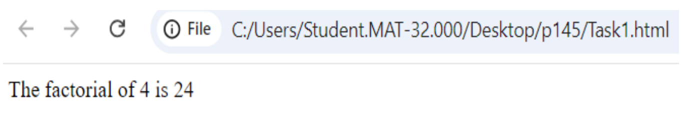
Recursion and stack:

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 factorial(n){
         if(n==0 || n==1)
         return 1
         else
         return n*factorial(n-1)
      }
      let n=4
      document.writeln("The factorial of "+n+ " is" +factorial(n));
   </script>
</body>
</html>
```

Output:



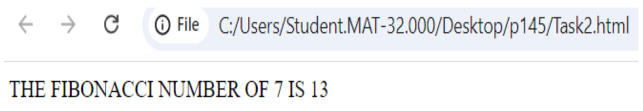
```
The factorial of 4 is 24
```

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 fibonacci(num){
         if(num==0)
```

```
            return 0
            else if(num==1)
            return 1
            else
            return fibonacci(num-1)+fibonacci(num-2)

        }

        let num=7
        document.writeln("THE FIBONACCI NUMBER OF "+num+" IS ",fibonacci(num))
    </script>
</body>
</html>
```

Output:



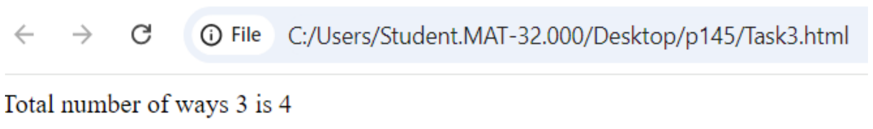
File C:/Users/Student.MAT-32.000/Desktop/p145/Task2.html

```
THE FIBONACCI NUMBER OF 7 IS 13
```

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 CountWays(num){
            if(num==1)
            return 1
            else if(num==2)
            return 2
            else if(num==3)
            return 4
            else
            return CountWays(n-1)+CountWays(n-2)+CountWays(n-3)

        }
        let num=3
        document.writeln("Total number of ways "+num+" is ",CountWays(num))
    </script>
</body>
</html>
```

Output:



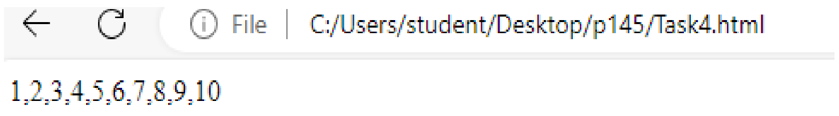
C:/Users/Student.MAT-32.000/Desktop/p145/Task3.html

```
Total number of ways 3 is 4
```

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 flatten(arr){
            let result=[];
            arr.forEach((Element)=>{
            if(Array.isArray(Element)){
            result=result.concat(flatten(Element))
            }else{
            result.push(Element)
            }})
        return result;
        }
        let nestedArray=[1,[2,3],4,[5,[6,7]],8,[9,10]]
        let flattenArray=flatten(nestedArray)
        document.writeln(flattenArray)
    </script>
</body>
</html>
```

Output:

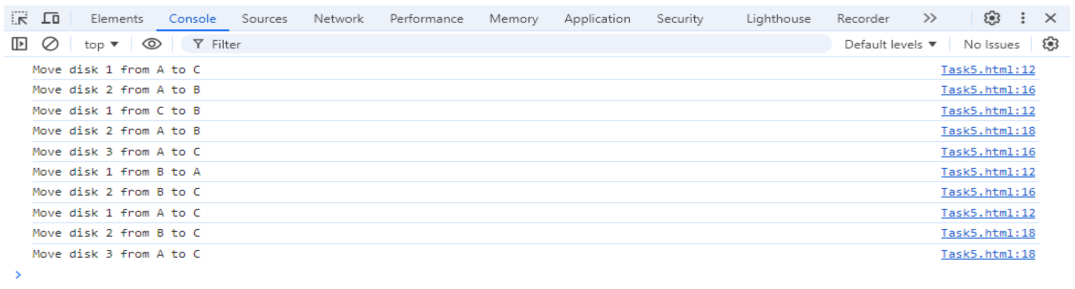


```
1,2,3,4,5,6,7,8,9,10
```

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,auxilary,target,){
      if(n==1){
         console.log(`Move disk 1 from ${source} to ${target}`)
         return;
      }
         TowerOfHanoi(n-1,source,target,auxilary)
         console.log(`Move disk ${n} from ${source} to ${target}`)
         TowerOfHanoi(n-1,auxilary,source,target)
         console.log(`Move disk ${n} from ${source} to ${target}`)
   }
      const disk=3;
      TowerOfHanoi(disk,'A','B','C')

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

Output:



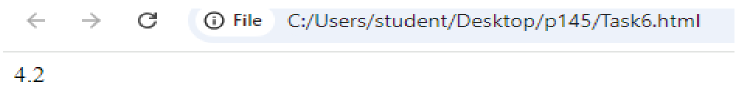
```
Move disk 1 from A to C     Task5.html:12
Move disk 2 from A to B     Task5.html:16
Move disk 1 from C to B     Task5.html:12
Move disk 2 from A to B     Task5.html:18
Move disk 3 from A to C     Task5.html:16
Move disk 1 from B to A     Task5.html:12
Move disk 2 from B to C     Task5.html:16
Move disk 1 from A to C     Task5.html:12
Move disk 2 from B to C     Task5.html:18
Move disk 3 from A to C     Task5.html:18
```

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 arbitarynum(num1,num2){
            return num1+num2
        }
        let result=arbitarynum(8/2,0.2)
        document.writeln(result)

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

Output:

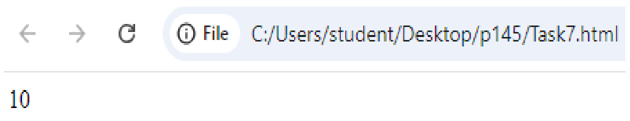

C:/Users/student/Desktop/p145/Task6.html

```
4.2
```

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 sumNumber(...Numbers){
            return Numbers.reduce((Total,Numbers)=>Total+Numbers,0)
        }
        const arr=[1,4,5]
        document.writeln(sumNumber(...arr))
    </script>
</body>
</html>
```

Output:

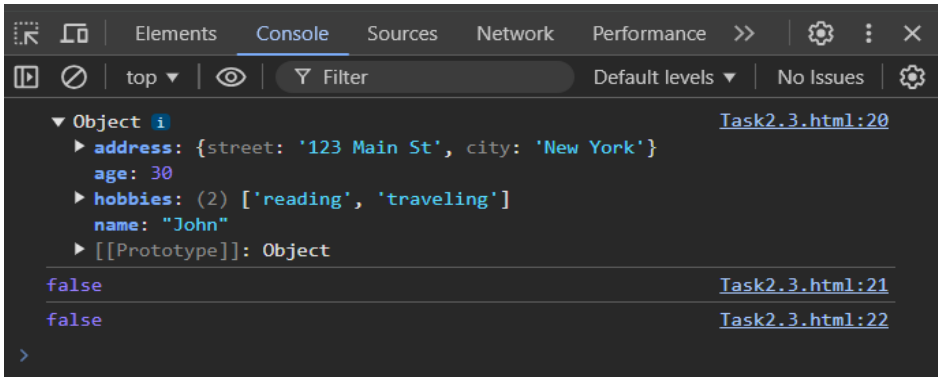


```
C:/Users/student/Desktop/p145/Task7.html

10
```

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>
        const originalObject = {
  name: 'John',
  age: 30,
  address: {
    street: '123 Main St',
    city: 'New York'
  },
  hobbies: ['reading', 'traveling']
};
const clonedObject = JSON.parse(JSON.stringify(originalObject));
console.log(clonedObject);
console.log(clonedObject === originalObject);
console.log(clonedObject.address === originalObject.address);
    </script>
</body>
</html>
```

Output:

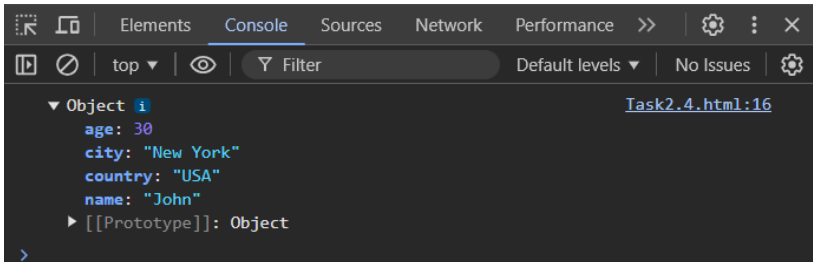


```
▼ Object i                                                        Task2.3.html:20
  ▶ address: {street: '123 Main St', city: 'New York'}
    age: 30
  ▶ hobbies: (2) ['reading', 'traveling']
    name: "John"
  ▶ [[Prototype]]: Object
false                                                             Task2.3.html:21
false                                                             Task2.3.html:22
>
```

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 mergeObjects(obj1, obj2) {
  return { ...obj1, ...obj2 };
}
const object1 = { name: 'John', age: 30 };
const object2 = { city: 'New York', country: 'USA' };
const mergedObject = mergeObjects(object1, object2);
console.log(mergedObject);
    </script>
</body>
</html>
```

Output:

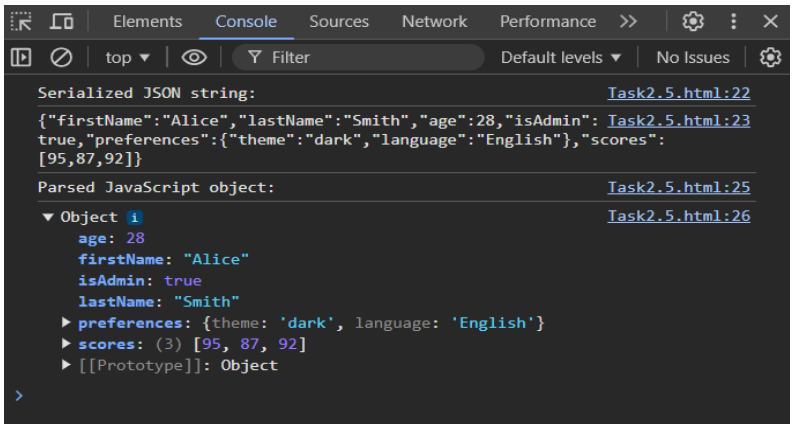


```
▼ Object
    age: 30
    city: "New York"
    country: "USA"
    name: "John"
  ▶ [[Prototype]]: Object
```

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>
const person = {
  firstName: 'Alice',
  lastName: 'Smith',
  age: 28,
  isAdmin: true,
  preferences: {
    theme: 'dark',
    language: 'English'
  },
  scores: [95, 87, 92]
};
const personJsonString = JSON.stringify(person);
console.log('Serialized JSON string:');
console.log(personJsonString);
const parsedPerson = JSON.parse(personJsonString);
console.log('Parsed JavaScript object:');
console.log(parsedPerson);
    </script>
</body>
</html>
```

Output:



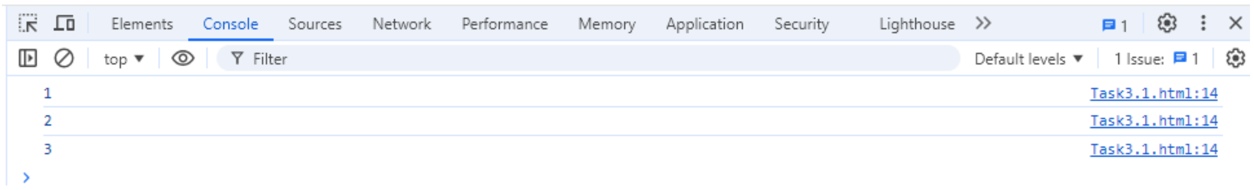
```
Serialized JSON string:                                    Task2.5.html:22
{"firstName":"Alice","lastName":"Smith","age":28,"isAdmin": Task2.5.html:23
true,"preferences":{"theme":"dark","language":"English"},"scores":
[95,87,92]}
Parsed JavaScript object:                                  Task2.5.html:25
▼ Object i                                                 Task2.5.html:26
    age: 28
    firstName: "Alice"
    isAdmin: true
    lastName: "Smith"
  ▶ preferences: {theme: 'dark', language: 'English'}
  ▶ scores: (3) [95, 87, 92]
  ▶ [[Prototype]]: Object
>
```

Closure:

Task 1:

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

Output:



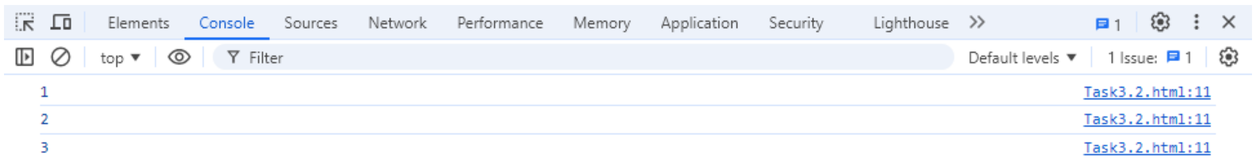
```
1                                                   Task3.1.html:14
2                                                   Task3.1.html:14
3                                                   Task3.1.html:14
```

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 function() {
         count++; console.log(count);
};
}
const counter = createCounter();
counter();
counter();
counter();
</script>
</body>
</html>
```

Output:



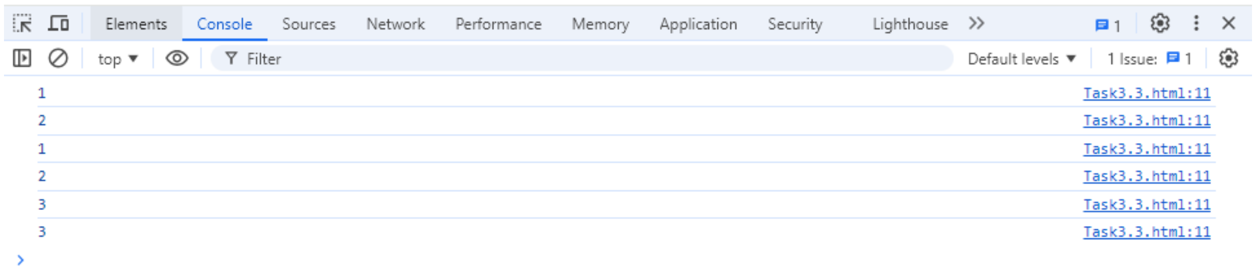
```
1                                                    Task3.2.html:11
2                                                    Task3.2.html:11
3                                                    Task3.2.html:11
```

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 function() {
    count++; console.log(count);
};
}
const counter1 = createCounter();
const counter2 = createCounter();
counter1();
counter1();
counter2();
counter2();
counter1();
counter2();
</script>
```

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

Output:

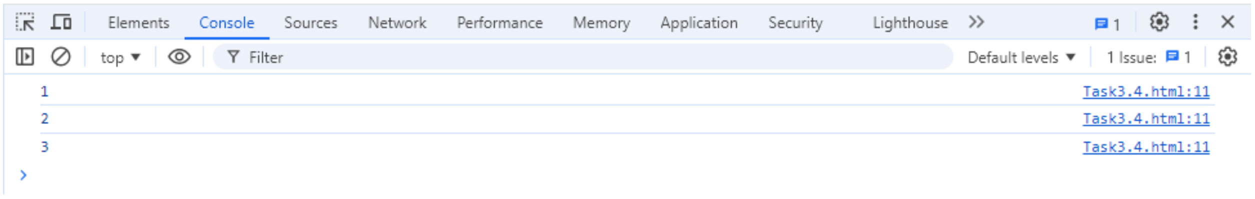


```
1      Task3.3.html:11
2      Task3.3.html:11
1      Task3.3.html:11
2      Task3.3.html:11
3      Task3.3.html:11
3      Task3.3.html:11
```

Task 4:

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

Output:

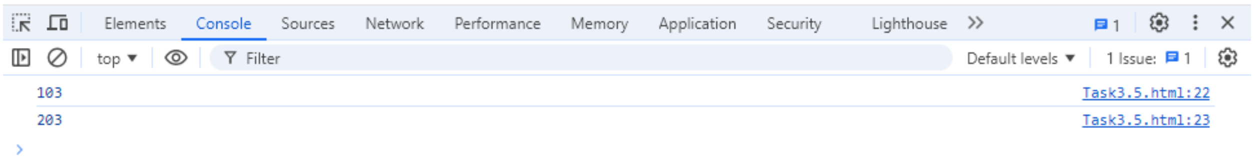


```
1                                                         Task3.4.html:11
2                                                         Task3.4.html:11
3                                                         Task3.4.html:11
```

Task 5:

```
<html>
<head>
<meta charset="UTF-8">
<meta name:"viewport" content="width=device-width,initial-scale=1.0">
</head><body>
<script>
function Createcounter(start_value){
   let count=start_value;
   return{
increment: function(){
   count+=1;},
getCount: function(){
   return count;} };}
const counter1=Createcounter(100);
const counter2=Createcounter(200)
counter1.increment();
counter1.increment();
counter1.increment();
counter2.increment();
counter2.increment();
counter2.increment();
console.log(counter1.getCount());
console.log(counter2.getCount());
</script>
</body>
</html>
```

Output:



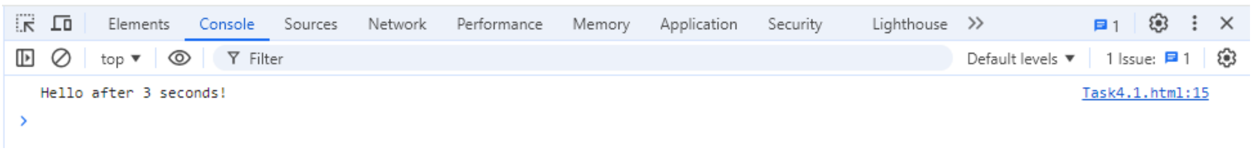
```
103                                                       Task3.5.html:22
203                                                       Task3.5.html:23
```

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 greetAfterDelay(seconds) {
  return new Promise(resolve => {
    setTimeout(() => {
resolve('Hello after ' + seconds + ' seconds!');
}, seconds * 1000);
});
}
greetAfterDelay(3).then(message => {
  console.log(message);
});
</script>
</body>
</html>
```

Output:



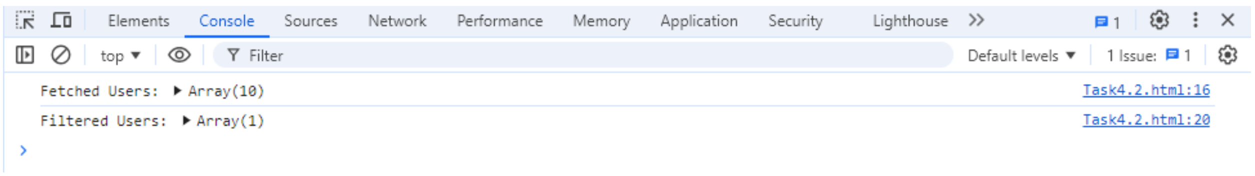
```
Hello after 3 seconds!                                   Task4.1.html:15
```

Task 2:

```
<html>
<head>
<meta charset="UTF-8">
<meta name:"viewport" content="width=device-width,initial-scale=1.0">
</head><body>
<script>
function fetchData() {
return fetch("https://jsonplaceholder.typicode.com/users")
.then(response => response.json());
}
```

```
function processData(users) {
return users.filter(user => user.name === 'Leanne Graham');
}
fetchData()
.then(users => {
console.log('Fetched Users:', users);
return processData(users);
})
.then(filteredUsers => {
console.log('Filtered Users:', filteredUsers);
})
.catch(error => {
   console.error('Error:', error);
});
</script>
</body>
</html>
```

Output:



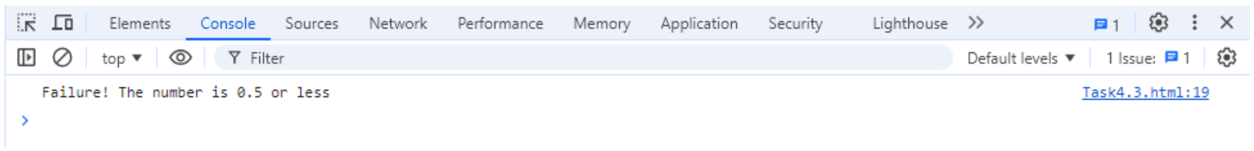
```
Fetched Users:  ▶ Array(10)                                  Task4.2.html:16
Filtered Users:  ▶ Array(1)                                  Task4.2.html:20
```

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.random();
if (randomNumber > 0.5) {
resolve('Success! The number is greater than 0.5');
} else {
reject('Failure! The number is 0.5 or less');
}
});
}
```

```
randomPromise()
.then(result => console.log(result))
.catch(error => console.log(error));
</script>
</body>
</html>
```

Output:



Elements Console Sources Network Performance Memory Application Security Lighthouse

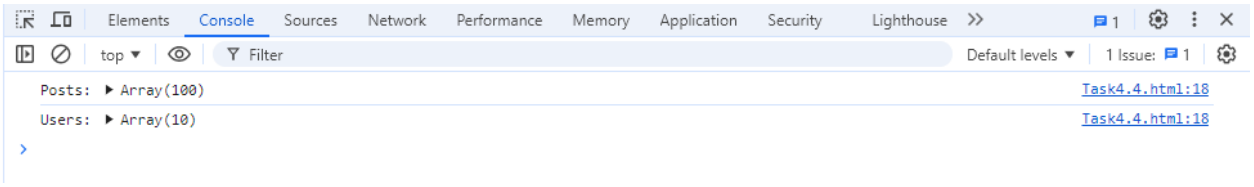
top Filter Default levels 1 Issue: 1

```
Failure! The number is 0.5 or less                                  Task4.3.html:19
```

Task 4:

```
<html>
<head>
<meta charset="UTF-8">
<meta name:"viewport" content="width=device-width,initial-scale=1.0">
</head><body>
<script>
function fetchPosts() {
return fetch('https://jsonplaceholder.typicode.com/posts')
.then(response => response.json());
}
function fetchUsers() {
return fetch("https://jsonplaceholder.typicode.com/users")
.then(response => response.json());
}
Promise.all([fetchPosts(), fetchUsers()])
.then(results => {
const [posts, users] = results;
console.log('Posts:', posts); console.log('Users:', users);
})
.catch(error => {
   console.log('Error:', error);
});

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

Output:



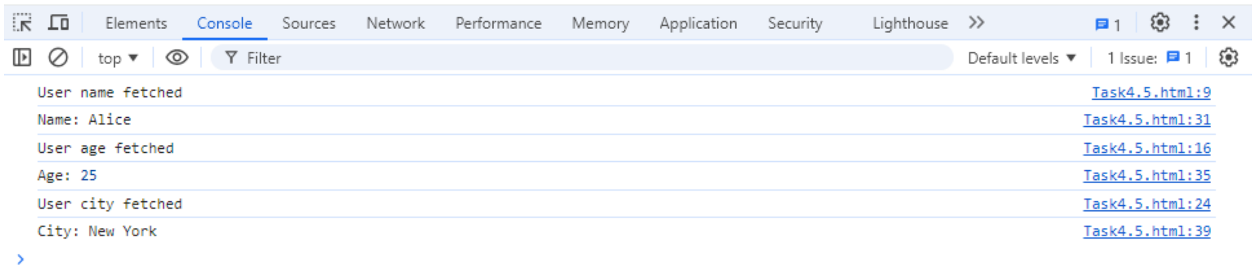
```
Posts:  ▶ Array(100)                                   Task4.4.html:18
Users:  ▶ Array(10)                                    Task4.4.html:18
```

Task 5:

```
<html>
<head>
<meta charset="UTF-8">
<meta name:"viewport" content="width=device-width,initial-scale=1.0">
</head><body>
<script>
function fetchUserName() {
return new Promise((resolve) => { setTimeout(() => {
console.log('User name fetched'); resolve('Alice');
}, 1000);
});
}
function fetchUserAge() {
return new Promise((resolve) => {
   setTimeout(() => {
console.log('User age fetched');
resolve(25);
}, 1000);
});
}
function fetchUserCity() {
return new Promise((resolve) => {
   setTimeout(() => {
console.log('User city fetched');
resolve('New York');
}, 1000);
});
}
fetchUserName()
.then((name) => {
   console.log('Name:', name);
   return fetchUserAge();
})
.then((age) => {
   console.log('Age:', age);
```

```
  return fetchUserCity();
})
.then((city) => {
  console.log('City:', city);
})
.catch((error) => {
  console.log('Error:', error);
});

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

Output:



```
User name fetched        Task4.5.html:9
Name: Alice              Task4.5.html:31
User age fetched         Task4.5.html:16
Age: 25                  Task4.5.html:35
User city fetched        Task4.5.html:24
City: New York           Task4.5.html:39
```

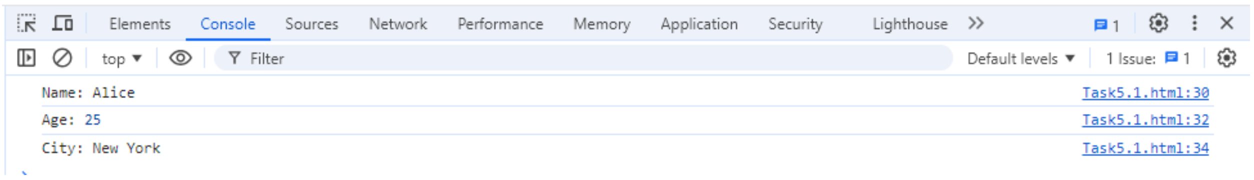
Async/await:

Task 1:

```
<html>
<head>
<meta charset="UTF-8">
<meta name:"viewport" content="width=device-width,initial-scale=1.0">
</head><body>
<script>
function fetchUserName() {
return new Promise((resolve) => {
  setTimeout(() => {
  resolve('Alice');
}, 1000);
});
}
function fetchUserAge() {
return new Promise((resolve) => {
  setTimeout(() => {
```

```
  resolve(25);
}, 1000);
});
}
function fetchUserCity() {
return new Promise((resolve) => {
  setTimeout(() => {
  resolve('New York');
}, 1000);
});
}
async function getUserDetails() {
  const name = await fetchUserName();
  console.log('Name:', name);
const age = await fetchUserAge();
console.log('Age:', age);
const city = await fetchUserCity();
console.log('City:', city);
}
getUserDetails();
</script>
</body>
</html>
```

Output:

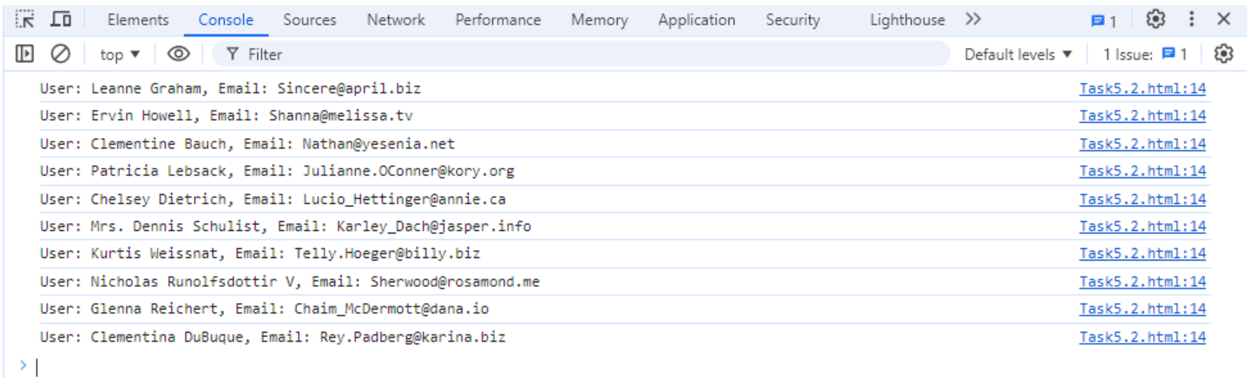


```
Name: Alice                                   Task5.1.html:30
Age: 25                                       Task5.1.html:32
City: New York                                Task5.1.html:34
```

Task 2:

```
<html>
<head>
<meta charset="UTF-8">
<meta name:"viewport" content="width=device-width,initial-scale=1.0">
</head><body>
<script>
async function fetchAndProcessData() { try {
const response = await fetch('https://jsonplaceholder.typicode.com/users');
if (!response.ok) {
throw new Error('Network response was not ok');
}
const data = await response.json();
data.forEach(user => {
console.log(`User: ${user.name}, Email: ${user.email}`);
});

} catch (error) {

console.error('There was an error fetching the data:', error);
}
}
fetchAndProcessData();

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

Output:

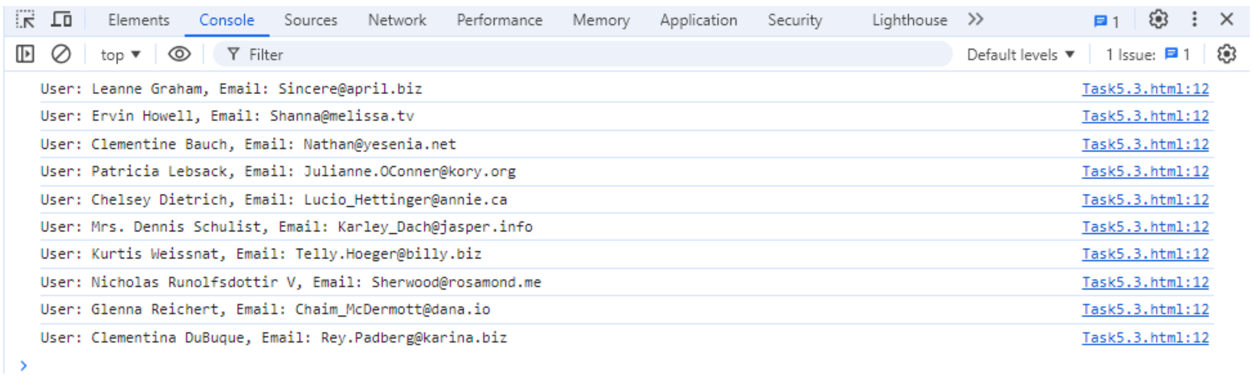


```
User: Leanne Graham, Email: Sincere@april.biz                    Task5.2.html:14
User: Ervin Howell, Email: Shanna@melissa.tv                     Task5.2.html:14
User: Clementine Bauch, Email: Nathan@yesenia.net                Task5.2.html:14
User: Patricia Lebsack, Email: Julianne.OConner@kory.org         Task5.2.html:14
User: Chelsey Dietrich, Email: Lucio_Hettinger@annie.ca          Task5.2.html:14
User: Mrs. Dennis Schulist, Email: Karley_Dach@jasper.info       Task5.2.html:14
User: Kurtis Weissnat, Email: Telly.Hoeger@billy.biz             Task5.2.html:14
User: Nicholas Runolfsdottir V, Email: Sherwood@rosamond.me      Task5.2.html:14
User: Glenna Reichert, Email: Chaim_McDermott@dana.io            Task5.2.html:14
User: Clementina DuBuque, Email: Rey.Padberg@karina.biz          Task5.2.html:14
```

Task 3:

```
<html>
<head>
<meta charset="UTF-8">
<meta name:"viewport" content="width=device-width,initial-scale=1.0">
</head><body>
<script>
async function fetchData() { try {
const response = await fetch('https://jsonplaceholder.typicode.com/users');
if (!response.ok) {
throw new Error(`HTTP error! Status: ${response.status}`);
}
const data = await response.json();
data.forEach(user => {
console.log(`User: ${user.name}, Email: ${user.email}`);
});

} catch (error) {
console.error('There was an error fetching the data:', error.message);
}
}
fetchData();
</script>
</body>
</html>
```

Output:

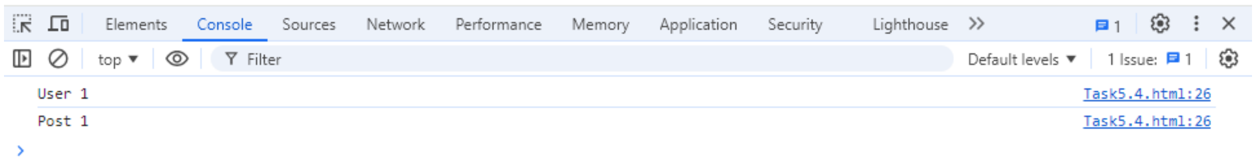


```
User: Leanne Graham, Email: Sincere@april.biz                    Task5.3.html:12
User: Ervin Howell, Email: Shanna@melissa.tv                     Task5.3.html:12
User: Clementine Bauch, Email: Nathan@yesenia.net                Task5.3.html:12
User: Patricia Lebsack, Email: Julianne.OConner@kory.org         Task5.3.html:12
User: Chelsey Dietrich, Email: Lucio_Hettinger@annie.ca          Task5.3.html:12
User: Mrs. Dennis Schulist, Email: Karley_Dach@jasper.info       Task5.3.html:12
User: Kurtis Weissnat, Email: Telly.Hoeger@billy.biz             Task5.3.html:12
User: Nicholas Runolfsdottir V, Email: Sherwood@rosamond.me      Task5.3.html:12
User: Glenna Reichert, Email: Chaim_McDermott@dana.io            Task5.3.html:12
User: Clementina DuBuque, Email: Rey.Padberg@karina.biz          Task5.3.html:12
>
```

Task 4:

```
<html>
<head>
<meta charset="UTF-8">
<meta name:"viewport" content="width=device-width,initial-scale=1.0">
</head><body>
<script>
function fetchUser(id) {
return new Promise((resolve) => {
   setTimeout(() => {
   resolve(`User ${id}`);
}, 1000);
});
}
function fetchPost(id) {
return new Promise((resolve) => {
   setTimeout(() => {
   resolve(`Post ${id}`);
}, 1500);
});
}
async function fetchData() {
   try {
const [user, post] = await Promise.all([ fetchUser(1),
fetchPost(1)
]);
console.log(user); console.log(post);
} catch (error) {
console.error('Error fetching data:', error);
}
}
fetchData();
</script>
</body>
</html>
```

Output:

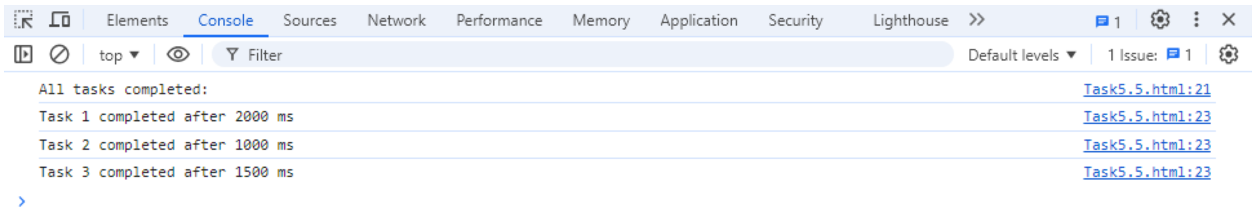


```
User 1                                                     Task5.4.html:26
Post 1                                                     Task5.4.html:26
>
```

Task 5:

```
<html>
<head>
<meta charset="UTF-8">
<meta name:"viewport" content="width=device-width,initial-scale=1.0">
</head><body>
<script>
function asyncTask(name, delay) {
   return new Promise((resolve) => {
setTimeout(() => {
resolve(`${name} completed after ${delay} ms`);
}, delay);
});
}
async function waitForAllTasks() { try {
   const results = await
   Promise.all([
asyncTask('Task 1', 2000),
asyncTask('Task 2', 1000),
asyncTask('Task 3', 1500)
]);
console.log('All tasks completed:');
results.forEach((result) =>
console.log(result));

} catch (error) {
console.error('An error occurred:', error);
}
}
waitForAllTasks();
</script>
</body>
</html>
```

Output:



```
All tasks completed:                              Task5.5.html:21
Task 1 completed after 2000 ms                    Task5.5.html:23
Task 2 completed after 1000 ms                    Task5.5.html:23
Task 3 completed after 1500 ms                    Task5.5.html:23
```

Modules introduction, Export and Import:

Task 1:

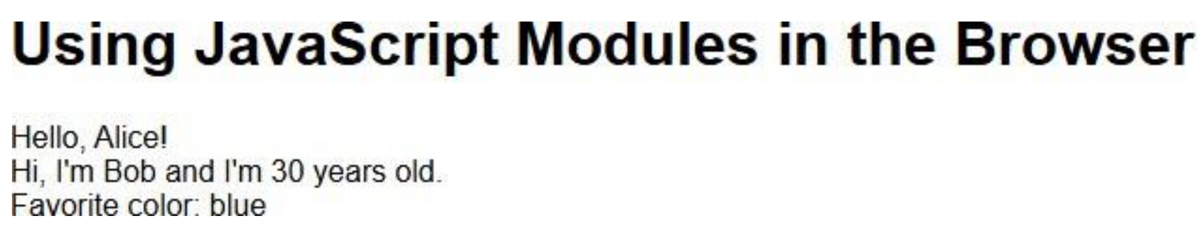
```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>ES Module Example</title>
<style> body {
font-family: Arial, sans-serif; margin: 20px;
}
</style>
</head>
<body>
<h1>Using JavaScript Modules in the Browser</h1>
<div id="greeting"></div>
<div id="introduction"></div>
```

```
<div id="color"></div>
<script type="module">
import { greet, Person, favoriteColor }
from './myModule.js' document.getElementById('greeting').textContent = greet('Alice');
const person1 = new Person('Bob', 30);
document.getElementById('introduction').textContent = person1.introduce();
document.getElementById('color').textContent = `Favorite color:
${favoriteColor}`;
</script>
</body>
</html>
```

Module.js:

```
export function greet(name) {
   return `Hello, ${name}!`;
}
export class Person {
   constructor(name, age) {
   this.name = name;
   this.age = age;
}
introduce() {
return `Hi, I'm ${this.name} and I'm ${this.age} years old.`;
}
}
export const favoriteColor = 'blue';
```

Output:



```
Using JavaScript Modules in the Browser

Hello, Alice!
Hi, I'm Bob and I'm 30 years old.
Favorite color: blue
```

Task 2:

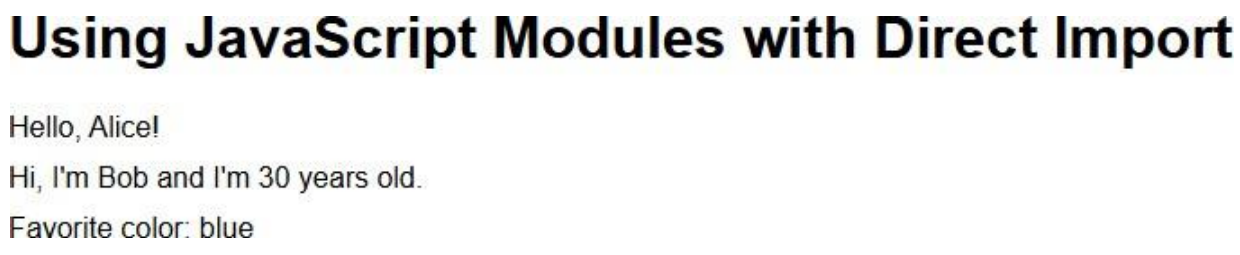
```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
```

```
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>JavaScript Modules Example</title>
<style> body {
font-family: Arial, sans-serif; margin: 20px;
}
#greeting, #introduction, #color { margin: 10px 0;
}
</style>
</head>
<body>
<h1>Using JavaScript Modules with Direct Import</h1>
<div id="greeting"></div>
<div id="introduction"></div>
<div id="color"></div>
<script type="module">
import { greet, Person, favoriteColor }
from './myModule.js'; const greetingElement = document.getElementById('greeting');
greetingElement.textContent = greet('Alice');
const person1 = new Person('Bob', 30);
const introductionElement = document.getElementById('introduction');
introductionElement.textContent = person1.introduce();
const colorElement = document.getElementById('color');
colorElement.textContent = `Favorite color: ${favoriteColor}`;
</script>
</body>
</html>
```

Module.js:

```
export function greet(name) {
  return `Hello, ${name}!`;
}
export class Person {
  constructor(name, age) {
    this.name = name;
    this.age = age;
  }
  introduce() {
  return `Hi, I'm ${this.name} and I'm ${this.age} years old.`;
  }
  }
  export const favoriteColor = 'blue';
```

Output:



```
Using JavaScript Modules with Direct Import

Hello, Alice!
Hi, I'm Bob and I'm 30 years old.
Favorite color: blue
```

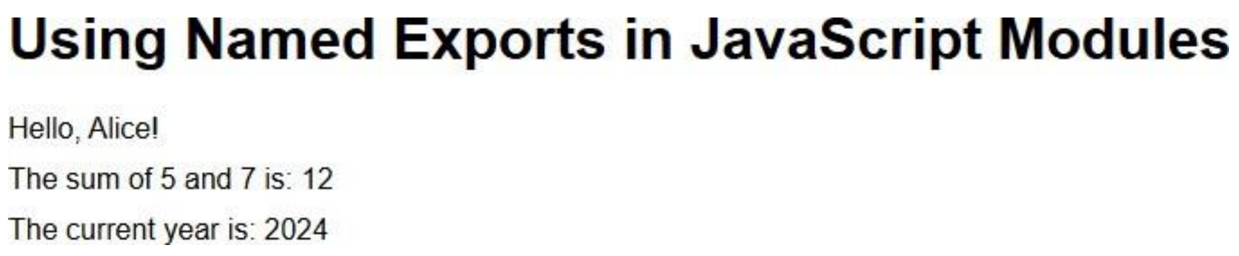
Task 3:

```html
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>JavaScript Modules Example</title>
<style> body {
font-family: Arial, sans-serif; margin: 20px;
}
#greeting, #sum, #year { margin: 10px 0;
}
</style>
</head>
<body>
<h1>Using Named Exports in JavaScript Modules</h1>
<div id="greeting"></div>
<div id="sum"></div>
<div id="year"></div>
<script type="module">
import { greet, sum, getCurrentYear }
from './myModule.js';
const greetingElement = document.getElementById('greeting');
greetingElement.textContent = greet('Alice');
const sumElement = document.getElementById('sum');
sumElement.textContent = `The sum of 5 and 7 is: ${sum(5, 7)}`;
const yearElement = document.getElementById('year');
yearElement.textContent = `The current year is: ${getCurrentYear()}`;
</script>
</body>
</html>
```

Module.js:

```
export function greet(name) {
   return `Hello, ${name}!`;
}
export function sum(a, b) {
   return a + b;
}
export function getCurrentYear() {
   return new Date().getFullYear();
}
```

Output:



```
Using Named Exports in JavaScript Modules

Hello, Alice!
The sum of 5 and 7 is: 12
The current year is: 2024
```

Task 4:

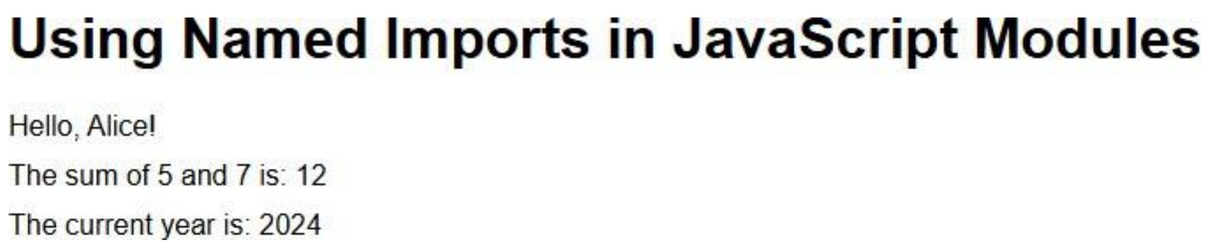
```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Using Named Imports in JavaScript Modules</title>
<style> body {
font-family: Arial, sans-serif; margin: 20px;
}
#greeting, #sum, #year { margin: 10px 0;
}
</style>
</head>
<body>
<h1>Using Named Imports in JavaScript Modules</h1>
<div id="greeting"></div>
<div id="sum"></div>
<div id="year"></div>
<script type="module">
import { greet, sum }
```

```
from './myModule.js';
const greetingElement = document.getElementById('greeting');
greetingElement.textContent = greet('Alice');
const sumElement = document.getElementById('sum');
sumElement.textContent = `The sum of 5 and 7 is: ${sum(5, 7)}`;
const yearElement = document.getElementById('year');
yearElement.textContent = `The current year is: ${new
Date().getFullYear()}`;
</script>
</body>
</html>
```

Module.js:

```
export function greet(name) {
    return `Hello, ${name}!`;
}
export function sum(a, b) {
    return a + b;
}
```

Output:



```
Using Named Imports in JavaScript Modules

Hello, Alice!
The sum of 5 and 7 is: 12
The current year is: 2024
```

Task 5:

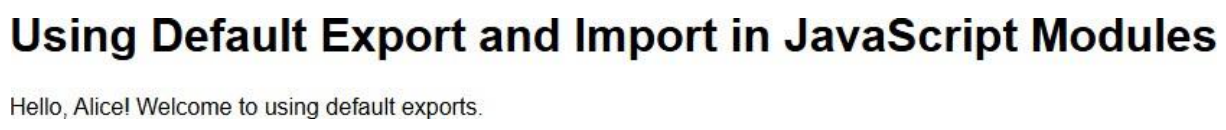
```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Using Default Export and Import</title>
<style> body {
```

```
font-family: Arial, sans-serif; margin: 20px;
}
#greeting { margin: 10px 0;
}
</style>
</head>
<body>
<h1>Using Default Export and Import in JavaScript Modules</h1>
<div id="greeting"></div>
<script type="module">
import greet from './myModule.js';
const greetingElement = document.getElementById('greeting');
greetingElement.textContent = greet('Alice');
</script>
</body>
</html>
```

Module.js:

```
function greet(name) {
    return `Hello, ${name}! Welcome to using default exports.`;
    }
    export default greet;
```

Output:



Using Default Export and Import in JavaScript Modules

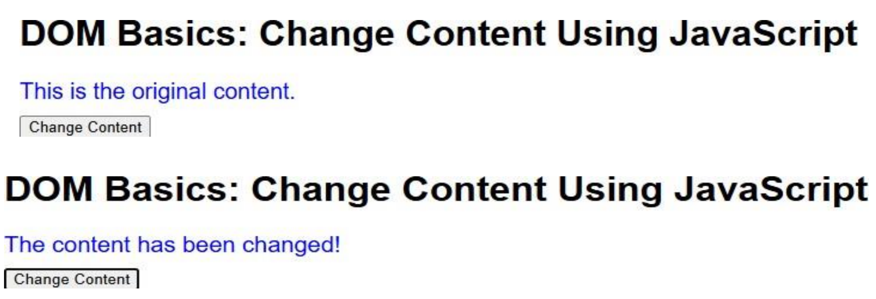
Hello, Alice! Welcome to using default exports.

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>DOM Basics: Change Content</title>
<style> body {
font-family: Arial, sans-serif; margin: 20px;
}
#message {
font-size: 20px; color: blue; margin: 10px 0;
}
</style>
</head>
<body>

<h1>DOM Basics: Change Content Using JavaScript</h1>
<p id="message">This is the original content.</p>
<button onclick="changeContent()">Change Content</button>
<script>
function changeContent() {
var element = document.getElementById('message');
element.textContent = 'The content has been changed!';
}
</script>
</body>
</html>
```

Output:



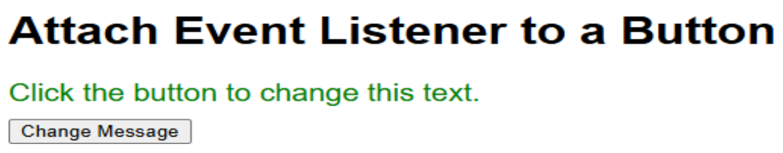
```
DOM Basics: Change Content Using JavaScript

This is the original content.
Change Content

DOM Basics: Change Content Using JavaScript

The content has been changed!
Change Content
```

Task 2:

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Button Event Listener Example</title>
<style> body {
font-family: Arial, sans-serif; margin: 20px;
}
```

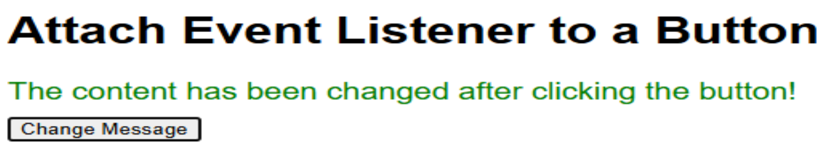
```
#message {
font-size: 20px; color: green; margin: 10px 0;
}
</style>
</head>
<body>

<h1>Attach Event Listener to a Button</h1>
<p id="message">Click the button to change this text.</p>
<button id="changeMessageButton">Change Message</button>

<script>
const button = document.getElementById('changeMessageButton');
const messageElement = document.getElementById('message');
button.addEventListener('click', function() {
messageElement.textContent = 'The content has been changed after clicking the button!';
});
</script>
</body>
</html>
```

Output:



Attach Event Listener to a Button

Click the button to change this text.



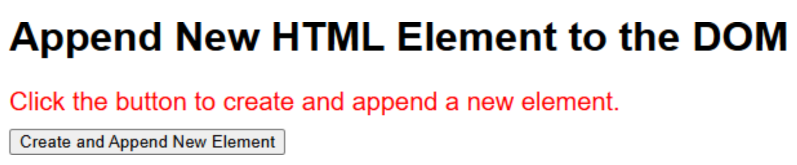
Attach Event Listener to a Button

The content has been changed after clicking the button!

Task 3:

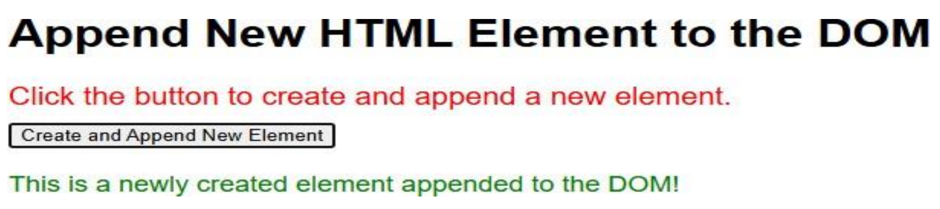
```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Create and Append a New HTML Element</title>
<style> body {
font-family: Arial, sans-serif;
margin: 20px;
}
#message {
font-size: 20px; color: red; margin: 10px 0;
}
#newElementContainer { margin-top: 20px;
}
</style>
</head>
<body>
<h1>Append New HTML Element to the DOM</h1>
<div id="message">Click the button to create and append a new element.</div>
<button id="createElementButton">Create and Append New Element</button>
<div id="newElementContainer"></div>
<script>
const button = document.getElementById('createElementButton');
const container = document.getElementById('newElementContainer');
button.addEventListener('click', function() {
const newElement = document.createElement('p');
newElement.textContent = 'This is a newly created element appended to the DOM!';
newElement.style.color = 'green';
newElement.style.fontSize = '18px';
container.appendChild(newElement);
});
</script>
</body>
</html>
```

Output:



Append New HTML Element to the DOM

Click the button to create and append a new element.


Append New HTML Element to the DOM

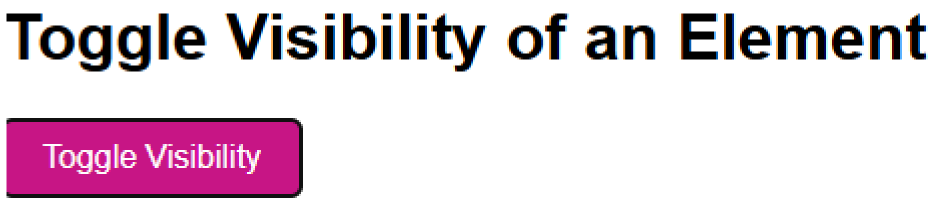
Click the button to create and append a new element.

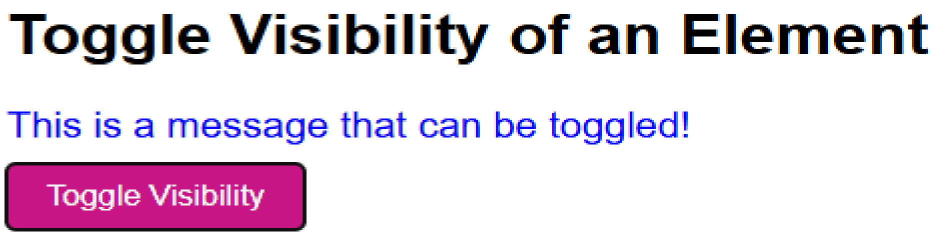
This is a newly created element appended to the DOM!

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 Element Visibility</title>
<style> body {
font-family: Arial, sans-serif; margin: 20px;
}
#toggleMessage { font-size: 20px; color: blue; margin: 10px 0; display: block;
}
#toggleButton { padding: 10px 20px; font-size: 16px; cursor: pointer;
background-color:mediumvioletred; color: white;
border: none; border-radius: 5px;
}
#toggleButton:hover { background-color: pink;
}
</style>
</head>
<body>

<h1>Toggle Visibility of an Element</h1>
<div id="toggleMessage">This is a message that can be toggled!</div>
<button id="toggleButton">Toggle Visibility</button>
<script>
  const toggleButton = document.getElementById('toggleButton');
  const toggleMessage = document.getElementById('toggleMessage');
  function toggleVisibility() {
  if (toggleMessage.style.display === 'none') {
    toggleMessage.style.display = 'block';
  } else {
  toggleMessage.style.display = 'none';
  }
  }
  toggleButton.addEventListener('click', toggleVisibility);
  </script>
  </body>
  </html>
```

Output:



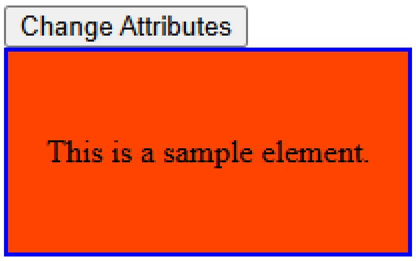

Toggle Visibility of an Element
This is a message that can be toggled!
Toggle Visibility

Task 5:

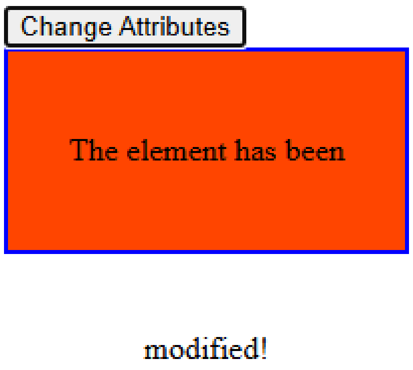
```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Modify Element Attributes</title>
<style> #myElement { width: 200px; height: 100px;
background-color: orangered; text-align: center;
line-height: 100px; border: 2px solid blue;
}
</style>
</head>
<body>
<button onclick="changeAttributes()">Change Attributes</button>
<div id="myElement" class="box" title="Original Title"> This is a sample element.
</div>
```

```
<script>
function changeAttributes() {
var element = document.getElementById("myElement");
var currentClass = element.getAttribute("class");
var currentTitle = element.getAttribute("title");
console.log("Current class:", currentClass);
console.log("Current title:", currentTitle);
element.setAttribute("class", "modified-box");
element.setAttribute("title", "Modified Title");
element.textContent = "The element has been modified!";
console.log("New class:", element.getAttribute("class"));
console.log("New title:", element.getAttribute("title"));
}
</script>
</body>
</html>
```

Output:



This is a sample element.



The element has been

modified!