    }
    let fruit2={
        Price:"60.0",
        Color:"Red",
        Weight:"2kg"
    }
   let merged=MergeFruit(fruit1,fruit2);
   console.log(merged);
</script>
</body>
</html>
```

OUTPUT:

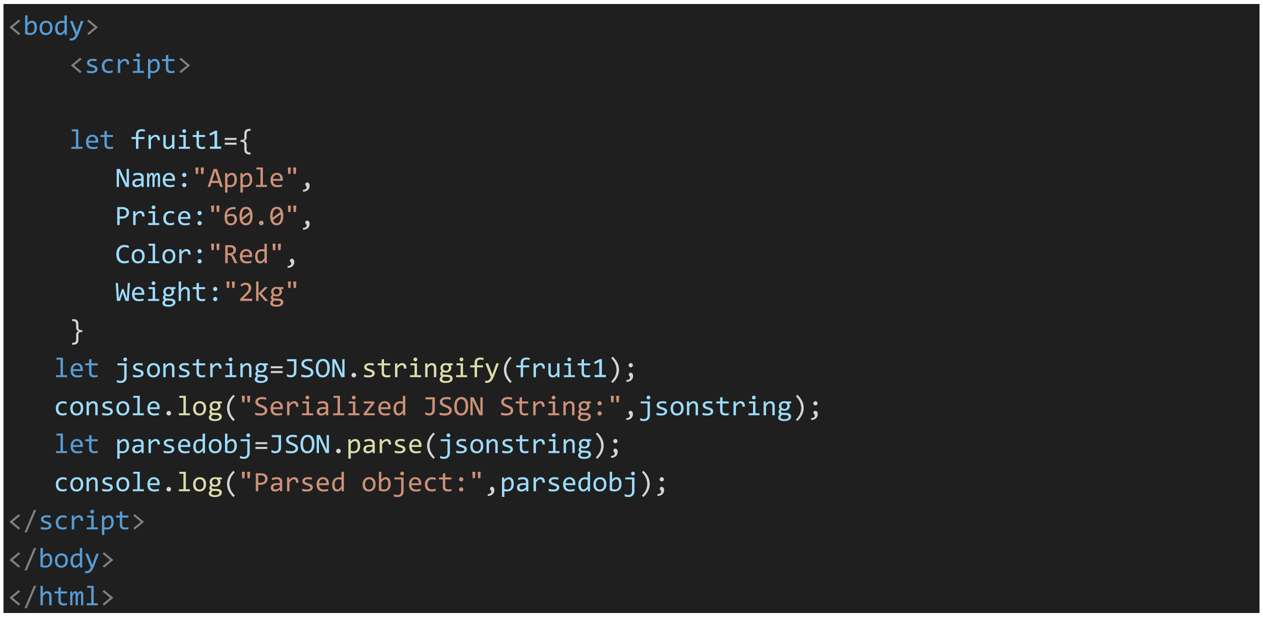


```
                                                                    task.htm:21
  ▼ {Name: 'Apple', Price: '60.0', Color: 'Red', Weight: '2kg'}
      Color: "Red"
      Name: "Apple"
      Price: "60.0"
      Weight: "2kg"
    ▶ [[Prototype]]: Object
>
```

TASK 5

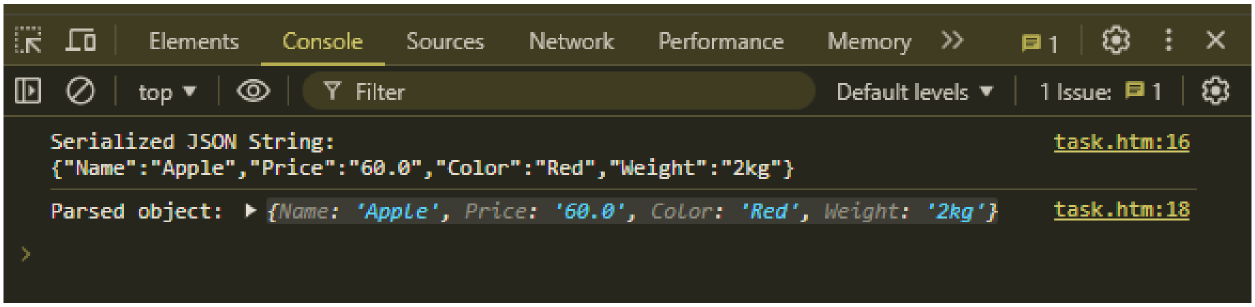


```
<html>
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width,initial-scale=1.0">
</head>
```



```
<body>
    <script>

    let fruit1={
       Name:"Apple",
       Price:"60.0",
       Color:"Red",
       Weight:"2kg"
    }
   let jsonstring=JSON.stringify(fruit1);
   console.log("Serialized JSON String:",jsonstring);
   let parsedobj=JSON.parse(jsonstring);
   console.log("Parsed object:",parsedobj);
</script>
</body>
</html>
```

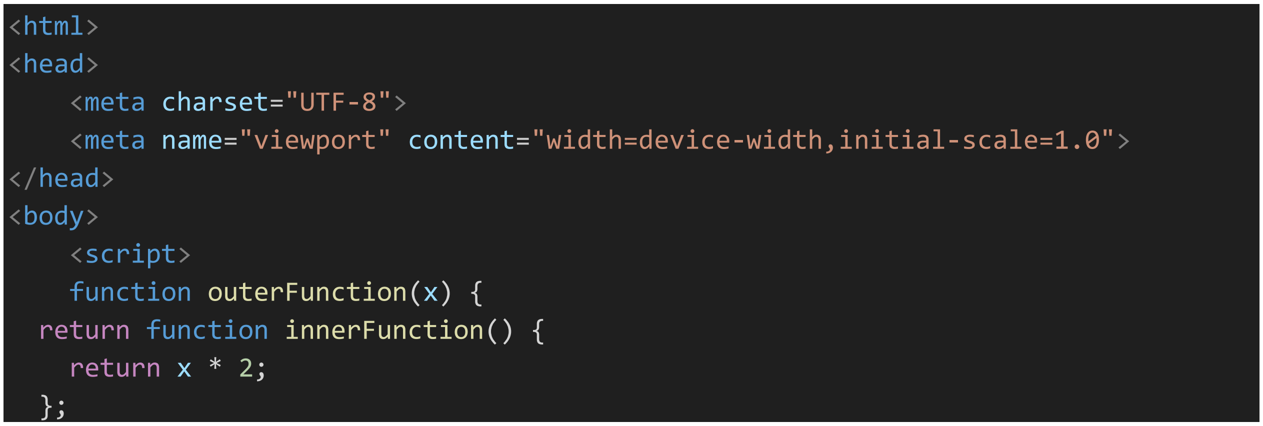
OUTPUT:



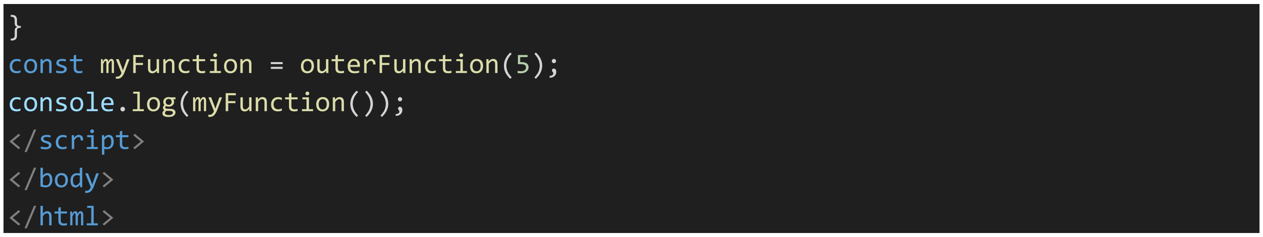
```
Serialized JSON String:                                                    task.htm:16
{"Name":"Apple","Price":"60.0","Color":"Red","Weight":"2kg"}
Parsed object:  ▶ {Name: 'Apple', Price: '60.0', Color: 'Red', Weight: '2kg'}  task.htm:18
>
```

3. Closure:

TASK 1

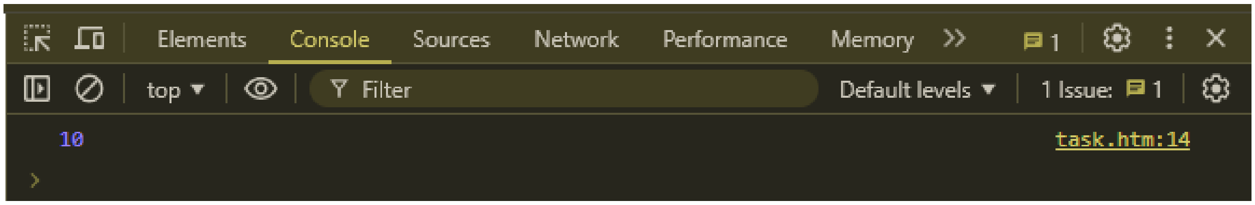


```
<html>
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width,initial-scale=1.0">
</head>
<body>
    <script>
    function outerFunction(x) {
  return function innerFunction() {
    return x * 2;
  };
```



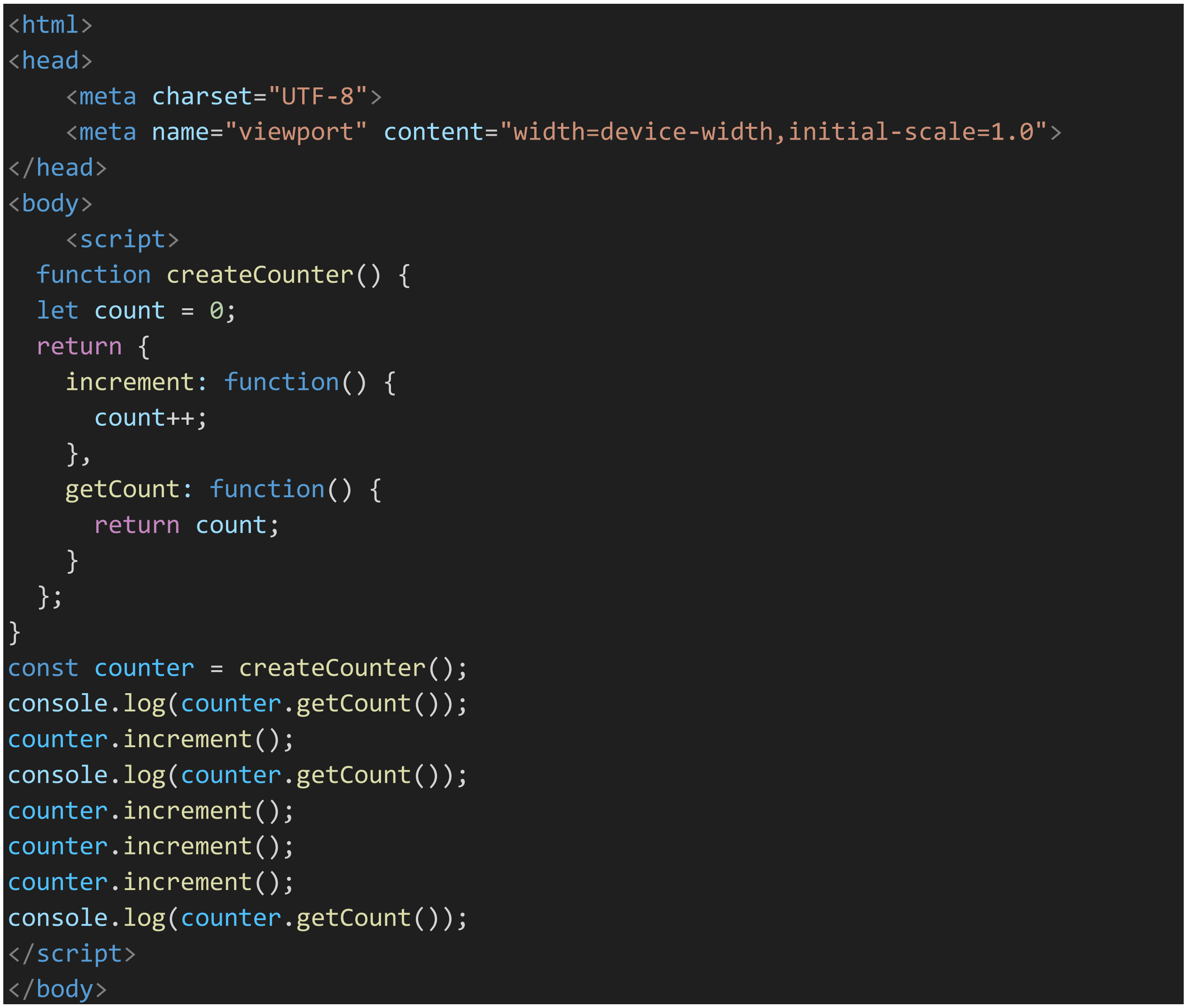
```
}
const myFunction = outerFunction(5);
console.log(myFunction());
</script>
</body>
</html>
```

OUTPUT:



```
10                                                              task.htm:14
>
```

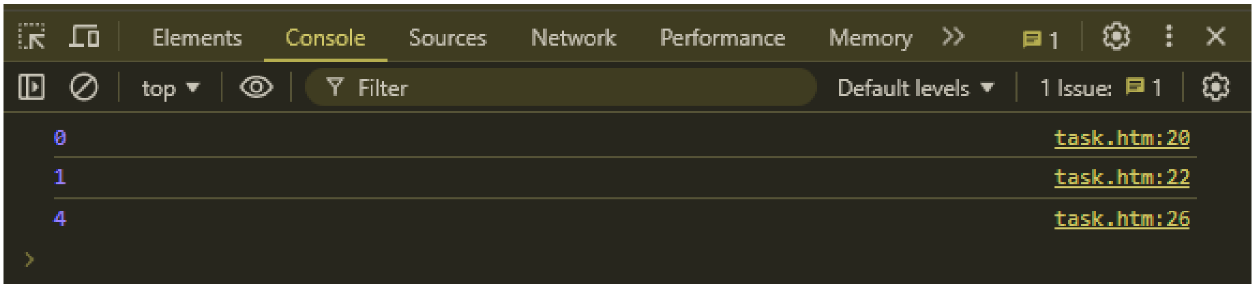
TASK 2



```
<html>
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width,initial-scale=1.0">
</head>
<body>
    <script>
  function createCounter() {
  let count = 0;
  return {
    increment: function() {
      count++;
    },
    getCount: function() {
      return count;
    }
  };
}
const counter = createCounter();
console.log(counter.getCount());
counter.increment();
console.log(counter.getCount());
counter.increment();
counter.increment();
counter.increment();
console.log(counter.getCount());
</script>
</body>
```

```
</html>
```

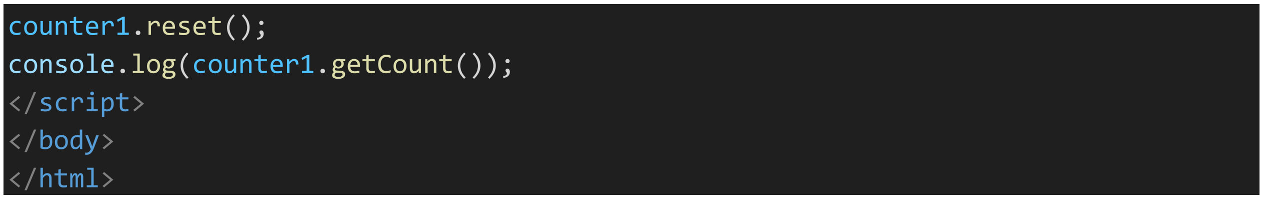
OUTPUT:



```
0                                   task.htm:20
1                                   task.htm:22
4                                   task.htm:26
>
```

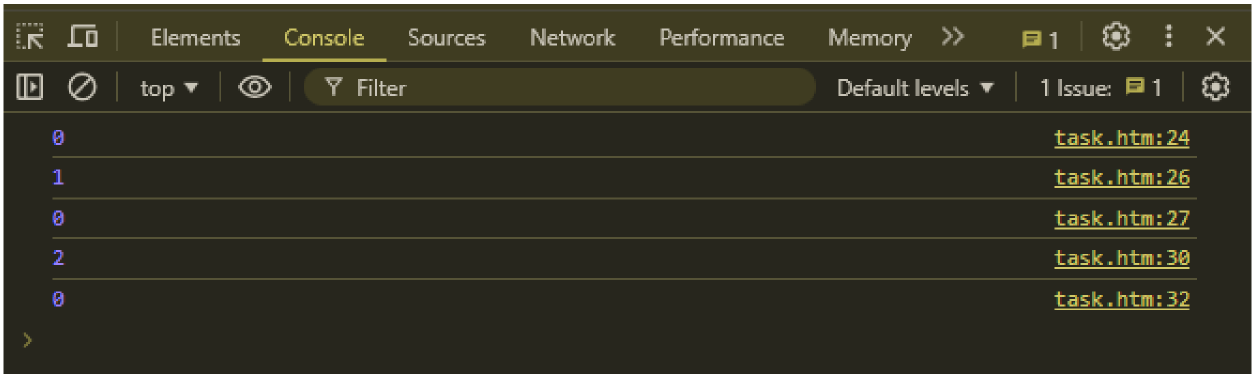
TASK 3

```
<html>
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width,initial-scale=1.0">
</head>
<body>
    <script>
  function createCounter() {
  let count = 0;
  return {
    increment: function() {
      count++;
    },
    getCount: function() {
      return count;
    },
    reset: function() {
      count = 0;
    }
  };
}
const counter1 = createCounter();
const counter2 = createCounter();
console.log(counter1.getCount());
counter1.increment();
console.log(counter1.getCount());
console.log(counter2.getCount());
counter2.increment();
counter2.increment();
console.log(counter2.getCount());
```



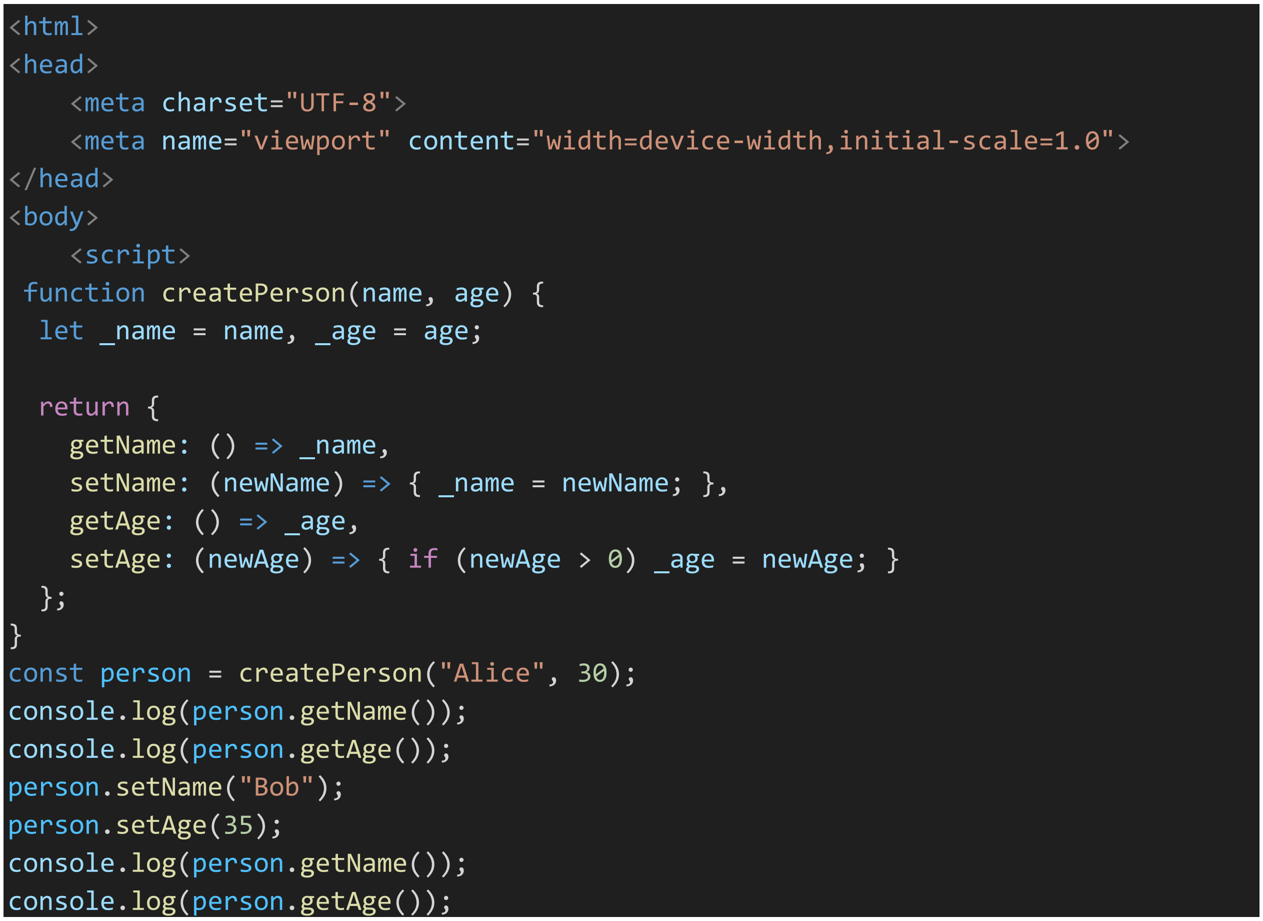
```
counter1.reset();
console.log(counter1.getCount());
</script>
</body>
</html>
```

OUTPUT:

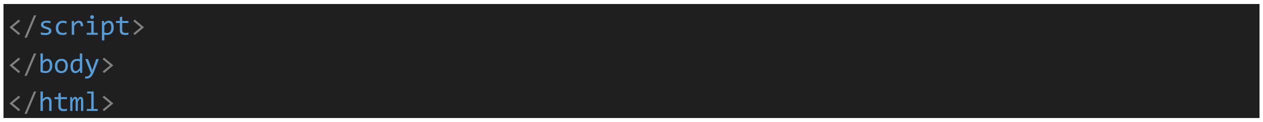


```
0                                                        task.htm:24
1                                                        task.htm:26
0                                                        task.htm:27
2                                                        task.htm:30
0                                                        task.htm:32
>
```

TASK 4

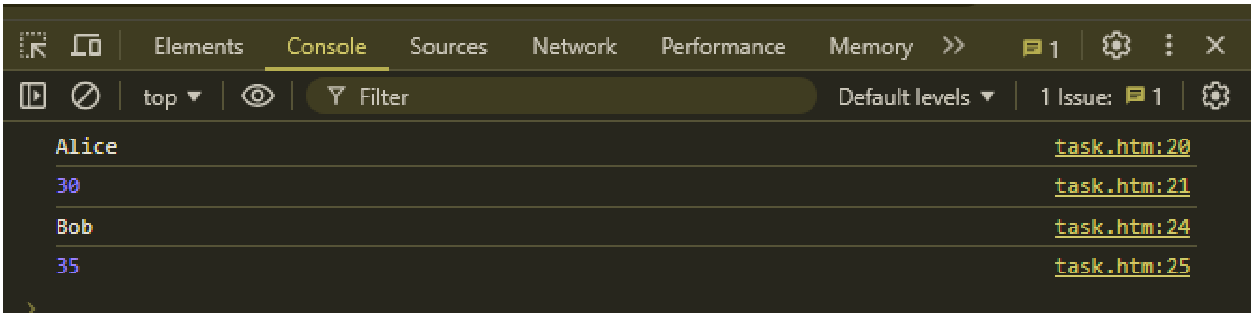


```
<html>
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width,initial-scale=1.0">
</head>
<body>
    <script>
 function createPerson(name, age) {
  let _name = name, _age = age;

  return {
    getName: () => _name,
    setName: (newName) => { _name = newName; },
    getAge: () => _age,
    setAge: (newAge) => { if (newAge > 0) _age = newAge; }
  };
}
const person = createPerson("Alice", 30);
console.log(person.getName());
console.log(person.getAge());
person.setName("Bob");
person.setAge(35);
console.log(person.getName());
console.log(person.getAge());
```



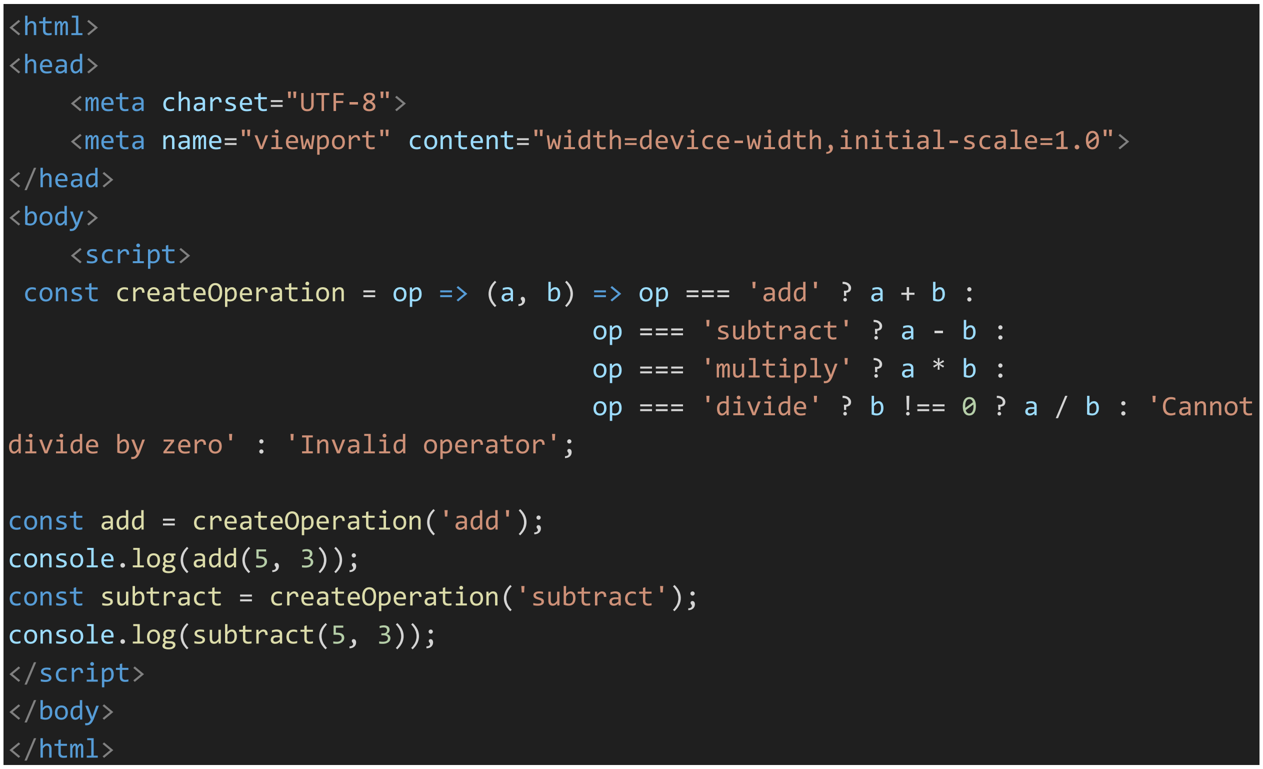
```
</script>
</body>
</html>
```

OUTPUT:



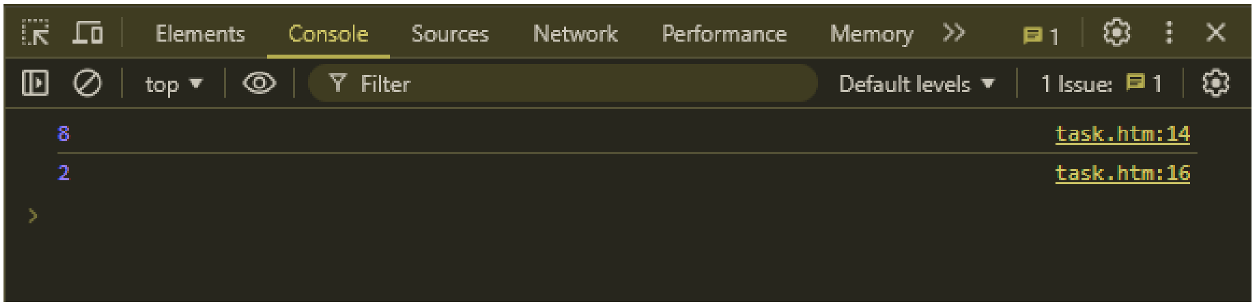
```
Alice                                                    task.htm:20
30                                                       task.htm:21
Bob                                                      task.htm:24
35                                                       task.htm:25
```

TASK 5



```
<html>
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width,initial-scale=1.0">
</head>
<body>
    <script>
 const createOperation = op => (a, b) => op === 'add' ? a + b :
                                     op === 'subtract' ? a - b :
                                     op === 'multiply' ? a * b :
                                     op === 'divide' ? b !== 0 ? a / b : 'Cannot
divide by zero' : 'Invalid operator';

const add = createOperation('add');
console.log(add(5, 3));
const subtract = createOperation('subtract');
console.log(subtract(5, 3));
</script>
</body>
</html>
```

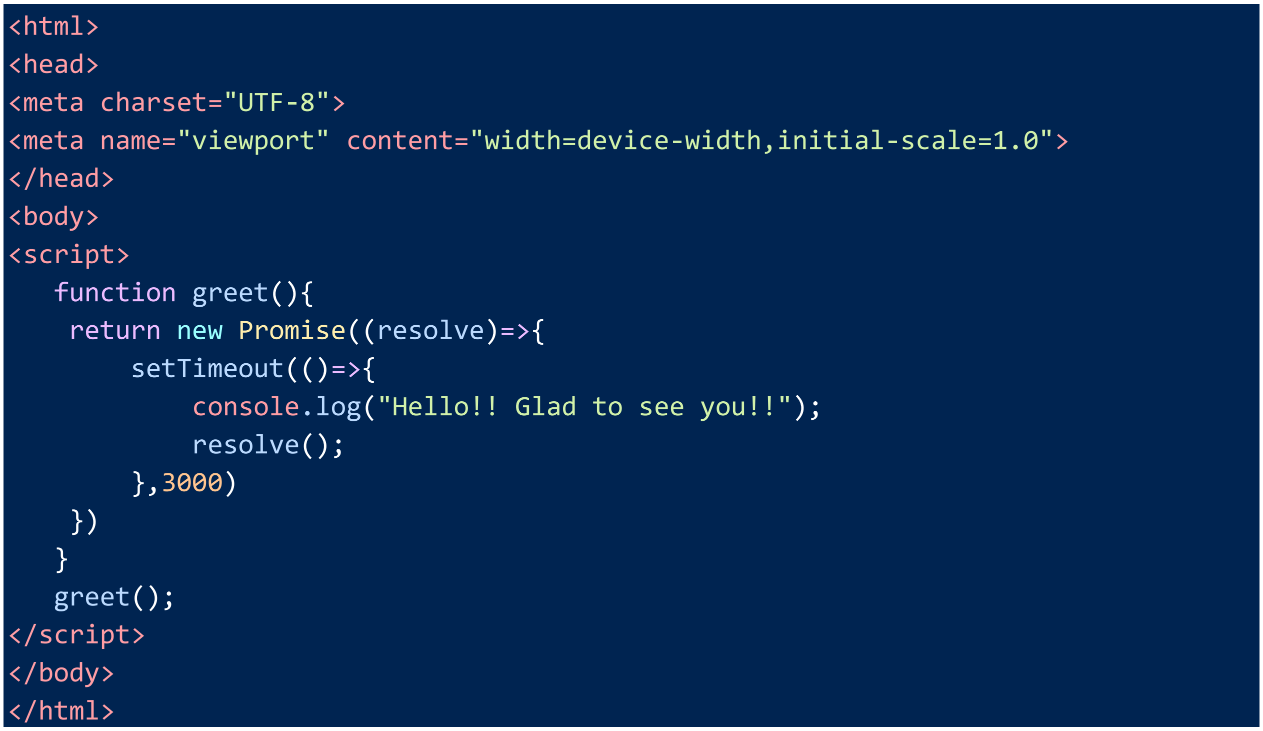
OUTPUT:



```
8                                                   task.htm:14
2                                                   task.htm:16
>
```

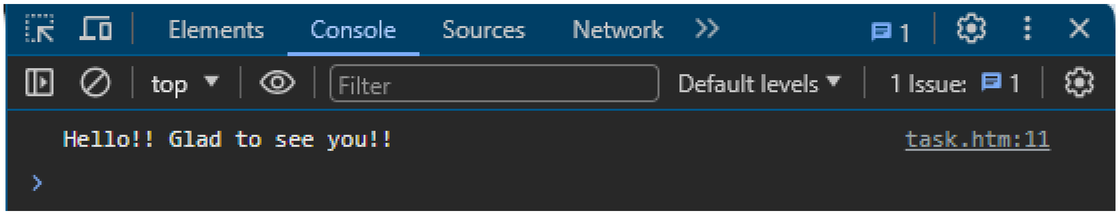
4. Promise, Promises chaining:

TASK 1



```
<html>
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width,initial-scale=1.0">
</head>
<body>
<script>
   function greet(){
    return new Promise((resolve)=>{
        setTimeout(()=>{
            console.log("Hello!! Glad to see you!!");
            resolve();
        },3000)
    })
   }
   greet();
</script>
</body>
</html>
```

OUTPUT:

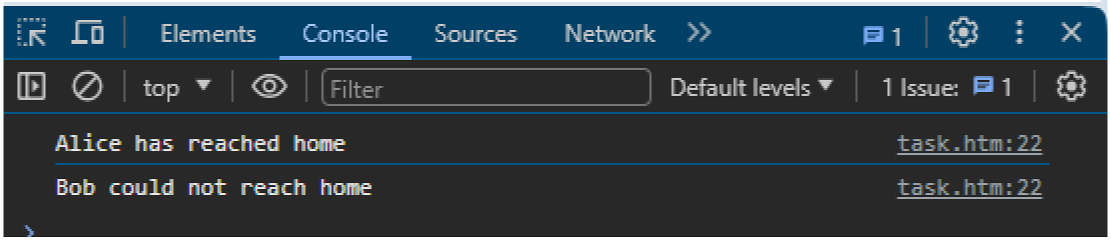


```
Hello!! Glad to see you!!                    task.htm:11
>
```

TASK 2

```
<html>
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width,initial-scale=1.0">
</head>
<body>
<script>
   function reachHome(name,timeToReach,isSuccessfull){
    return new Promise((resolve,reject)=>{
        setTimeout(()=>{
            if(isSuccessfull){
                resolve(`${name} has reached home`);
            }else{
                reject(`${name} could not reach home`);
            }
        })
    },timeToReach)
   }
   let friend1=reachHome('Alice',1000,true);
   let friend2=reachHome('Bob',2000,false);
   Promise.allSettled([friend1,friend2]).then((response)=>response.forEach((res)=>{
      console.log(res.status=="fulfilled"?res.value:res.reason);
   })
).catch((error)=>{console.log(error.message)})
.finally("Party Over");
</script>
</body>
</html>
```

OUTPUT:



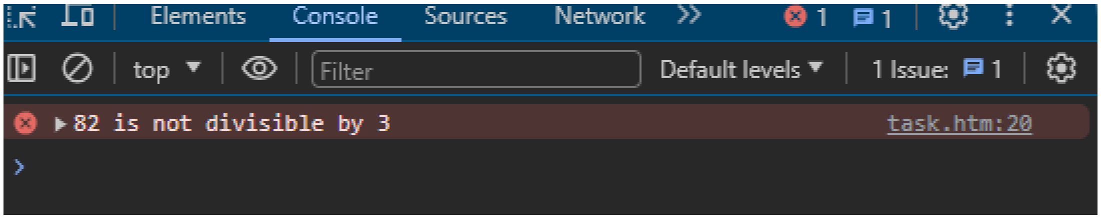
```
Alice has reached home                                        task.htm:22
Bob could not reach home                                      task.htm:22
```

TASK 3

```
<html>
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width,initial-scale=1.0">
```

```
</head>
<body>
<script>
 function randomPromise() {
      return new Promise((resolve, reject) => {
        const randomNumber = Math.floor(Math.random() * 100);
        if (randomNumber % 3 === 0) {
          resolve(`${randomNumber} is divisible by 3`);
        } else {
          reject(`${randomNumber} is not divisible by 3`);
        }
      });
    }
    randomPromise()
      .then(result => console.log(result))
      .catch(error => console.error(error));
</script>
</body>
</html>
```

OUTPUT:



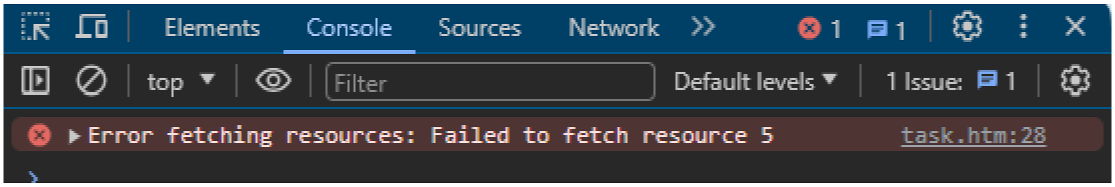
```
▶82 is not divisible by 3                          task.htm:20
```

TASK 4

```
<html>
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width,initial-scale=1.0">
</head>
<body>
<script>
 function fetchData(id) {
      return new Promise((resolve, reject) => {
        const delay = Math.floor(Math.random() * 2000) + 500;
        setTimeout(() => {
          if (Math.random() > 0.2) {
            resolve(`Resource ${id} fetched successfully in ${delay}ms`);
          } else {
```

```
      reject(`Failed to fetch resource ${id}`);
     }
    }, delay);
   });
  }
  const resourceIds = [1, 2, 3, 4, 5];
  const fetchPromises = resourceIds.map(id => fetchData(id));
  Promise.all(fetchPromises)
   .then(results => {
    console.log('All resources fetched successfully:');
    console.log(results);
   })
   .catch(error => {
    console.error('Error fetching resources:', error);
   });

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

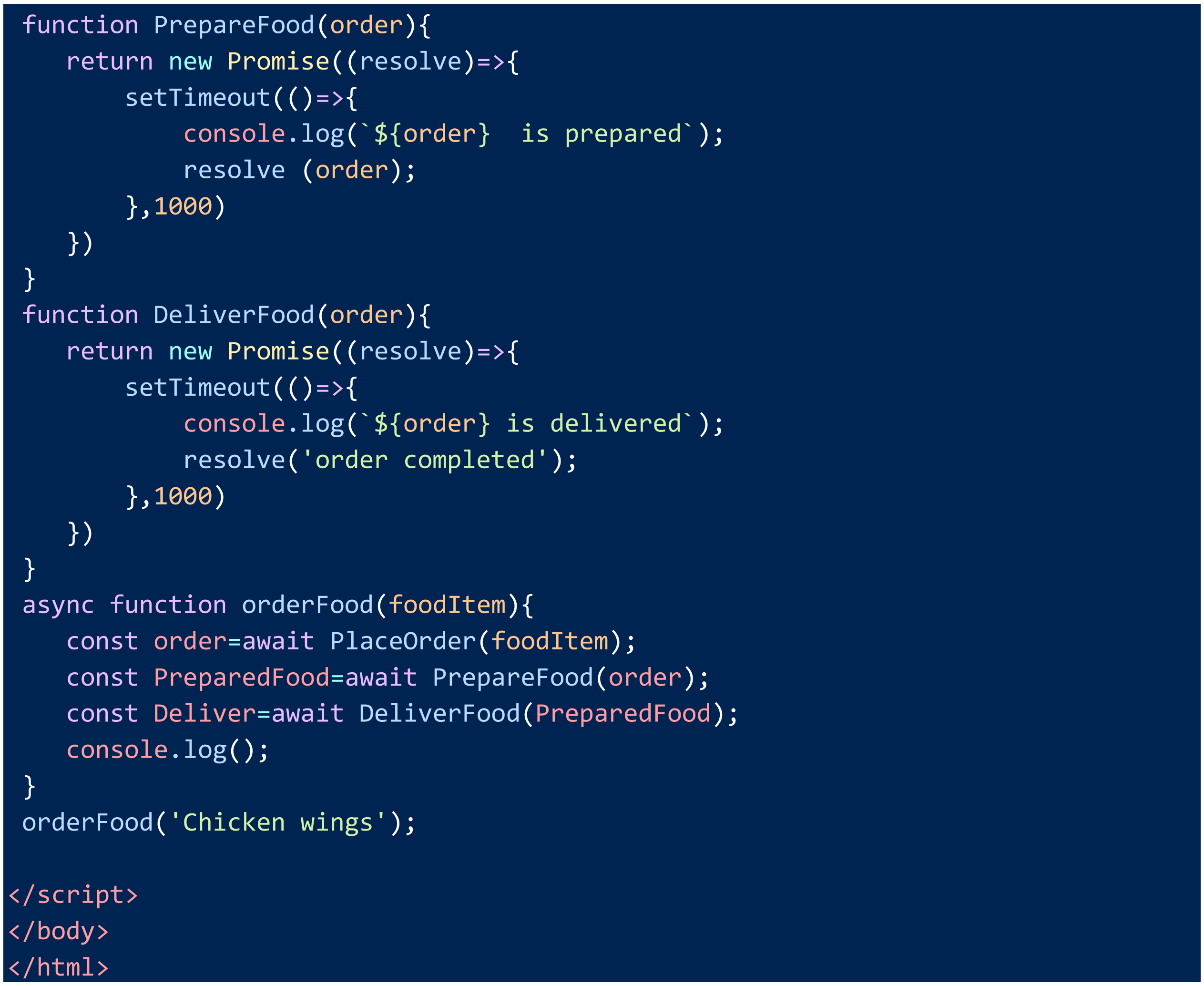
OUTPUT:



```
Error fetching resources: Failed to fetch resource 5    task.htm:28
```

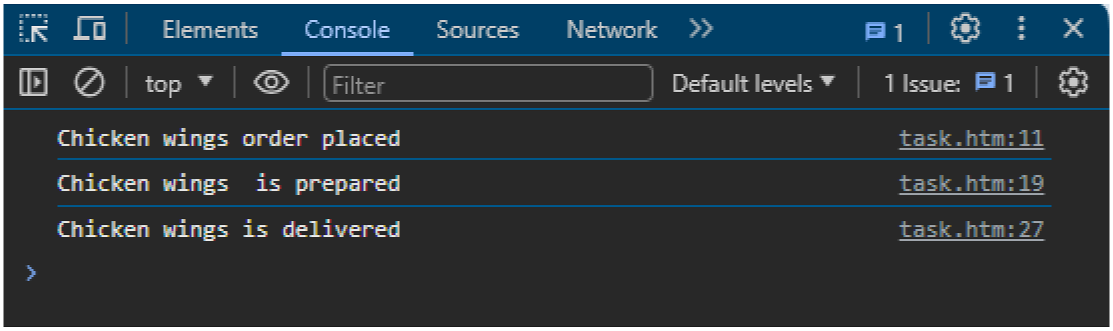
TASK 5

```
<html>
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width,initial-scale=1.0">
</head>
<body>
<script>
 function PlaceOrder(order){
  return new Promise((resolve)=>{
    setTimeout(()=>{
      console.log(`${order} order placed`);
      resolve(order);
    },1000)
  })
 }
```



```
function PrepareFood(order){
   return new Promise((resolve)=>{
       setTimeout(()=>{
           console.log(`${order}  is prepared`);
           resolve (order);
       },1000)
   })
}
function DeliverFood(order){
   return new Promise((resolve)=>{
       setTimeout(()=>{
           console.log(`${order} is delivered`);
           resolve('order completed');
       },1000)
   })
}
async function orderFood(foodItem){
   const order=await PlaceOrder(foodItem);
   const PreparedFood=await PrepareFood(order);
   const Deliver=await DeliverFood(PreparedFood);
   console.log();
}
orderFood('Chicken wings');

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

OUTPUT:



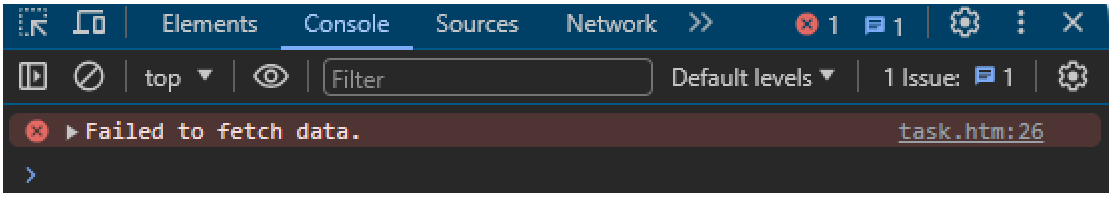
```
Chicken wings order placed                                    task.htm:11
Chicken wings  is prepared                                    task.htm:19
Chicken wings is delivered                                    task.htm:27
>
```

5. Async/await:

TASK 1

```
<html>
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width,initial-scale=1.0">
</head>
<body>
<script>
 async function fetchData() {
  const success = Math.random() > 0.5;
  return new Promise((resolve, reject) => {
    setTimeout(() => {
      if (success) {
        resolve("Data fetched successfully!");
      } else {
        reject("Failed to fetch data.");
      }
    }, 1000);
  });
}
async function main() {
  try {
    const data = await fetchData();
    console.log(data);
  } catch (error) {
    console.error(error);
  }
}
main();
</script>
</body>
</html>
```

OUTPUT:

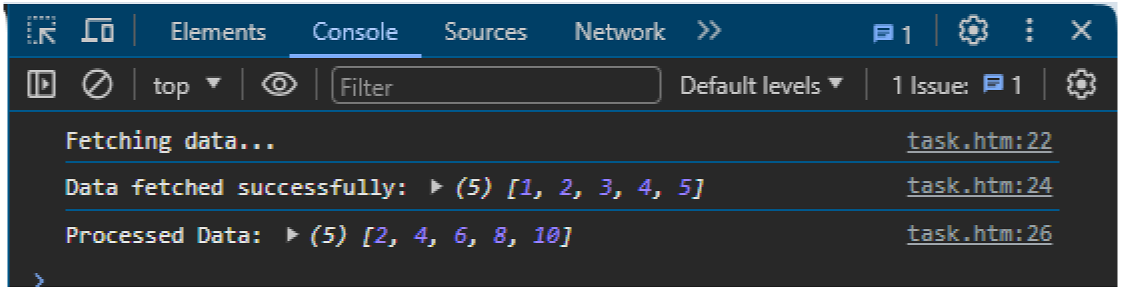


```
▶Failed to fetch data.                                    task.htm:26
```

TASK 2

```
<html>
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width,initial-scale=1.0">
</head>
<body>
<script>
 async function fetchData() {
      return new Promise((resolve, reject) => {
        const delay = Math.floor(Math.random() * 2000) + 500;
        setTimeout(() => {
          if (Math.random() > 0.2) {
            resolve({ data: [1, 2, 3, 4, 5] });
          } else {
            reject("Failed to fetch data");
          }
        }, delay);
      });
    }
    async function fetchAndProcessData() {
      try {
        console.log("Fetching data...");
        const response = await fetchData();
        console.log("Data fetched successfully:", response.data);
        const processedData = response.data.map(num => num * 2);
        console.log("Processed Data:", processedData);

        return processedData;
      } catch (error) {
        console.error("Error:", error);
      }
    }
    fetchAndProcessData();

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

OUTPUT:

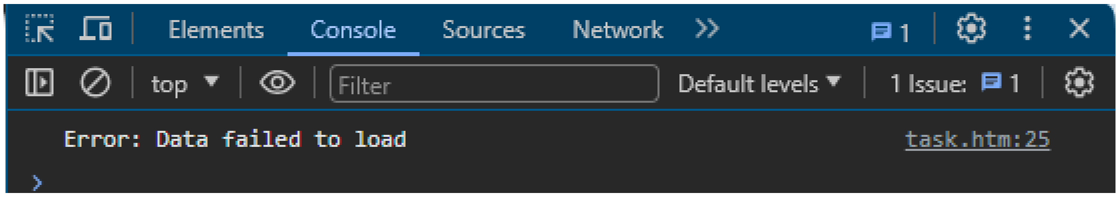


```
Fetching data...                                      task.htm:22
Data fetched successfully: ▶ (5) [1, 2, 3, 4, 5]      task.htm:24
Processed Data: ▶ (5) [2, 4, 6, 8, 10]                task.htm:26
>
```

TASK 3

```
<html>
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width,initial-scale=1.0">
</head>
<body>
<script>
   async function getData() {
      return new Promise((resolve, reject) => {
        let success = Math.random() > 0.5;
        setTimeout(() => {
          if (success) {
            resolve("Data loaded successfully");
          } else {
            reject("Data failed to load");
          }
        }, 1000);
      });
    }
    async function processTask() {
      try {
        let result = await getData();
        console.log(result);
      } catch (err) {
        console.log("Error:", err);
      }
    }
    processTask();
</script>
</body>
</html>
```

OUTPUT:

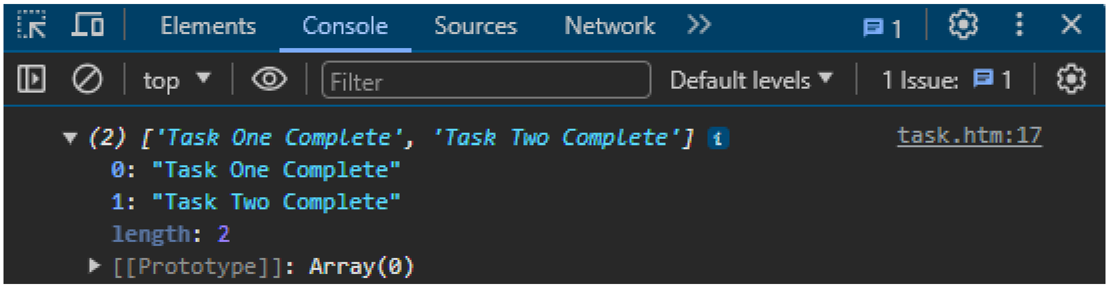


```
Error: Data failed to load                                    task.htm:25
```

TASK 4

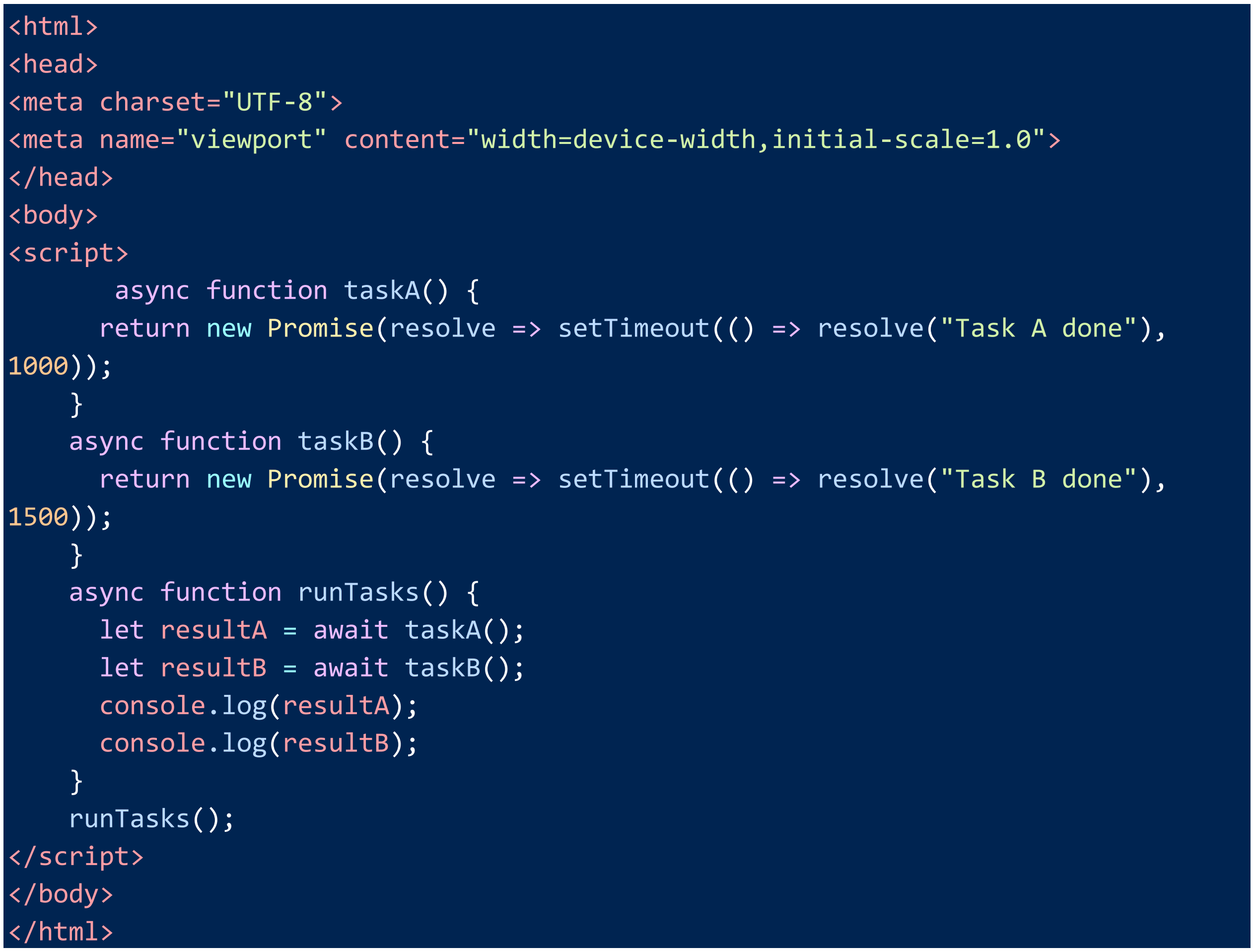
```
<html>
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width,initial-scale=1.0">
</head>
<body>
<script>
    async function taskOne() {
      return new Promise(resolve => setTimeout(() => resolve("Task One
Complete"), 1000));
    }
    async function taskTwo() {
      return new Promise(resolve => setTimeout(() => resolve("Task Two
Complete"), 1500));
    }
    async function runTasks() {
      try {
        let results = await Promise.all([taskOne(), taskTwo()]);
        console.log(results);
      } catch (err) {
        console.log("Error:", err);
      }
    }
    runTasks();
</script>
</body>
</html>
```

OUTPUT:



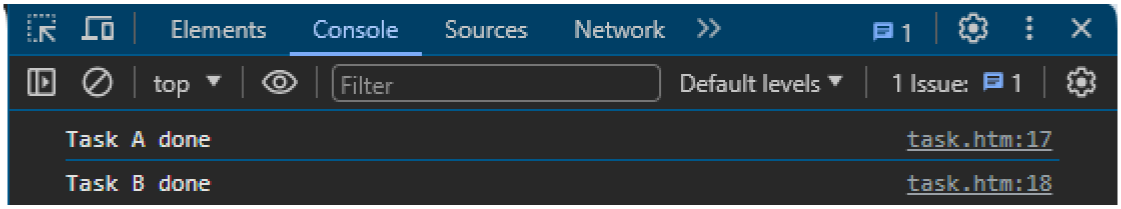
```
▼ (2) ['Task One Complete', 'Task Two Complete']                task.htm:17
    0: "Task One Complete"
    1: "Task Two Complete"
    length: 2
  ▶ [[Prototype]]: Array(0)
```

TASK 5



```
<html>
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width,initial-scale=1.0">
</head>
<body>
<script>
       async function taskA() {
      return new Promise(resolve => setTimeout(() => resolve("Task A done"),
1000));
    }
    async function taskB() {
      return new Promise(resolve => setTimeout(() => resolve("Task B done"),
1500));
    }
    async function runTasks() {
      let resultA = await taskA();
      let resultB = await taskB();
      console.log(resultA);
      console.log(resultB);
    }
    runTasks();
</script>
</body>
</html>
```

OUTPUT:



```
Task A done                                     task.htm:17
Task B done                                     task.htm:18
```

6.Modules introduction, Export and Import

TASK 1

module.js

```
export function greet(name) {
    return `Hello, ${name}!`;
}
export class Person {
    constructor(name, age) {
        this.name = name;
        this.age = age;
    }

    describe() {
        return `${this.name} is ${this.age} years old.`;
    }
}
export const currentYear = new Date().getFullYear();
```

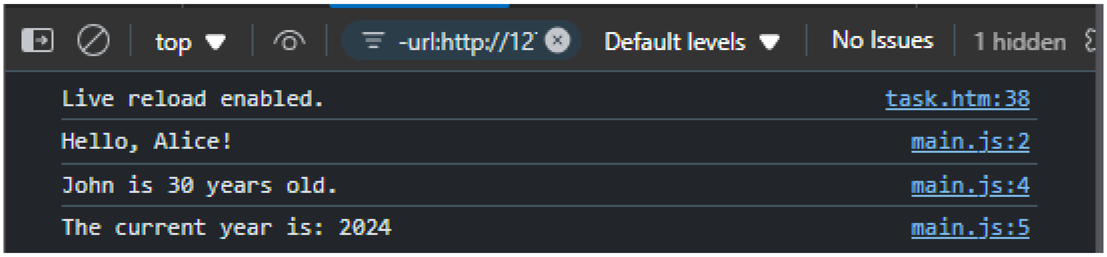
main.js

```
import { greet, Person, currentYear } from './module.js';
console.log(greet("Alice"));
const person1 = new Person("John", 30);
console.log(person1.describe());
console.log(`The current year is: ${currentYear}`);
```

task.htm

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

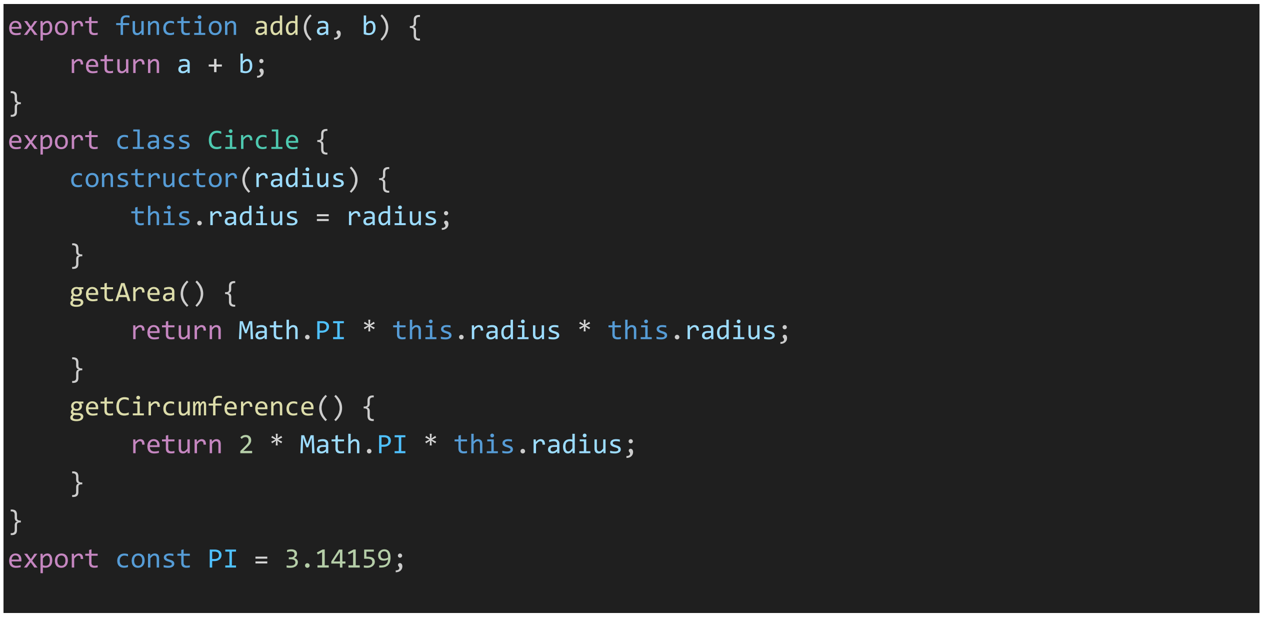
OUTPUT:



```
Live reload enabled.                                          task.htm:38
Hello, Alice!                                                  main.js:2
John is 30 years old.                                          main.js:4
The current year is: 2024                                      main.js:5
```

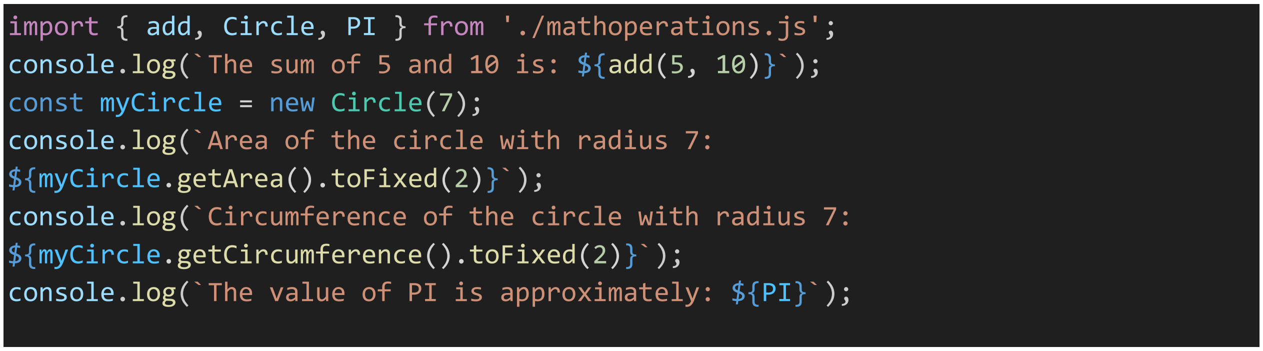
TASK 2

mathoperations.js



```
export function add(a, b) {
    return a + b;
}
export class Circle {
    constructor(radius) {
        this.radius = radius;
    }
    getArea() {
        return Math.PI * this.radius * this.radius;
    }
    getCircumference() {
        return 2 * Math.PI * this.radius;
    }
}
export const PI = 3.14159;
```

main.js

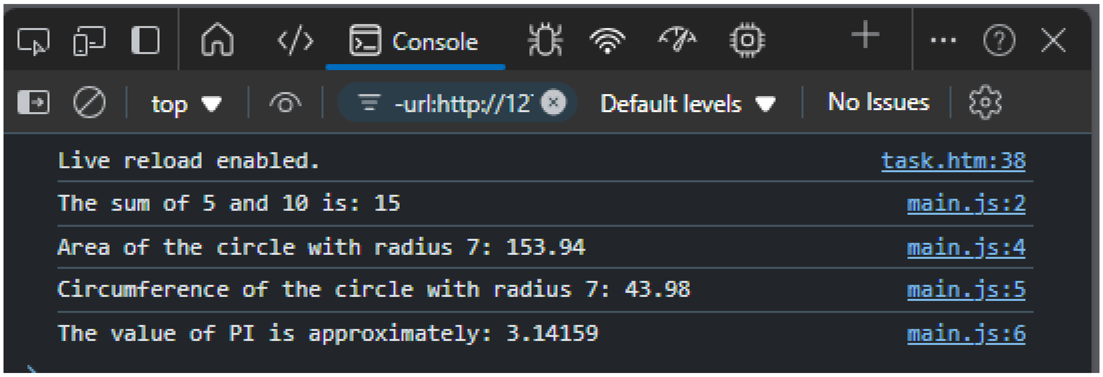


```
import { add, Circle, PI } from './mathoperations.js';
console.log(`The sum of 5 and 10 is: ${add(5, 10)}`);
const myCircle = new Circle(7);
console.log(`Area of the circle with radius 7:
${myCircle.getArea().toFixed(2)}`);
console.log(`Circumference of the circle with radius 7:
${myCircle.getCircumference().toFixed(2)}`);
console.log(`The value of PI is approximately: ${PI}`);
```

task.htm

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

OUTPUT:



```
Live reload enabled.                                  task.htm:38
The sum of 5 and 10 is: 15                              main.js:2
Area of the circle with radius 7: 153.94                main.js:4
Circumference of the circle with radius 7: 43.98        main.js:5
The value of PI is approximately: 3.14159               main.js:6
```

TASK 3

mathUtils.js

```javascript
export function add(a, b) {
    return a + b;
}
export function subtract(a, b) {
    return a - b;
}
export function multiply(a, b) {
    return a * b;
}
export function divide(a, b) {
    if (b === 0) {
        throw new Error("Division by zero is not allowed.");
    }
    return a / b;
}
```

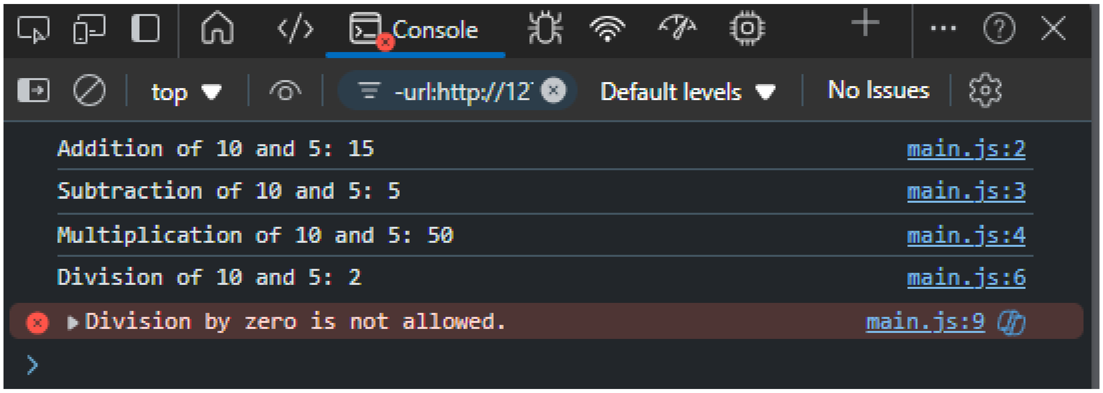
main.js

```
import { add, subtract, multiply, divide } from './mathUtils.js';
console.log(`Addition of 10 and 5: ${add(10, 5)}`);
console.log(`Subtraction of 10 and 5: ${subtract(10, 5)}`);
console.log(`Multiplication of 10 and 5: ${multiply(10, 5)}`);
try {
    console.log(`Division of 10 and 5: ${divide(10, 5)}`);
    console.log(`Division of 10 and 0: ${divide(10, 0)}`);
} catch (error) {
    console.error(error.message);
}
```

task.htm

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

OUTPUT:



```
Addition of 10 and 5: 15                          main.js:2
Subtraction of 10 and 5: 5                        main.js:3
Multiplication of 10 and 5: 50                    main.js:4
Division of 10 and 5: 2                           main.js:6
⊗ ▸Division by zero is not allowed.               main.js:9
>
```

TASK 4

stringUtils.js

```
export function toUpperCase(str) {
    return str.toUpperCase();
}
export function toLowerCase(str) {
    return str.toLowerCase();
}
export function reverseString(str) {
    return str.split('').reverse().join('');
}
export function getLength(str) {
    return str.length;
}
```

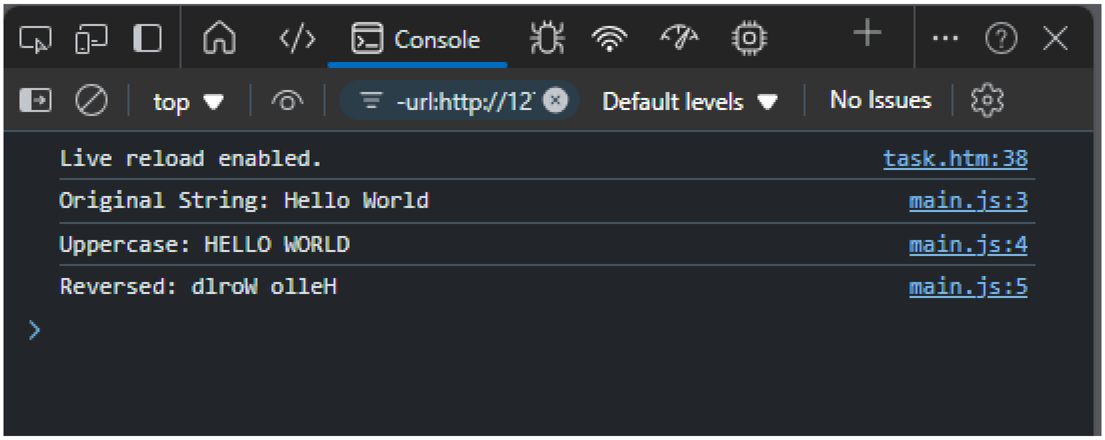
main.js

```
import { toUpperCase, reverseString } from './stringUtils.js';
const originalString = "Hello World";
console.log(`Original String: ${originalString}`);
console.log(`Uppercase: ${toUpperCase(originalString)}`);
console.log(`Reversed: ${reverseString(originalString)}`);
```

task.htm

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

OUTPUT:



```
Live reload enabled.
Original String: Hello World
Uppercase: HELLO WORLD
Reversed: dlroW olleH
```

TASK 5

mathUtils.js

```
export default function calculateSquare(number) {
    return number * number;
}
export function calculateCube(number) {
    return number * number * number;
}
export function calculateSquareRoot(number) {
    return Math.sqrt(number);
}
```

main.js

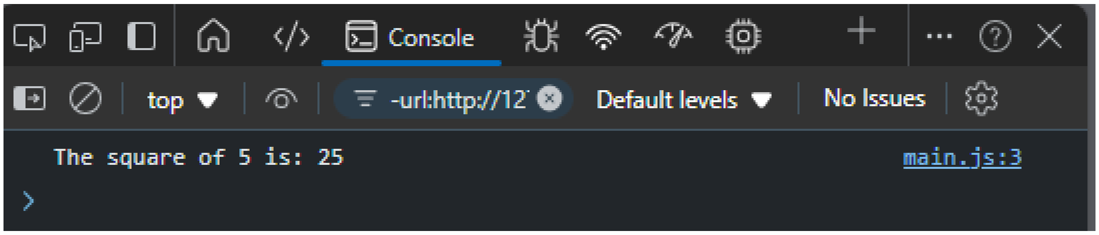
```
import calculateSquare from './mathUtils.js';
const number = 5;
console.log(`The square of ${number} is: ${calculateSquare(number)}`);
```

task.htm

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Default Export Example</title>
```

```
</head>
<body>
    <script type="module" src="main.js"></script>
</body>
</html>
```

OUTPUT:



```
The square of 5 is: 25                                   main.js:3
>
```

TASK 1

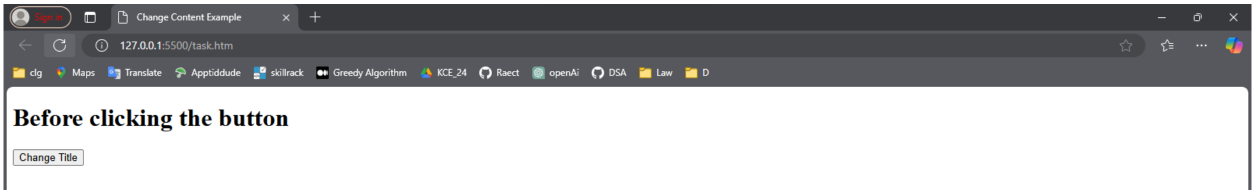
Task.htm

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Change Content Example</title>
</head>
<body>
    <h1 id="title">Before clicking the button</h1>
    <button id="changeButton">Change Title</button>
    <script src="main.js"></script>
</body>
</html>
```

Main.js

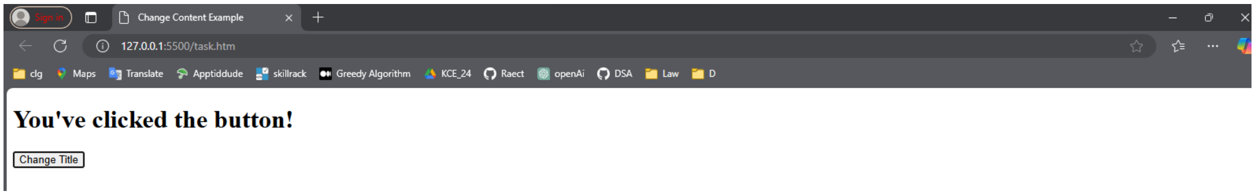
```
const titleElement = document.getElementById("title");
const button = document.getElementById("changeButton");
button.addEventListener("click", () => {
    titleElement.textContent = "You've clicked the button!";
});
```

OUTPUT:



Before clicking the button

Change Title



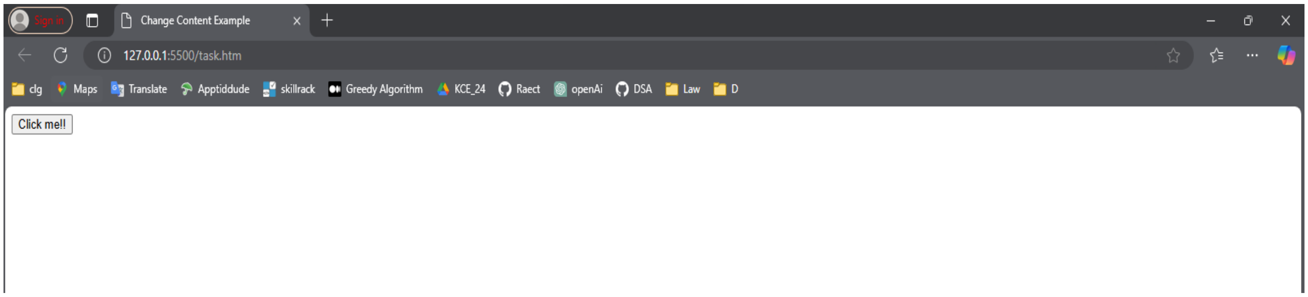
You've clicked the button!

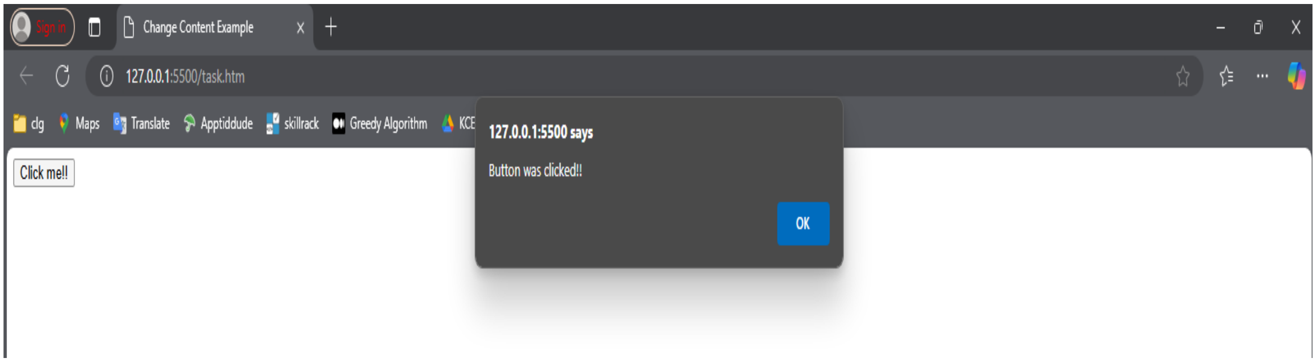
Change Title

TASK 2

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Change Content Example</title>
</head>
<body>
    <button id="actionButton">Click me!!</button>
    <script>
        const button =document.getElementById("actionButton");
        button.addEventListener("click",function(){
            alert("Button was clicked!!");
        });
    </script>
</body>
</html>
```

OUTPUT:

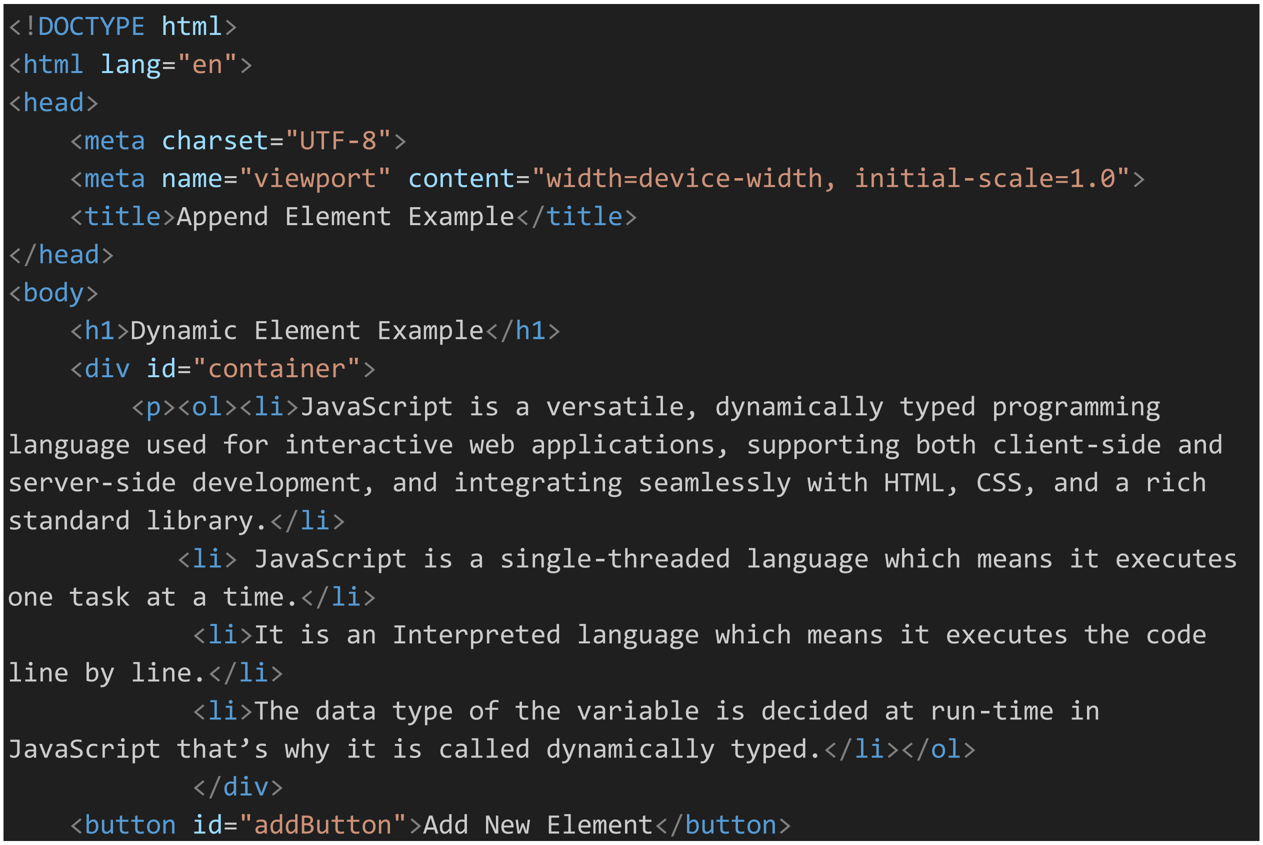




127.0.0.1:5500 says

Button was clicked!!

TASK 3

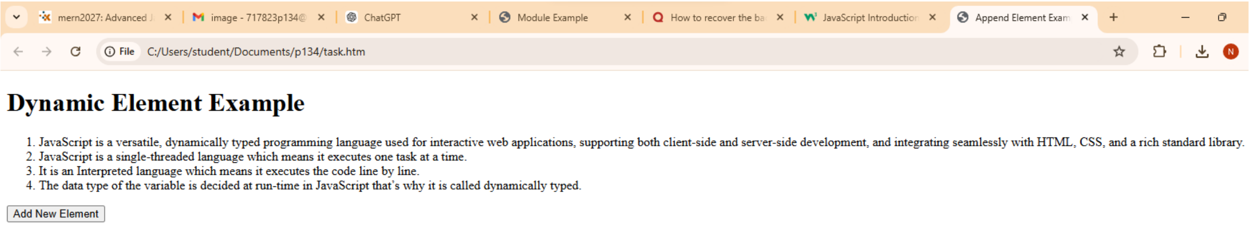


```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Append Element Example</title>
</head>
<body>
    <h1>Dynamic Element Example</h1>
    <div id="container">
        <p><ol><li>JavaScript is a versatile, dynamically typed programming
language used for interactive web applications, supporting both client-side and
server-side development, and integrating seamlessly with HTML, CSS, and a rich
standard library.</li>
           <li> JavaScript is a single-threaded language which means it executes
one task at a time.</li>
            <li>It is an Interpreted language which means it executes the code
line by line.</li>
            <li>The data type of the variable is decided at run-time in
JavaScript that’s why it is called dynamically typed.</li></ol>
            </div>
    <button id="addButton">Add New Element</button>
    <script src="main.js"></script>
</body>
</html>
```

Main.js

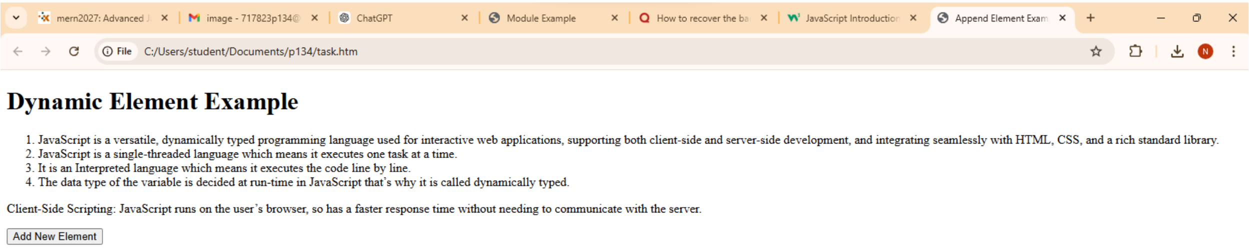
```
const container = document.getElementById("container");
const button = document.getElementById("addButton");
button.addEventListener("click", () => {
    const newParagraph = document.createElement("p");
    newParagraph.textContent = "Client-Side Scripting: JavaScript runs on the
user’s browser, so has a faster response time without needing to communicate with
the server."
    container.appendChild(newParagraph);
});
```

OUTPUT:



Dynamic Element Example

1. JavaScript is a versatile, dynamically typed programming language used for interactive web applications, supporting both client-side and server-side development, and integrating seamlessly with HTML, CSS, and a rich standard library.
2. JavaScript is a single-threaded language which means it executes one task at a time.
3. It is an Interpreted language which means it executes the code line by line.
4. The data type of the variable is decided at run-time in JavaScript that’s why it is called dynamically typed.



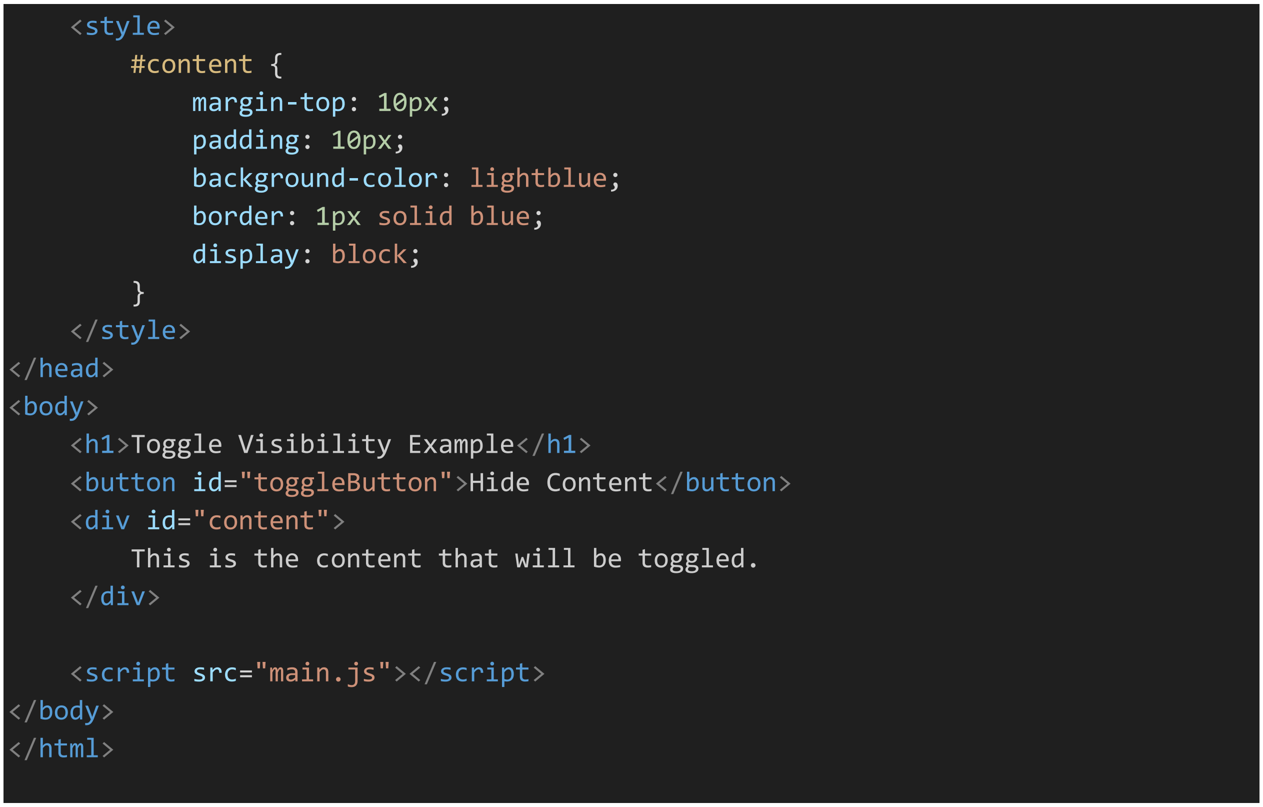
Dynamic Element Example

1. JavaScript is a versatile, dynamically typed programming language used for interactive web applications, supporting both client-side and server-side development, and integrating seamlessly with HTML, CSS, and a rich standard library.
2. JavaScript is a single-threaded language which means it executes one task at a time.
3. It is an Interpreted language which means it executes the code line by line.
4. The data type of the variable is decided at run-time in JavaScript that’s why it is called dynamically typed.

Client-Side Scripting: JavaScript runs on the user’s browser, so has a faster response time without needing to communicate with the server.

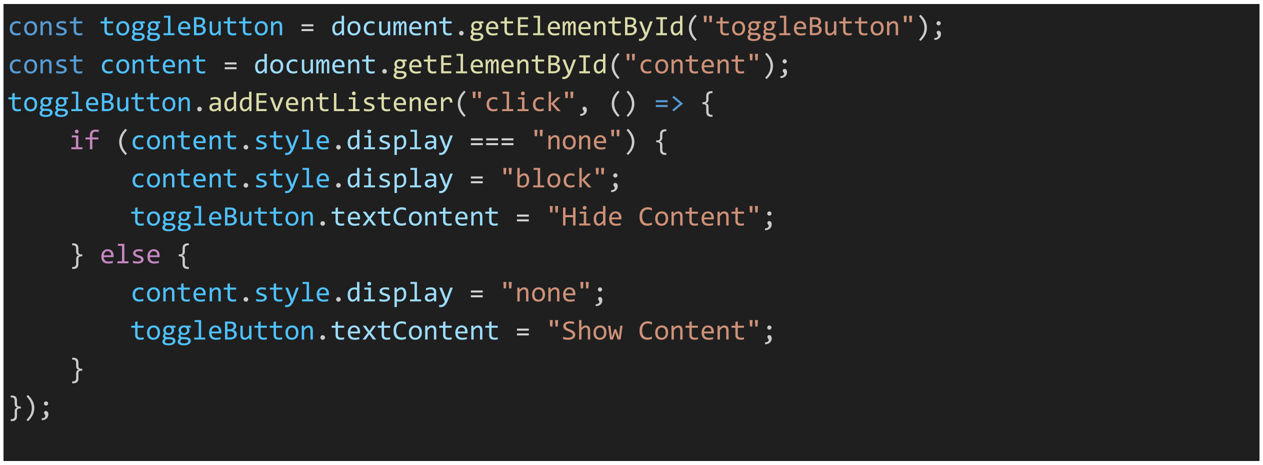
TASK 4

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Toggle Visibility Example</title>
```



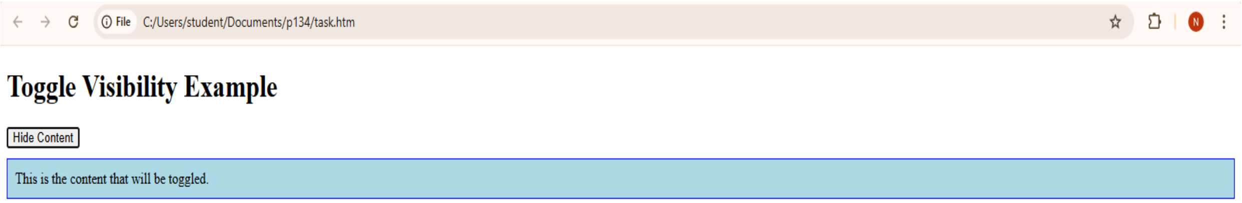
```
    <style>
        #content {
            margin-top: 10px;
            padding: 10px;
            background-color: lightblue;
            border: 1px solid blue;
            display: block;
        }
    </style>
</head>
<body>
    <h1>Toggle Visibility Example</h1>
    <button id="toggleButton">Hide Content</button>
    <div id="content">
        This is the content that will be toggled.
    </div>

    <script src="main.js"></script>
</body>
</html>
```

Main.js



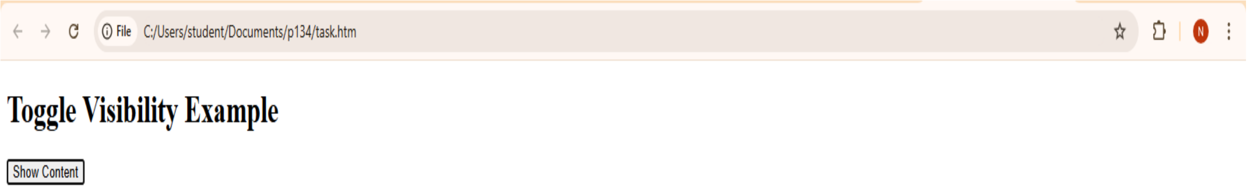
```
const toggleButton = document.getElementById("toggleButton");
const content = document.getElementById("content");
toggleButton.addEventListener("click", () => {
    if (content.style.display === "none") {
        content.style.display = "block";
        toggleButton.textContent = "Hide Content";
    } else {
        content.style.display = "none";
        toggleButton.textContent = "Show Content";
    }
});
```

OUTPUT:



Toggle Visibility Example

This is the content that will be toggled.



Toggle Visibility Example

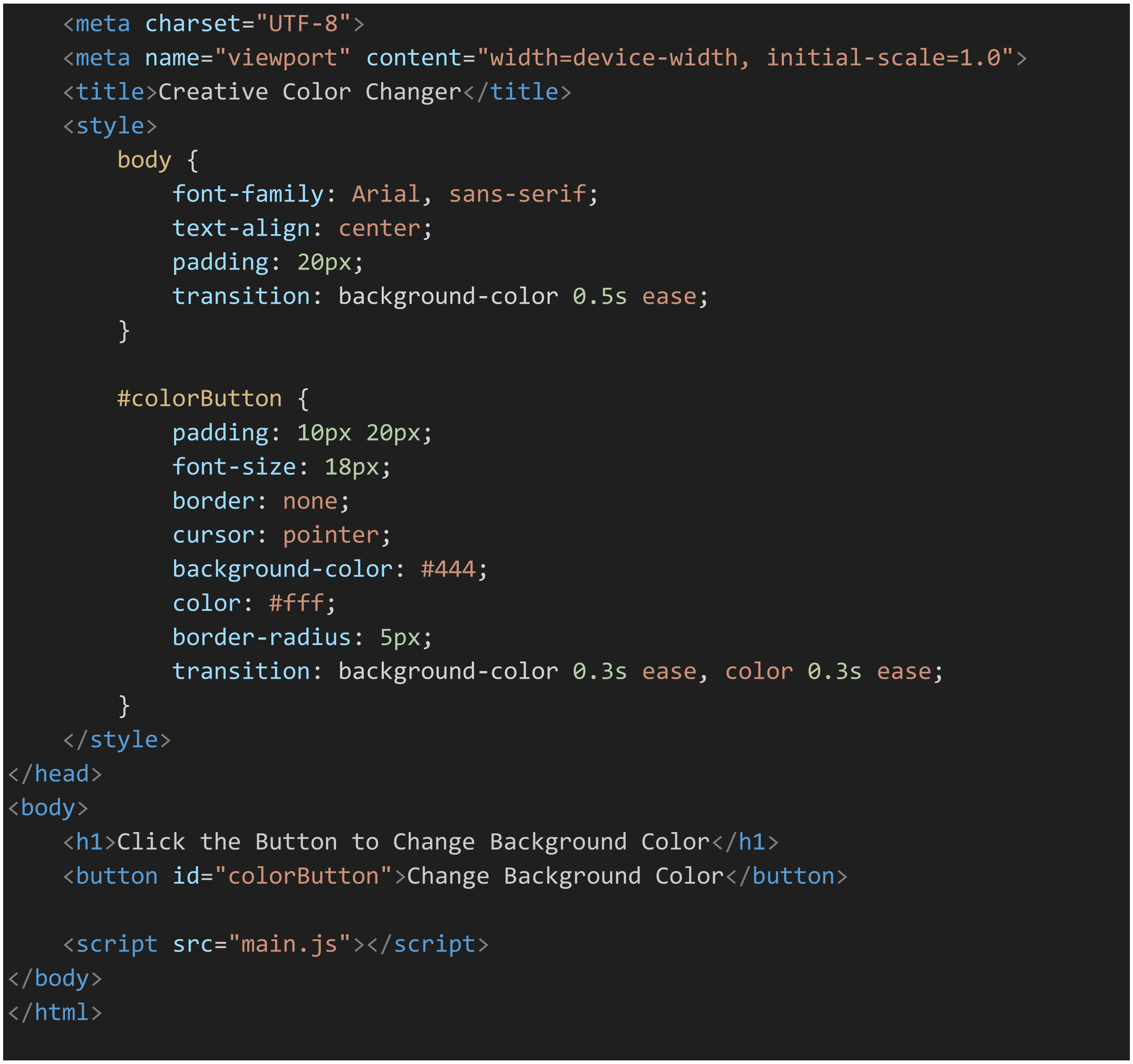
TASK 5

Main.js

```
const colorButton = document.getElementById("colorButton");
function getRandomColor() {
    const letters = "0123456789ABCDEF";
    let color = "#";
    for (let i = 0; i < 6; i++) {
        color += letters[Math.floor(Math.random() * 16)];
    }
    return color;
}
colorButton.addEventListener("click", () => {
    const randomColor = getRandomColor();
    document.body.style.backgroundColor = randomColor;
    colorButton.textContent = `Background: ${randomColor}`;
    colorButton.style.backgroundColor = randomColor;
    colorButton.style.color = getRandomColor();
});
```

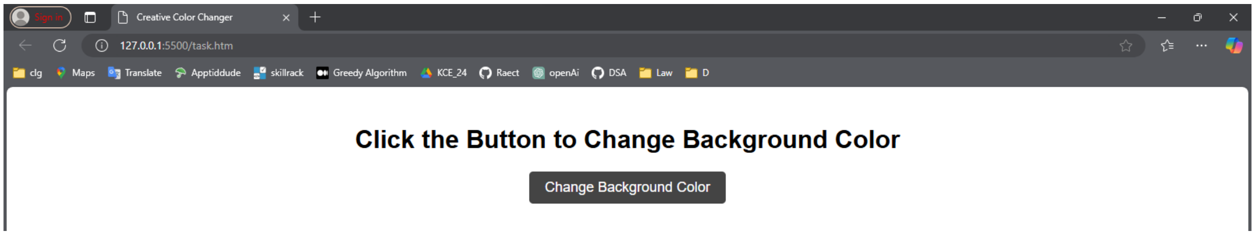
Task.htm

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

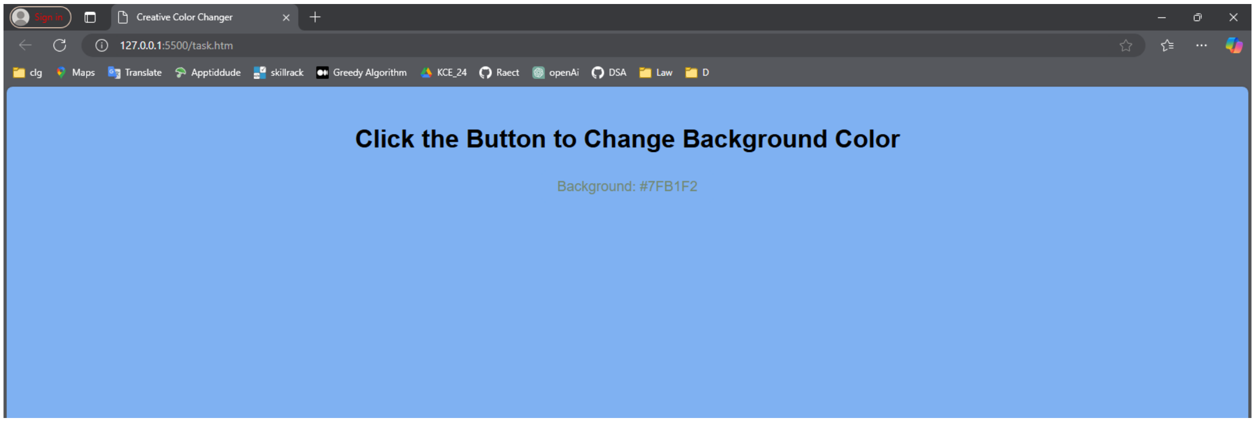


```
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Creative Color Changer</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            text-align: center;
            padding: 20px;
            transition: background-color 0.5s ease;
        }

        #colorButton {
            padding: 10px 20px;
            font-size: 18px;
            border: none;
            cursor: pointer;
            background-color: #444;
            color: #fff;
            border-radius: 5px;
            transition: background-color 0.3s ease, color 0.3s ease;
        }
    </style>
</head>
<body>
    <h1>Click the Button to Change Background Color</h1>
    <button id="colorButton">Change Background Color</button>

    <script src="main.js"></script>
</body>
</html>
```

OUTPUT:

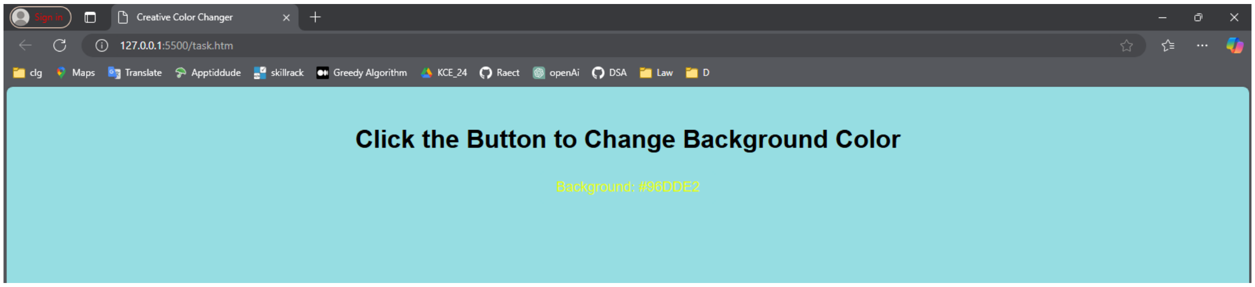


Click the Button to Change Background Color



Click the Button to Change Background Color

Background: #7FB1F2



Click the Button to Change Background Color

Background: #96DDE2