MERN Stack Training

Weekly Tasks

Week 3 & 4:

1. **Recursion and stack**:

    - Task 1: Implement a function to calculate the factorial of a number using recursion.
    - Task 2: Write a recursive function to find the nth Fibonacci number.
    - Task 3: Create a function to determine the total number of ways one can climb a staircase with 1, 2, or 3 steps at a time using recursion.
    - Task 4: Write a recursive function to flatten a nested array structure.
    - Task 5: Implement the recursive Tower of Hanoi solution.

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

    - Task 1: Write a function that takes an arbitrary number of arguments and returns their sum.
    - Task 2: Modify a function to accept an array of numbers and return their sum using the spread syntax.
    - Task 3: Create a deep clone of an object using JSON methods.
    - Task 4: Write a function that returns a new object, merging two provided objects using the spread syntax.
    - Task 5: Serialize a JavaScript object into a JSON string and then parse it back into an object.

3. **Closure**:

    - Task 1: Create a function that returns another function, capturing a local variable.
    - Task 2: Implement a basic counter function using closure, allowing incrementing and displaying the current count.
    - Task 3: Write a function to create multiple counters, each with its own separate count.
    - Task 4: Use closures to create private variables within a function.
    - Task 5: Build a function factory that generates functions based on some input using closures.

4. **Promise, Promises chaining**:

- Task 1: Create a new promise that resolves after a set number of seconds and returns a greeting.
- Task 2: Fetch data from an API using promises, and then chain another promise to process this data.
- Task 3: Create a promise that either resolves or rejects based on a random number.
- Task 4: Use Promise.all to fetch multiple resources in parallel from an API.
- Task 5: Chain multiple promises to perform a series of asynchronous actions in sequence.

5. **Async/await**:

   - Task 1: Rewrite a promise-based function using async/await.
   - Task 2: Create an async function that fetches data from an API and processes it.
   - Task 3: Implement error handling in an async function using try/catch.
   - Task 4: Use async/await in combination with Promise.all.
   - Task 5: Create an async function that waits for multiple asynchronous operations to complete before proceeding.

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

   - Task 1: Create a module that exports a function, a class, and a variable.
   - Task 2: Import the module in another JavaScript file and use the exported entities.
   - Task 3: Use named exports to export multiple functions from a module.
   - Task 4: Use named imports to import specific functions from a module.
   - Task 5: Use default export and import for a primary function of a module.

7. **Browser: DOM Basics**:

   - Task 1: Select an HTML element by its ID and change its content using JavaScript.
   - Task 2: Attach an event listener to a button, making it perform an action when clicked.
   - Task 3: Create a new HTML element and append it to the DOM.
   - Task 4: Implement a function to toggle the visibility of an element.
   - Task 5: Use the DOM API to retrieve and modify the attributes of an element.

Mini Project: **"Task Scheduler"**

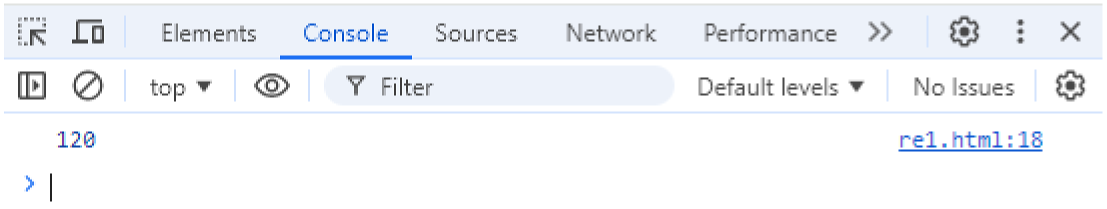
Objective:
Develop a web-based task management application where users can add, delete, modify, and

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(number){
            if(number==0 || number==1){
                return 1;
            }else{
                return number*factorial(number-1);
            }
        }
        console.log(factorial(5));
    </script>
</body>
</html>
```

Output:

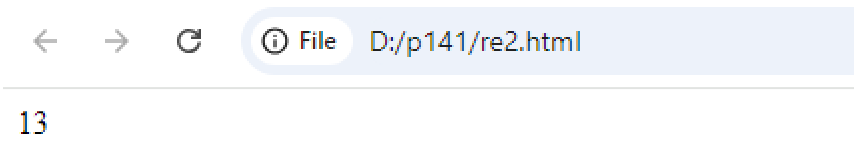


```
120                                                    re1.html:18
```

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 fibonaaci(num){
            if(num==0){
                return 0;
            }else if(num==1){
                return 1;
            }else{
                return fibonaaci(num-1)+fibonaaci(num-2);
            }
        }
        document.write(fibonaaci(7));
    </script>
</body>
</html>
```

Output:



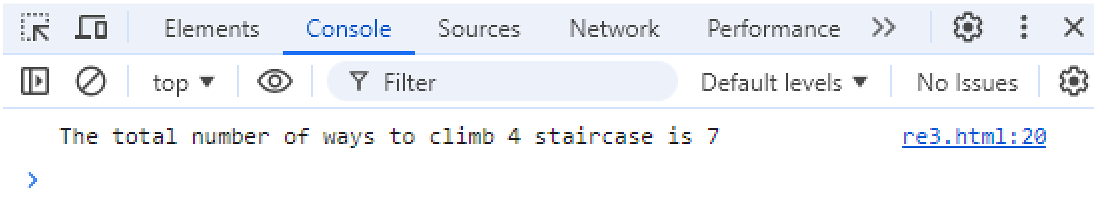
File D:/p141/re2.html

```
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 climbStaircase(num){
        if(num==0){
            return 1;
        }else if(num<0){
            return 0;
        }else{
            return climbStaircase(num-1) + climbStaircase(num-2) +
climbStaircase(num-3);
        }
    }
    let num=4;
    console.log("The total number of ways to climb "+num+" staircase is "
+climbStaircase(num));
</script>
</body>
</html>
```

Output:

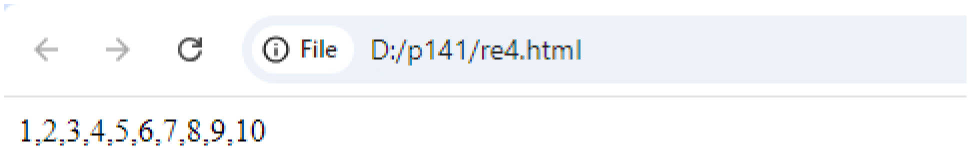


```
The total number of ways to climb 4 staircase is 7        re3.html:20
```

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 flattenArray(arr) {
        let result = [];

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

Output:

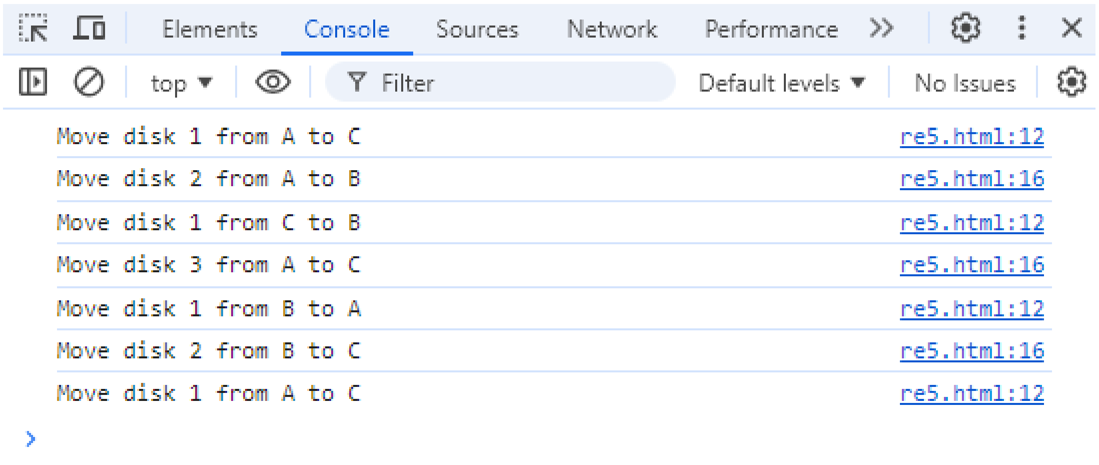


```
1,2,3,4,5,6,7,8,9,10
```

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, auxillary, destination) {
        if (n == 1) {
          console.log(`Move disk 1 from ${source} to ${destination}`);
          return;
        }
        towerOfHanoi(n - 1, source, destination, auxillary);
        console.log(`Move disk ${n} from ${source} to ${destination}`);
        towerOfHanoi(n - 1, auxillary, source, destination);
      }
      let numOfDisk = 3;
      towerOfHanoi(numOfDisk, "A", "B", "C");
    </script>
  </body>
</html>
```

Output:



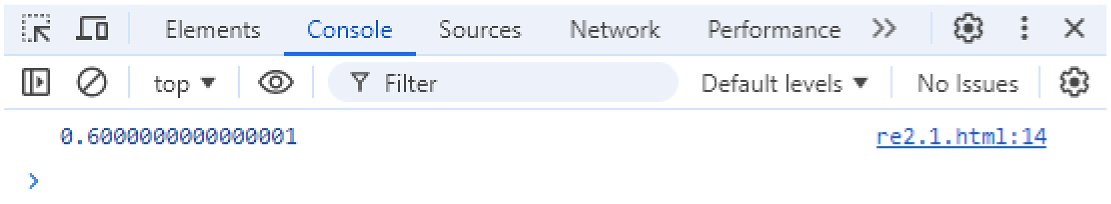
```
Move disk 1 from A to C                                   re5.html:12
Move disk 2 from A to B                                   re5.html:16
Move disk 1 from C to B                                   re5.html:12
Move disk 3 from A to C                                   re5.html:16
Move disk 1 from B to A                                   re5.html:12
Move disk 2 from B to C                                   re5.html:16
Move disk 1 from A to C                                   re5.html:12
```

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 arbitrary(num1, num2) {
        return num1 + num2;
      }
      let res = arbitrary(8 / 10, -0.2);
      console.log(res);
    </script>
  </body>
</html>
```

Output:

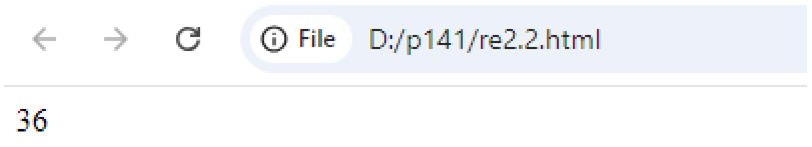


```
0.6000000000000001                                    re2.1.html:14
```

Task 2:

```html
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Document</title>
  </head>
  <body>
    <script>
      function sum(...arr) {
        return arr.reduce((total, currentValue) => total + currentValue);
      }
      const array = [17, 19];
      document.write(sum(...array));
    </script>
  </body>
</html>
```

Output:

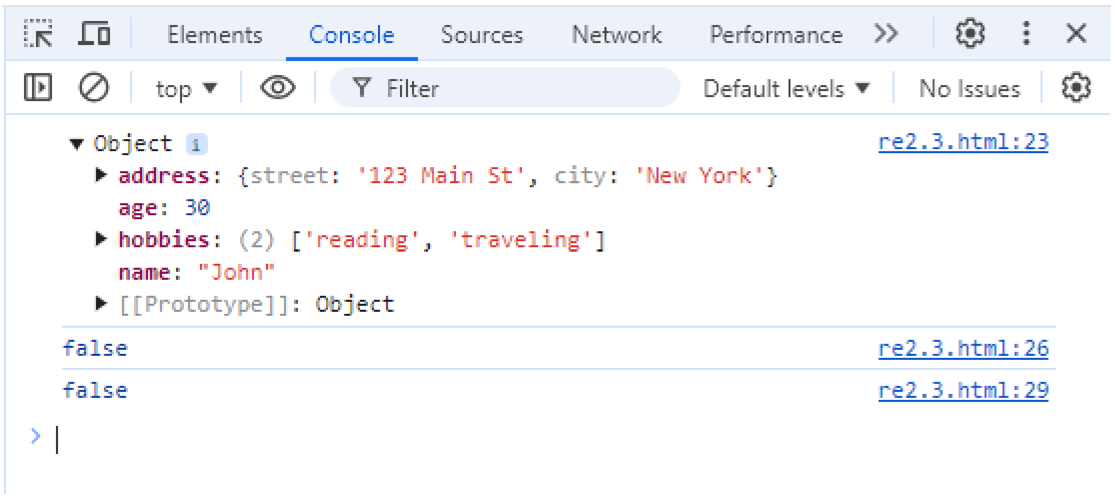


```
36
```

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:

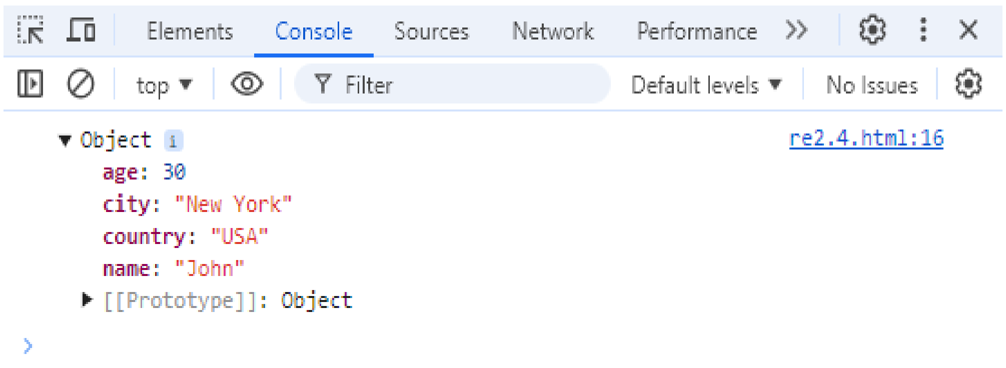


```
▼ Object                                                        re2.3.html:23
  ▶ address: {street: '123 Main St', city: 'New York'}
    age: 30
  ▶ hobbies: (2) ['reading', 'traveling']
    name: "John"
  ▶ [[Prototype]]: Object
false                                                           re2.3.html:26
false                                                           re2.3.html:29
```

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:



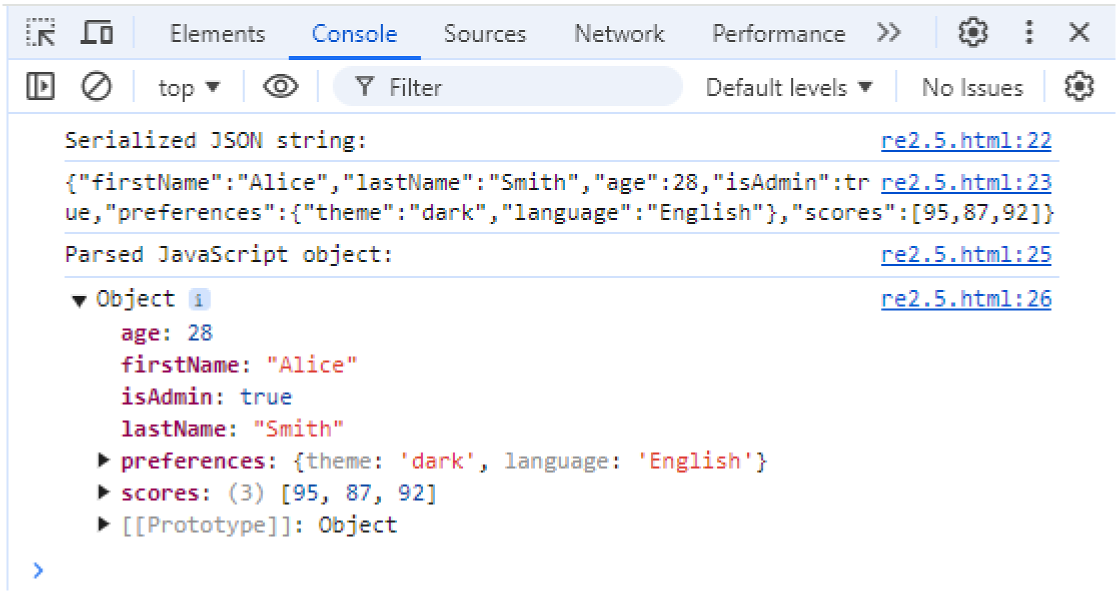
```
▼ Object                                    re2.4.html:16
    age: 30
    city: "New York"
    country: "USA"
    name: "John"
  ▶ [[Prototype]]: Object
```

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:



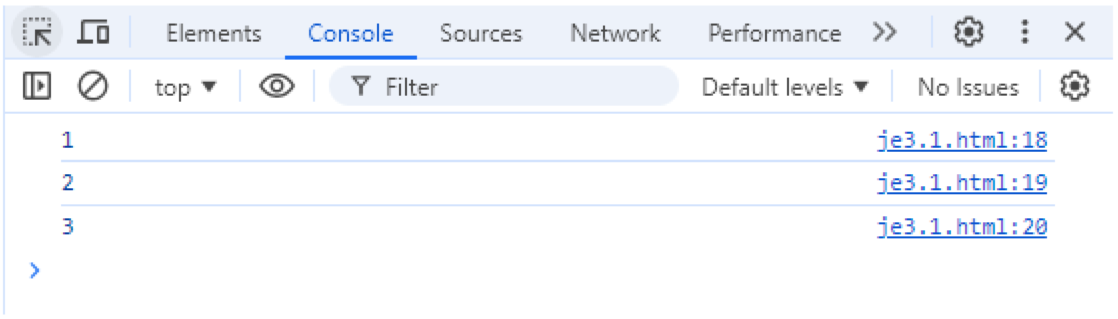
```
Serialized JSON string:                                          re2.5.html:22
{"firstName":"Alice","lastName":"Smith","age":28,"isAdmin":tr re2.5.html:23
ue,"preferences":{"theme":"dark","language":"English"},"scores":[95,87,92]}
Parsed JavaScript object:                                        re2.5.html:25
▼ Object i                                                       re2.5.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:

```
<!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 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:



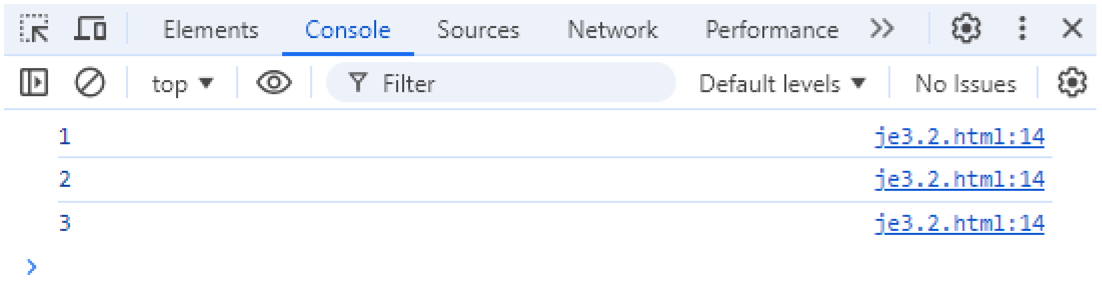
```
1                                              je3.1.html:18
2                                              je3.1.html:19
3                                              je3.1.html:20
>
```

Task 2:

```
<!DOCTYPE html>
```

```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 createCounter() {
        let count = 0;
        return function () {
          count++;
          console.log(count);
        };
      }
      const counter = createCounter();
      counter();
      counter();
      counter();
    </script>
  </body>
</html>
```

Output:



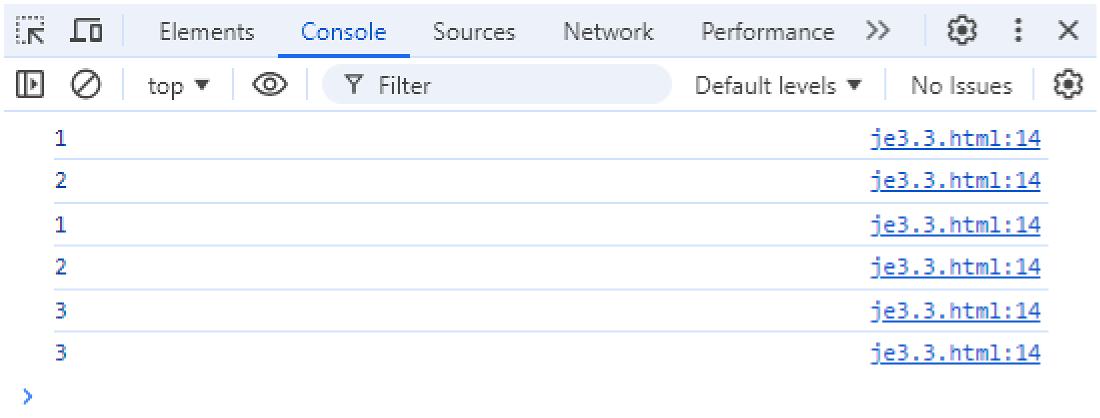
```
1                                              je3.2.html:14
2                                              je3.2.html:14
3                                              je3.2.html:14
```

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>
      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:



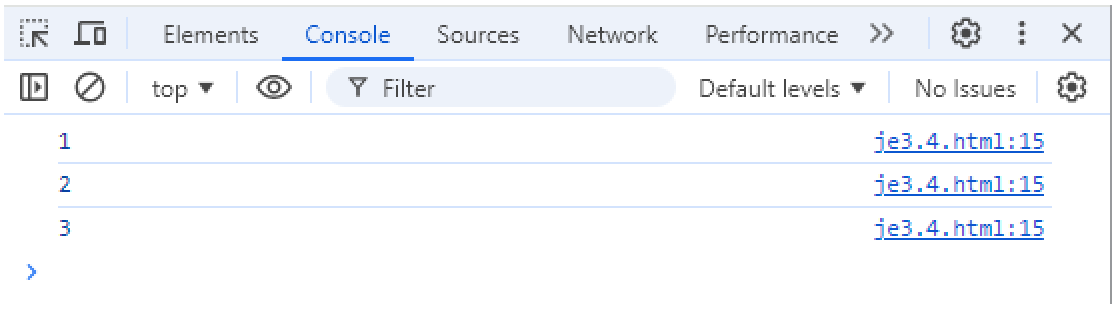
```
1    je3.3.html:14
2    je3.3.html:14
1    je3.3.html:14
2    je3.3.html:14
3    je3.3.html:14
3    je3.3.html:14
```

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 createCounter() {
        let count = 0;
        return {
          increment: function () {
            count++;
            console.log(count);
          },
        };
      }
      const counter = createCounter();
      counter.increment();
      counter.increment();
      counter.increment();
    </script>
  </body>
</html>
```

Output:

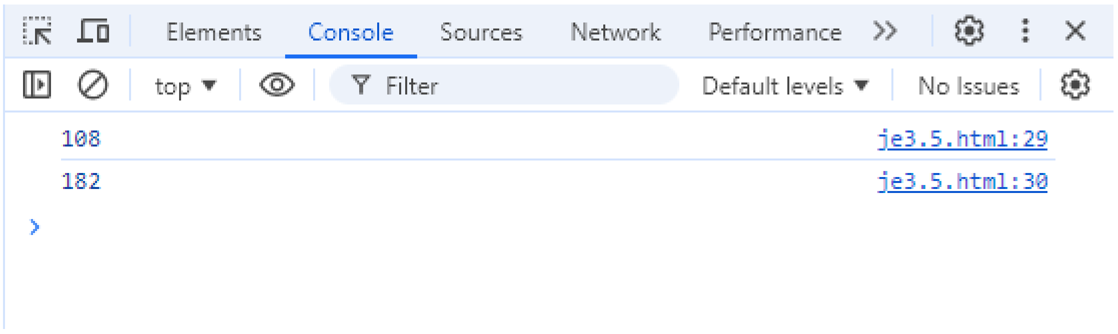


```
1                                                  je3.4.html:15
2                                                  je3.4.html:15
3                                                  je3.4.html:15
>
```

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 Createcounter(start_value) {
        let count = start_value;
        return {
          increment: function () {
            count += 1;
          },
          getCount: function () {
            return count;
          },
        };
      }
      const counter1 = Createcounter(105);
      const counter2 = Createcounter(179);
      counter1.increment();
      counter1.increment();
      counter1.increment();
      counter2.increment();
      counter2.increment();
      counter2.increment();
      console.log(counter1.getCount());
      console.log(counter2.getCount());
    </script>
  </body>
</html>
```

Output:



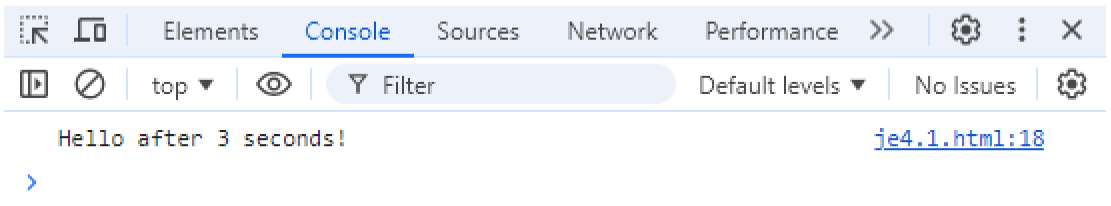
```
108                                          je3.5.html:29
182                                          je3.5.html:30
```

Promise, Promises chaining:

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 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:

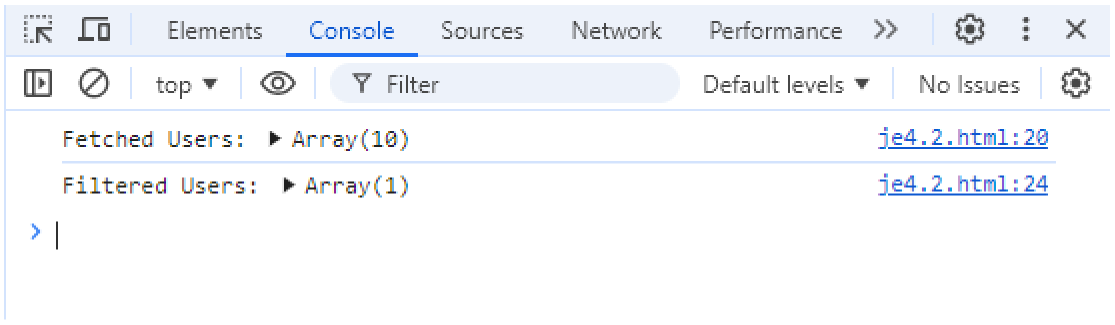


```
Hello after 3 seconds!                                je4.1.html:18
```

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 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:

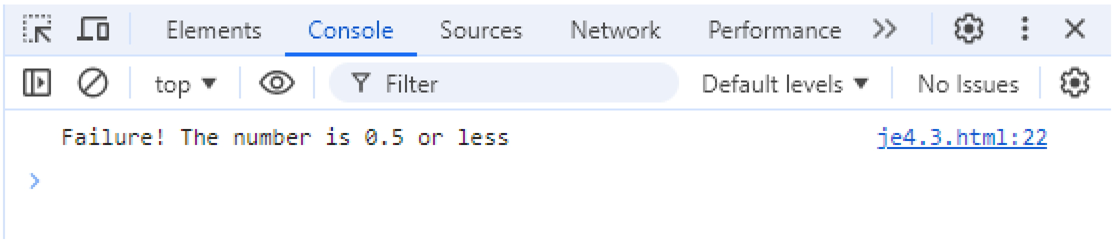


```
Fetched Users: ▶ Array(10)                          je4.2.html:20
Filtered Users: ▶ Array(1)                          je4.2.html:24
```

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 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:

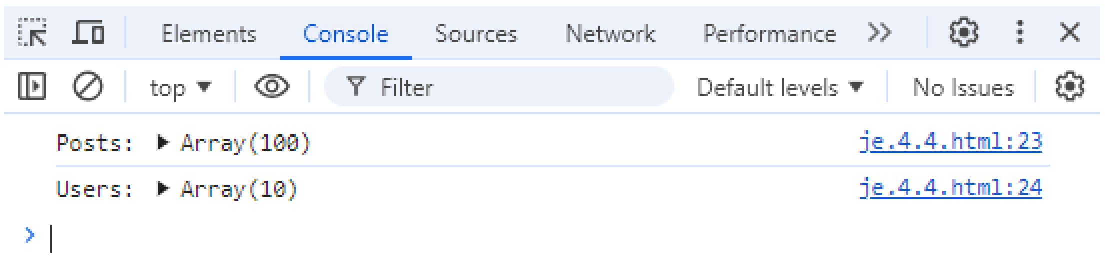


```
Failure! The number is 0.5 or less                    je4.3.html:22
```

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 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:

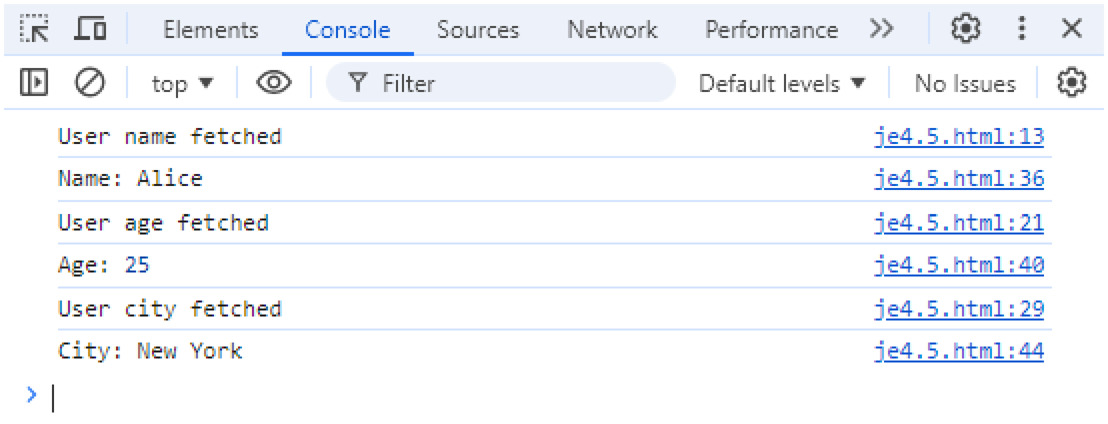


```
Posts:  ▶ Array(100)                                    je.4.4.html:23
Users:  ▶ Array(10)                                     je.4.4.html:24
```

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 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                                je4.5.html:13
Name: Alice                                      je4.5.html:36
User age fetched                                 je4.5.html:21
Age: 25                                          je4.5.html:40
User city fetched                                je4.5.html:29
City: New York                                   je4.5.html:44
```

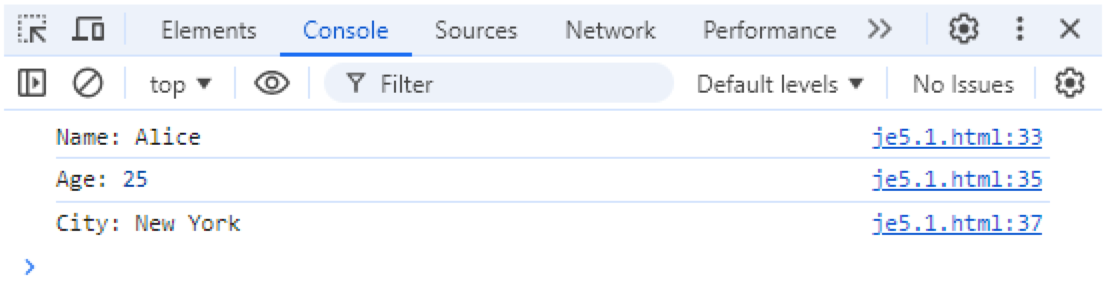
Async/await:

Task 1:

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

```html
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Document</title>
  </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:

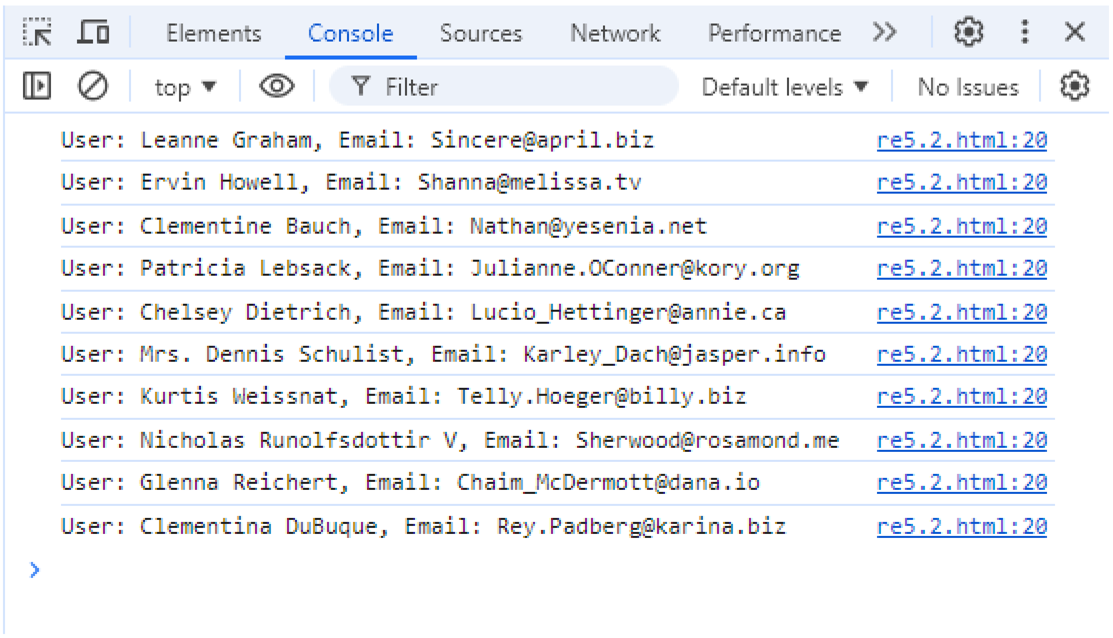


```
Name: Alice                                   je5.1.html:33
Age: 25                                       je5.1.html:35
City: New York                                je5.1.html:37
```

Task 2:

```html
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Document</title>
  </head>
  <body>
    <script>
      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:

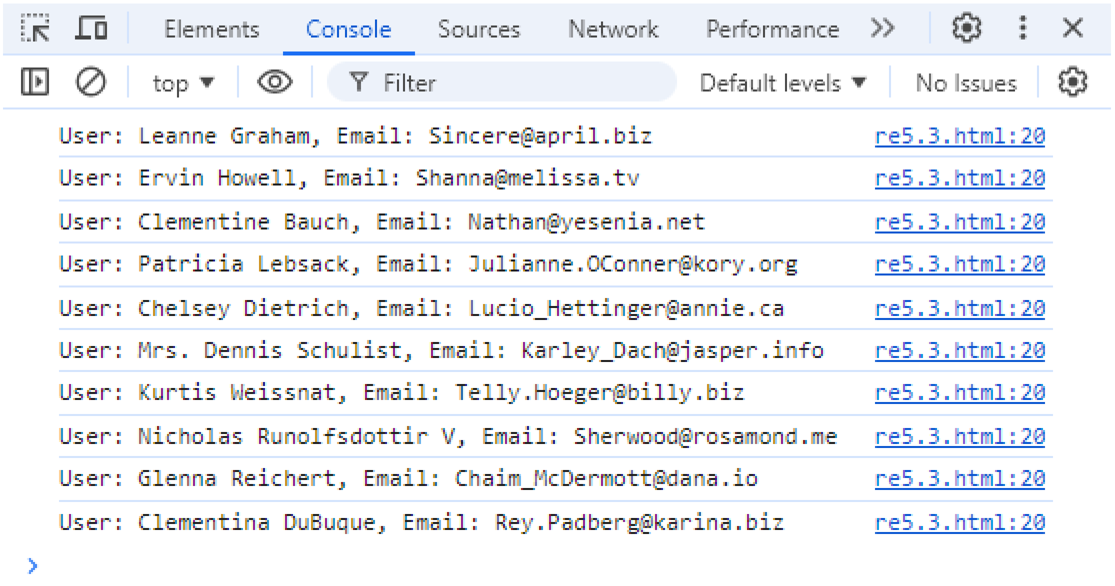


```
User: Leanne Graham, Email: Sincere@april.biz                      re5.2.html:20
User: Ervin Howell, Email: Shanna@melissa.tv                       re5.2.html:20
User: Clementine Bauch, Email: Nathan@yesenia.net                  re5.2.html:20
User: Patricia Lebsack, Email: Julianne.OConner@kory.org           re5.2.html:20
User: Chelsey Dietrich, Email: Lucio_Hettinger@annie.ca            re5.2.html:20
User: Mrs. Dennis Schulist, Email: Karley_Dach@jasper.info         re5.2.html:20
User: Kurtis Weissnat, Email: Telly.Hoeger@billy.biz               re5.2.html:20
User: Nicholas Runolfsdottir V, Email: Sherwood@rosamond.me        re5.2.html:20
User: Glenna Reichert, Email: Chaim_McDermott@dana.io              re5.2.html:20
User: Clementina DuBuque, Email: Rey.Padberg@karina.biz            re5.2.html:20
>
```

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>
      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:

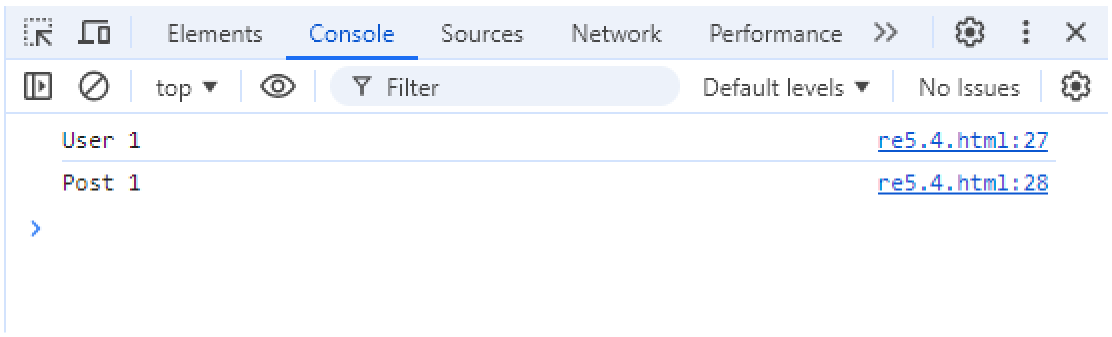


```
User: Leanne Graham, Email: Sincere@april.biz                    re5.3.html:20
User: Ervin Howell, Email: Shanna@melissa.tv                     re5.3.html:20
User: Clementine Bauch, Email: Nathan@yesenia.net                re5.3.html:20
User: Patricia Lebsack, Email: Julianne.OConner@kory.org         re5.3.html:20
User: Chelsey Dietrich, Email: Lucio_Hettinger@annie.ca          re5.3.html:20
User: Mrs. Dennis Schulist, Email: Karley_Dach@jasper.info       re5.3.html:20
User: Kurtis Weissnat, Email: Telly.Hoeger@billy.biz             re5.3.html:20
User: Nicholas Runolfsdottir V, Email: Sherwood@rosamond.me      re5.3.html:20
User: Glenna Reichert, Email: Chaim_McDermott@dana.io            re5.3.html:20
User: Clementina DuBuque, Email: Rey.Padberg@karina.biz          re5.3.html:20
```

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 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:

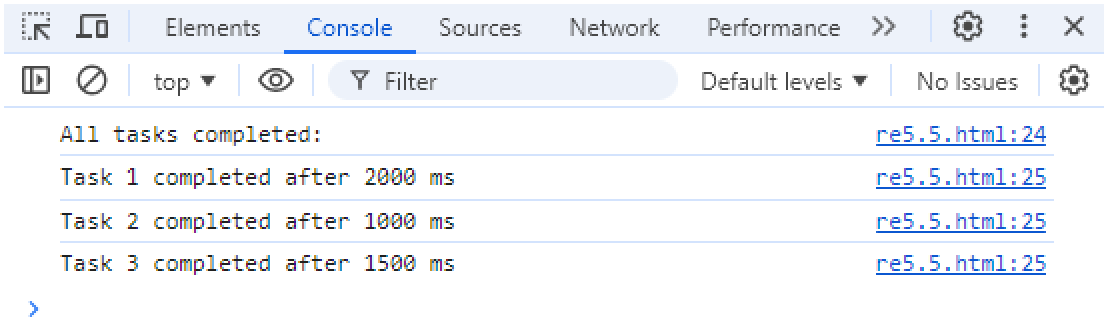


```
User 1                                                re5.4.html:27
Post 1                                                re5.4.html:28
```

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 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:                        re5.5.html:24
Task 1 completed after 2000 ms              re5.5.html:25
Task 2 completed after 1000 ms              re5.5.html:25
Task 3 completed after 1500 ms              re5.5.html:25
```

Modules introduction, Export and Import

Task 1:

Index.html

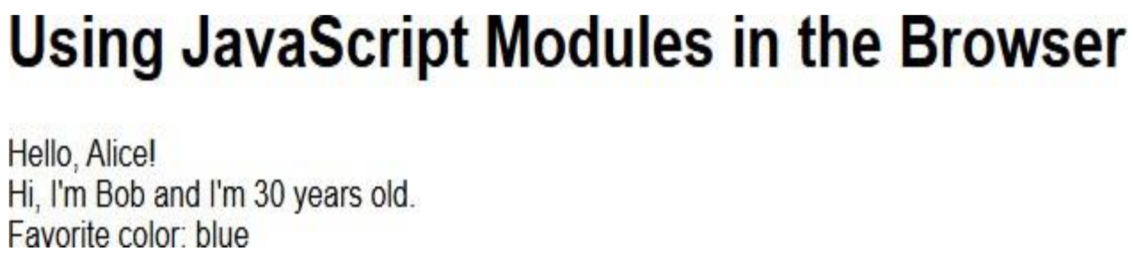
```
<!DOCTYPE html>
<html lang="en">
  <head>
```

```
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Document</title>
  </head>
  <body>
    <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 './re6.modulus.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:

Index.html

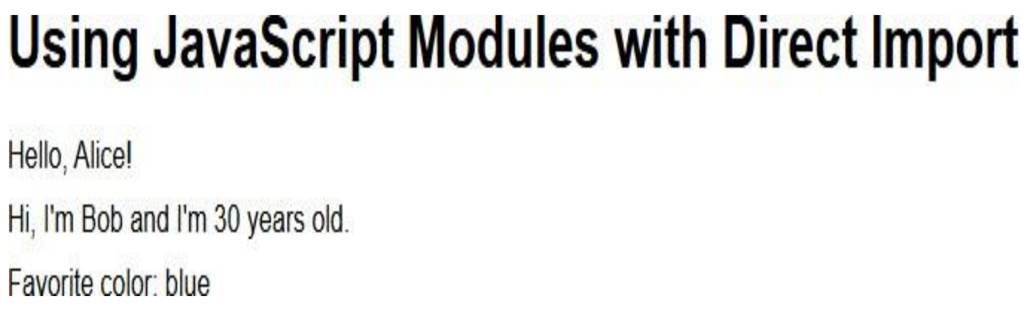
```html
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>JavaScript Modules Example</title>
    <style>
      body {
        font-family: Arial, sans-serif;
        margin: 20px;
      }
   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

```javascript
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:

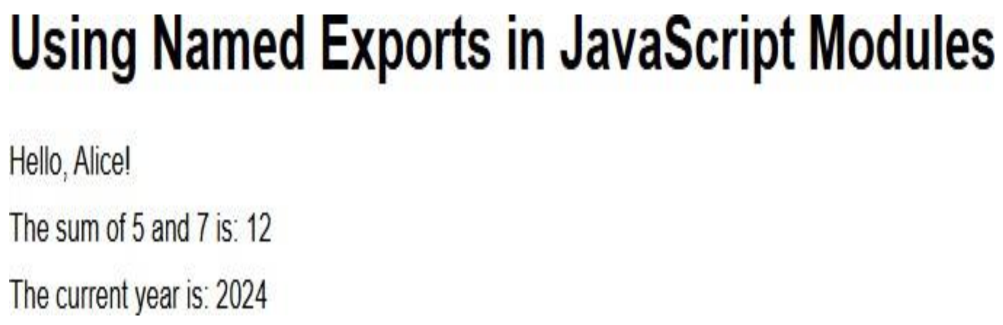
Index.html

```
<!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 "./re6.3.module.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:

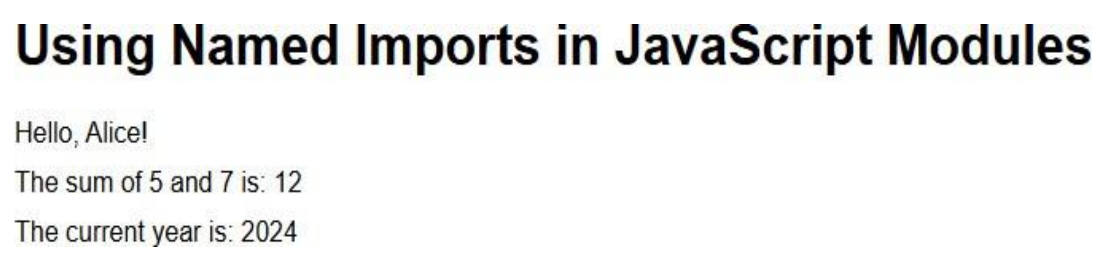
Index.html

```
<!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 "./re6.4.module.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;
}
export function subtract(a, b) {
  return a - b;
}
export function getCurrentYear() {
  return new Date().getFullYear();
}
```

Output:



```
Using Named Imports in JavaScript Modules

Hello, Alice!

The sum of 5 and 7 is: 12

The current year is: 2024
```

Task 5:

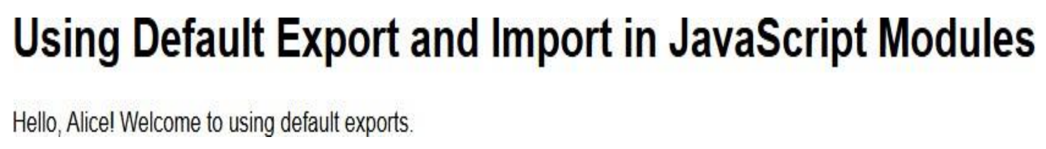
```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Using Default Export and Import</title>
<style> body {
font-family: Arial, sans-serif; margin: 20px;
}
#greeting { margin: 10px 0;
}
</style>
</head>
<body>
<h1>Using Default Export and Import in JavaScript Modules</h1>
<div id="greeting"></div>
<script type="module">
import greet from './re6.5.module.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.

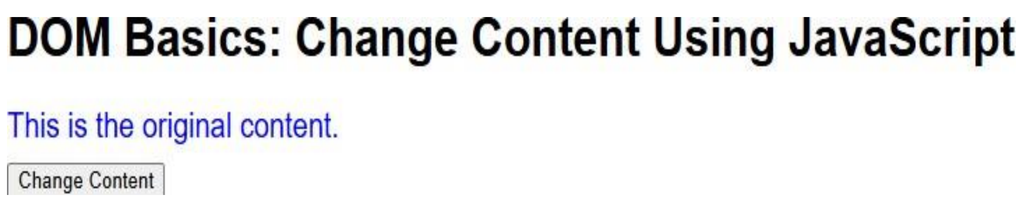
Browser: DOM Basics:

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>
```

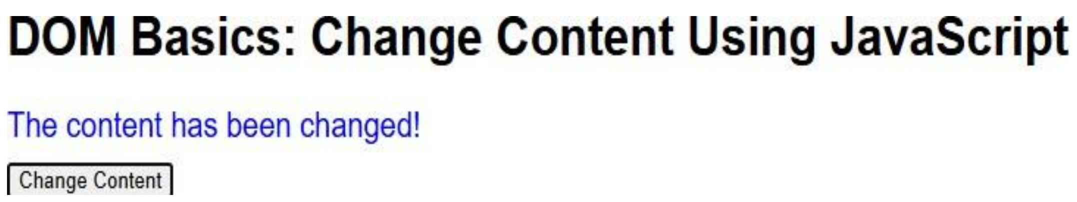
```
      function changeContent() {
        var element = document.getElementById("message");
        element.textContent = "The content has been changed!";
      }
    </script>
  </body>
</html>
```

Output:



DOM Basics: Change Content Using JavaScript

This is the original content.



DOM Basics: Change Content Using JavaScript

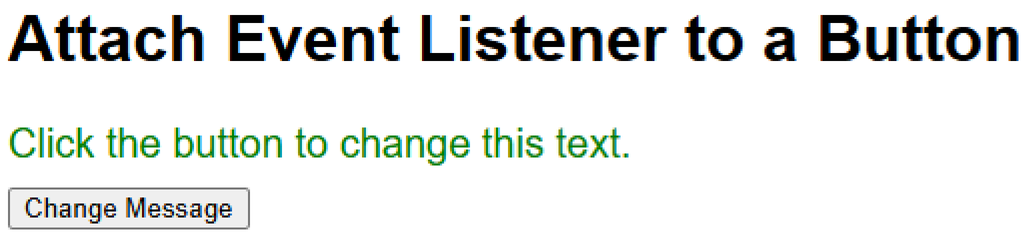
The content has been changed!

Task 2:

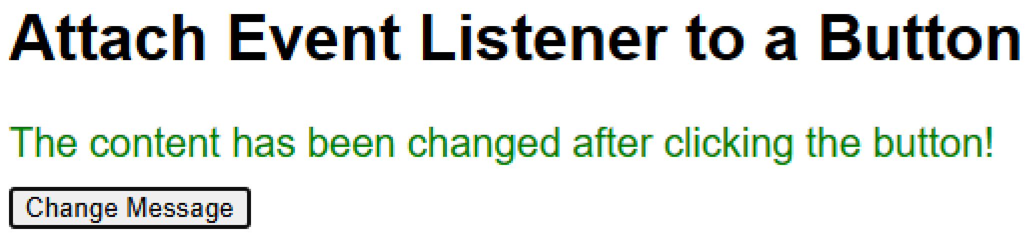
```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Button Event Listener Example</title>
<style> body {
font-family: Arial, sans-serif; margin: 20px;
}
#message {
font-size: 20px; color: green; margin: 10px 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.

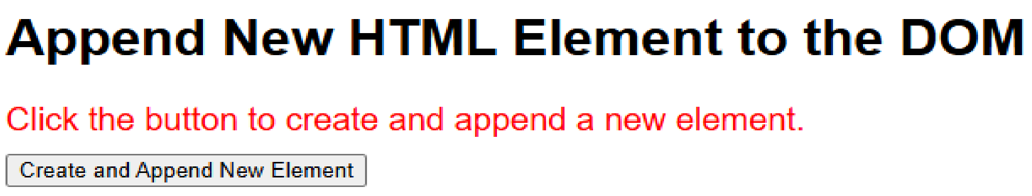

Attach Event Listener to a Button

The content has been changed after clicking the button!

Task 3:

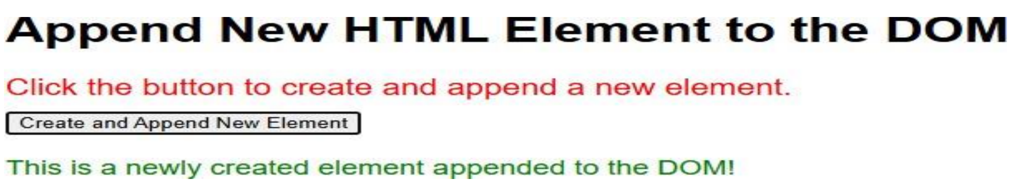
```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Create and Append a New HTML Element</title>
<style> body {
font-family: Arial, sans-serif;
margin: 20px;
}
#message {
font-size: 20px; color: red; margin: 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

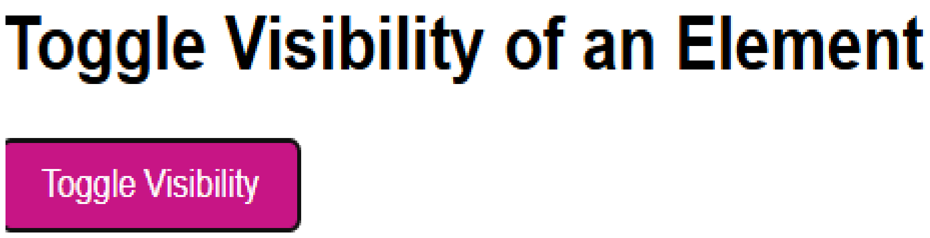
Click the button to create and append a new element.

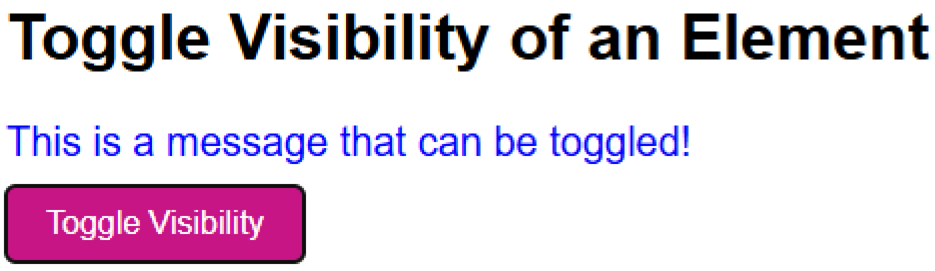
This is a newly created element appended to the DOM!

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:

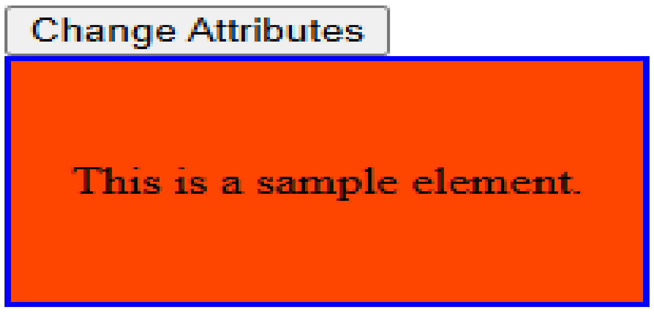



Toggle Visibility of an Element
This is a message that can be toggled!
Toggle Visibility

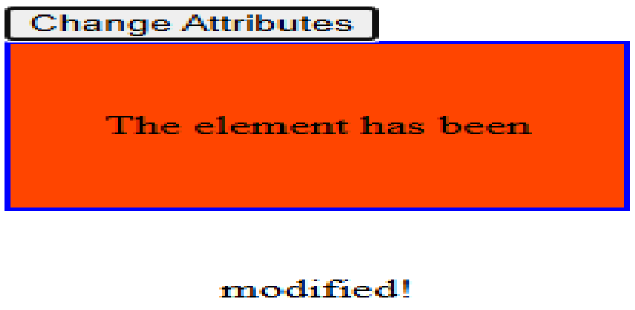
Task 5:

```
<!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!