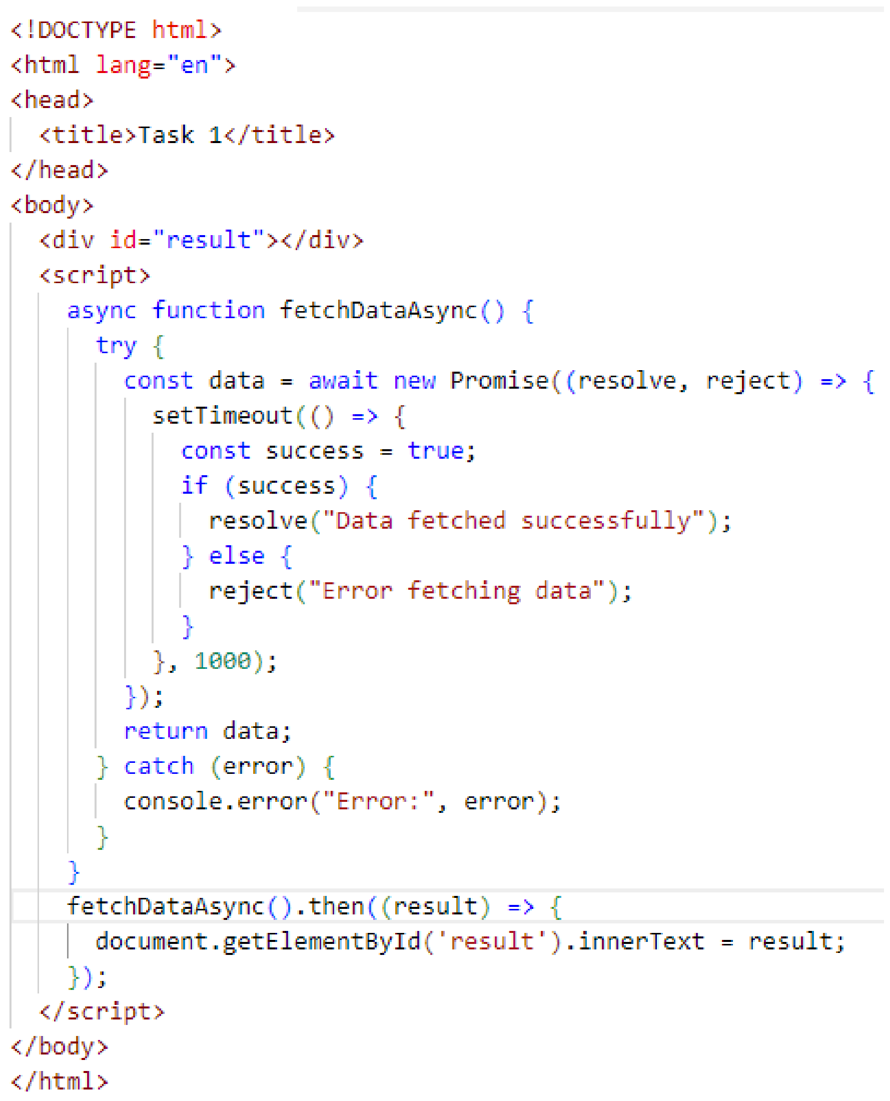
5. Async/await

Task 1:

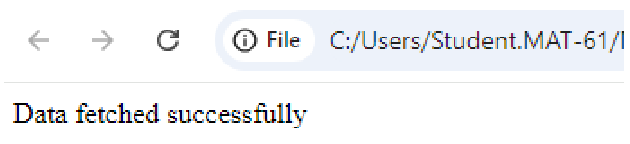
Rewrite a promise-based function using async/await.

Code:



```
<!DOCTYPE html>
<html lang="en">
<head>
  <title>Task 1</title>
</head>
<body>
  <div id="result"></div>
  <script>
    async function fetchDataAsync() {
      try {
        const data = await new Promise((resolve, reject) => {
          setTimeout(() => {
            const success = true;
            if (success) {
              resolve("Data fetched successfully");
            } else {
              reject("Error fetching data");
            }
          }, 1000);
        });
        return data;
      } catch (error) {
        console.error("Error:", error);
      }
    }
    fetchDataAsync().then((result) => {
      document.getElementById('result').innerText = result;
    });
  </script>
</body>
</html>
```

Output:

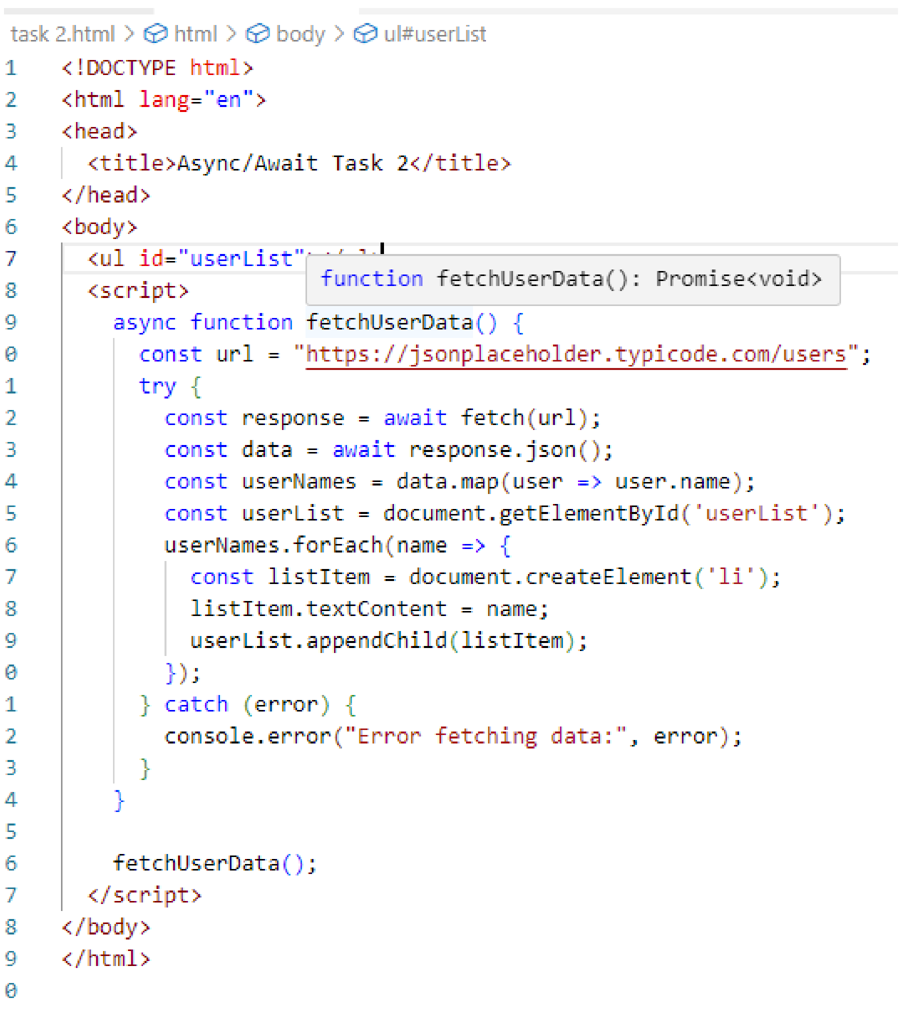


```
Data fetched successfully
```

Task 2:

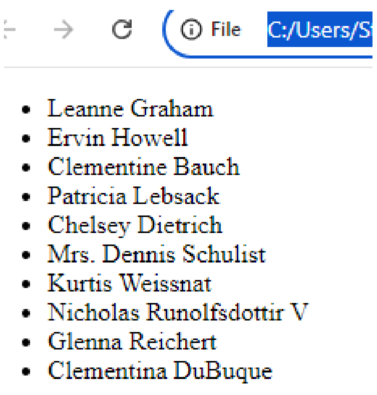
Create an async function that fetches data from an API and processes it.

Code:


task 2.html > html > body > ul#userList
function fetchUserData(): Promise<void>

```html
<!DOCTYPE html>
<html lang="en">
<head>
  <title>Async/Await Task 2</title>
</head>
<body>
  <ul id="userList"></ul>
  <script>
    async function fetchUserData() {
      const url = "https://jsonplaceholder.typicode.com/users";
      try {
        const response = await fetch(url);
        const data = await response.json();
        const userNames = data.map(user => user.name);
        const userList = document.getElementById('userList');
        userNames.forEach(name => {
          const listItem = document.createElement('li');
          listItem.textContent = name;
          userList.appendChild(listItem);
        });
      } catch (error) {
        console.error("Error fetching data:", error);
      }
    }

    fetchUserData();
  </script>
</body>
</html>
```

Output:

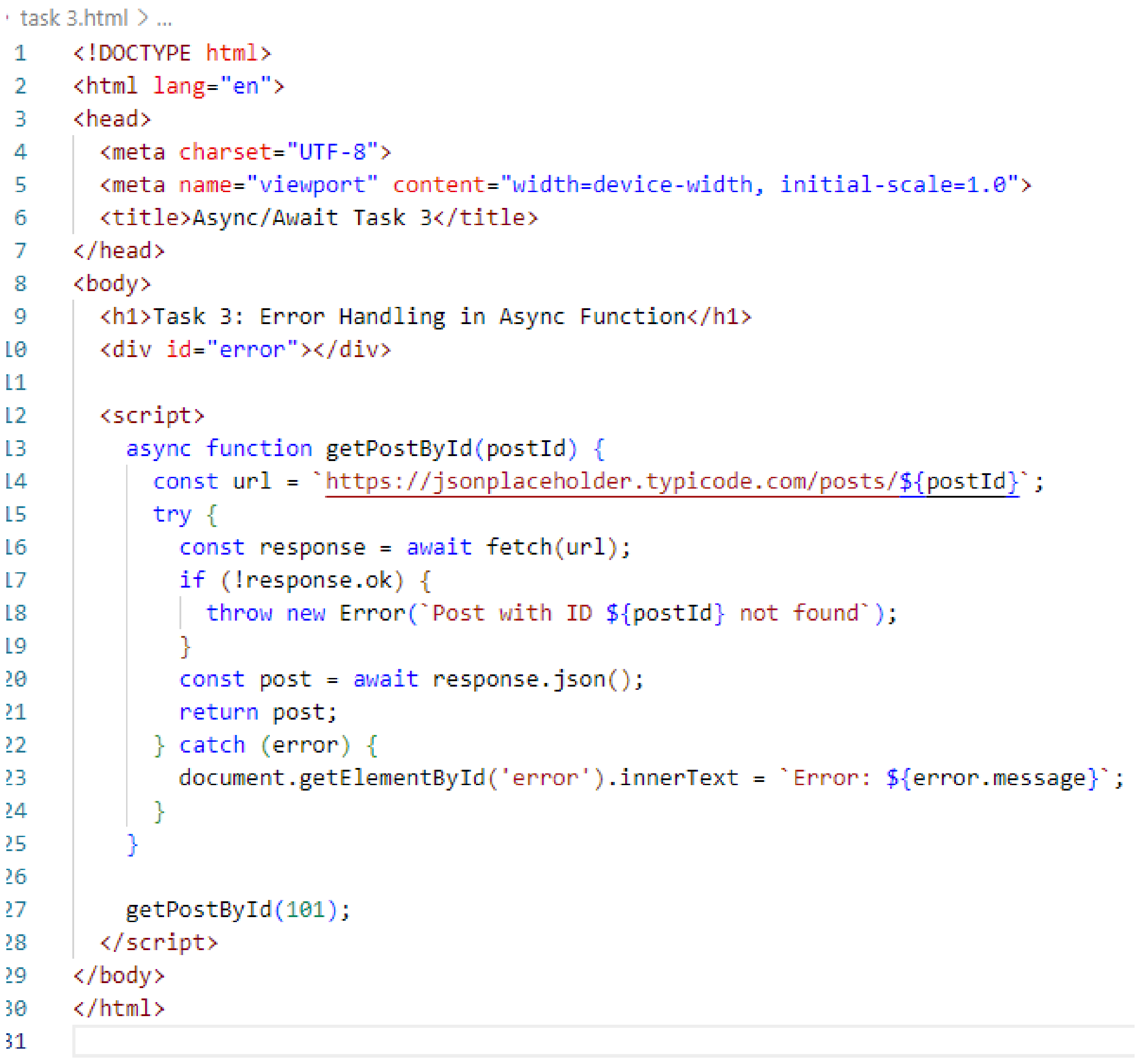


```
• Leanne Graham
• Ervin Howell
• Clementine Bauch
• Patricia Lebsack
• Chelsey Dietrich
• Mrs. Dennis Schulist
• Kurtis Weissnat
• Nicholas Runolfsdottir V
• Glenna Reichert
• Clementina DuBuque
```

Task 3:

Implement error handling in an async function using try/catch.

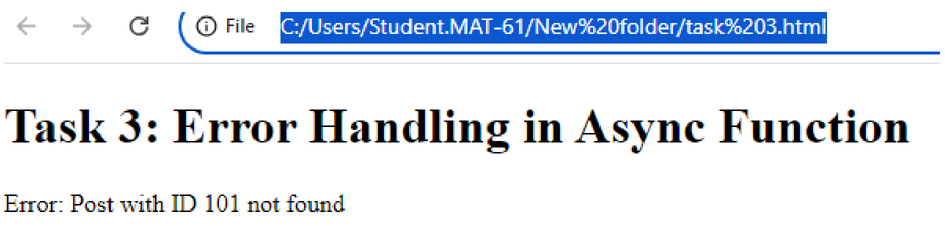
Code:



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>Async/Await Task 3</title>
</head>
<body>
  <h1>Task 3: Error Handling in Async Function</h1>
  <div id="error"></div>

  <script>
    async function getPostById(postId) {
      const url = `https://jsonplaceholder.typicode.com/posts/${postId}`;
      try {
        const response = await fetch(url);
        if (!response.ok) {
          throw new Error(`Post with ID ${postId} not found`);
        }
        const post = await response.json();
        return post;
      } catch (error) {
        document.getElementById('error').innerText = `Error: ${error.message}`;
      }
    }

    getPostById(101);
  </script>
</body>
</html>
```

Output:


C:/Users/Student.MAT-61/New%20folder/task%203.html

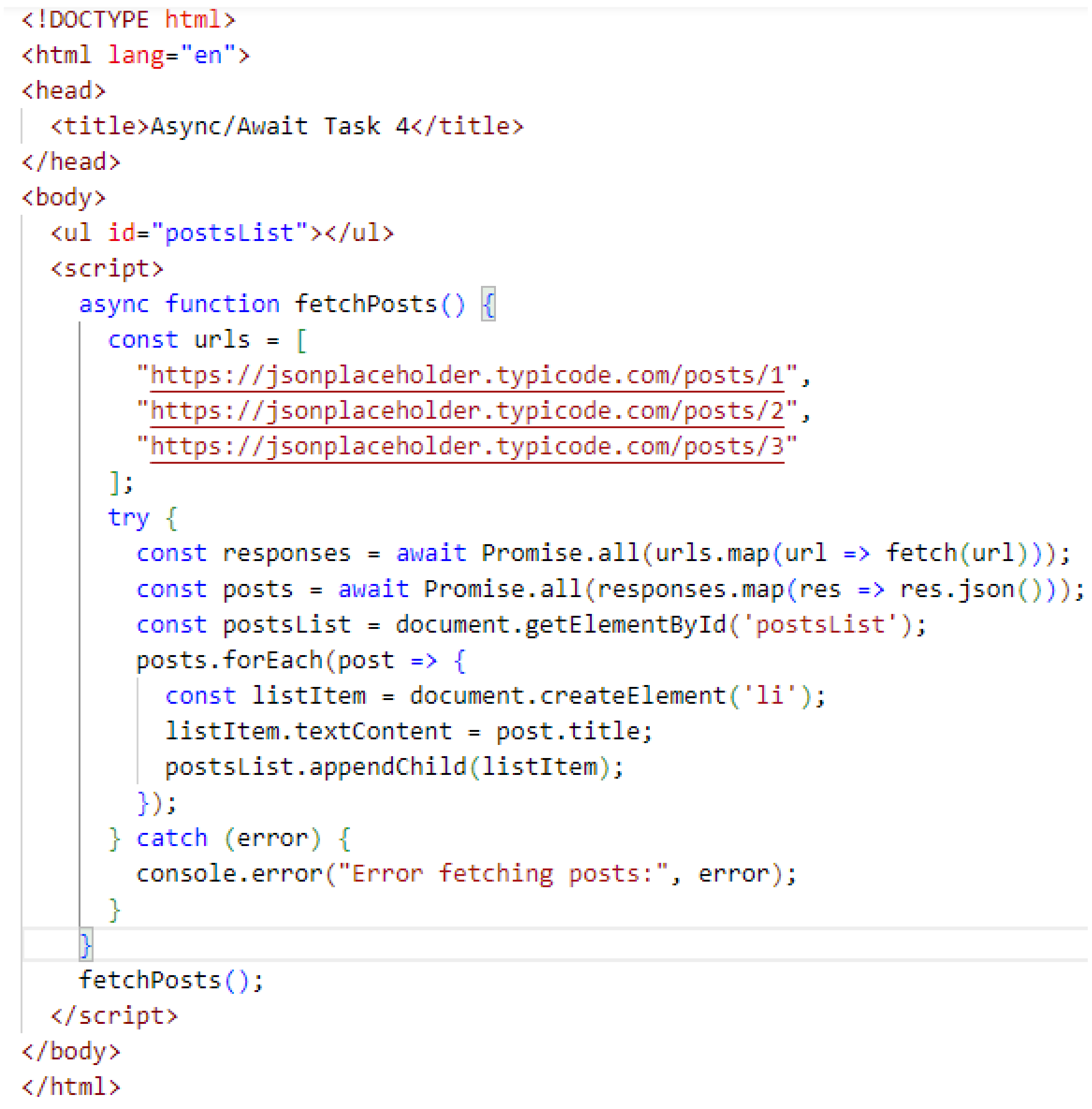
Task 3: Error Handling in Async Function

```
Error: Post with ID 101 not found
```

Task 4:

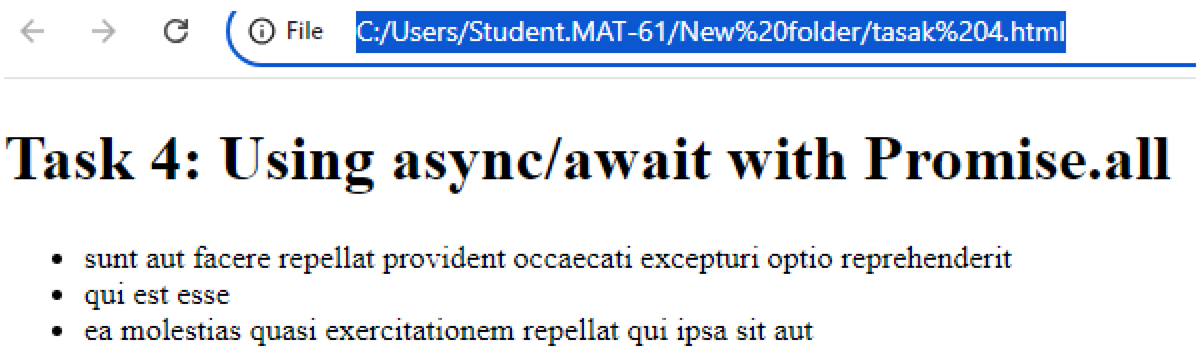
Use async/await in combination with Promise all.

Code:



```
<!DOCTYPE html>
<html lang="en">
<head>
  <title>Async/Await Task 4</title>
</head>
<body>
  <ul id="postsList"></ul>
  <script>
    async function fetchPosts() {
      const urls = [
        "https://jsonplaceholder.typicode.com/posts/1",
        "https://jsonplaceholder.typicode.com/posts/2",
        "https://jsonplaceholder.typicode.com/posts/3"
      ];
      try {
        const responses = await Promise.all(urls.map(url => fetch(url)));
        const posts = await Promise.all(responses.map(res => res.json()));
        const postsList = document.getElementById('postsList');
        posts.forEach(post => {
          const listItem = document.createElement('li');
          listItem.textContent = post.title;
          postsList.appendChild(listItem);
        });
      } catch (error) {
        console.error("Error fetching posts:", error);
      }
    }
    fetchPosts();
  </script>
</body>
</html>
```

Output:



File C:/Users/Student.MAT-61/New%20folder/tasak%204.html

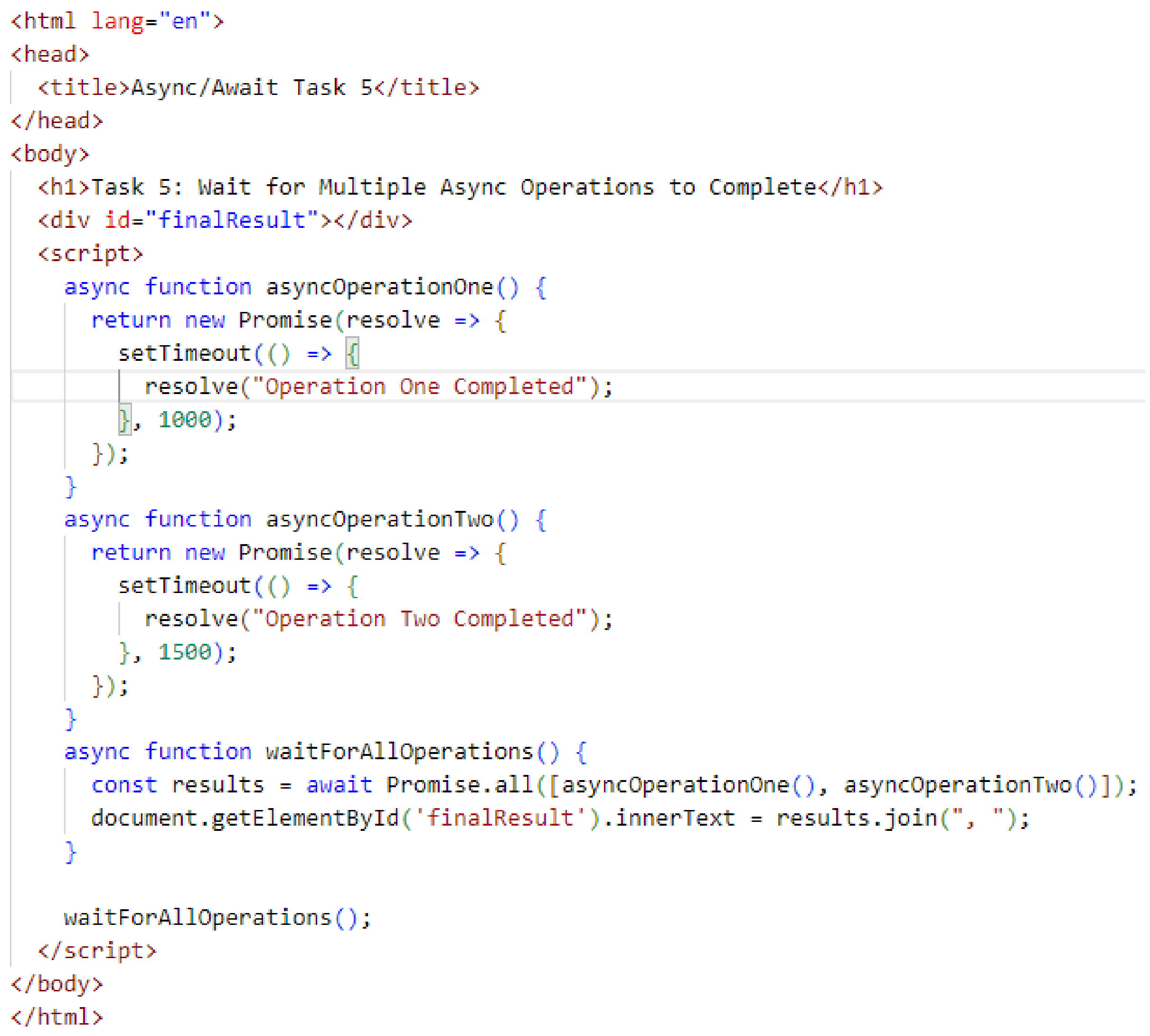
Task 4: Using async/await with Promise.all

- sunt aut facere repellat provident occaecati excepturi optio reprehenderit
- qui est esse
- ea molestias quasi exercitationem repellat qui ipsa sit aut

Task 5:

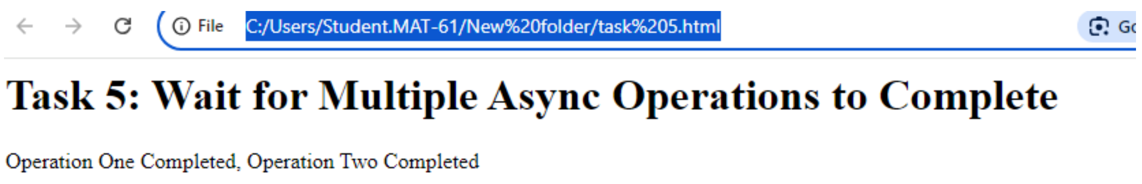
Create an async function that waits for multiple asynchronous operations to complete before proceeding.

Code:



```
<html lang="en">
<head>
  <title>Async/Await Task 5</title>
</head>
<body>
  <h1>Task 5: Wait for Multiple Async Operations to Complete</h1>
  <div id="finalResult"></div>
  <script>
    async function asyncOperationOne() {
      return new Promise(resolve => {
        setTimeout(() => {
          resolve("Operation One Completed");
        }, 1000);
      });
    }
    async function asyncOperationTwo() {
      return new Promise(resolve => {
        setTimeout(() => {
          resolve("Operation Two Completed");
        }, 1500);
      });
    }
    async function waitForAllOperations() {
      const results = await Promise.all([asyncOperationOne(), asyncOperationTwo()]);
      document.getElementById('finalResult').innerText = results.join(", ");
    }

    waitForAllOperations();
  </script>
</body>
</html>
```

Output:



Task 5: Wait for Multiple Async Operations to Complete

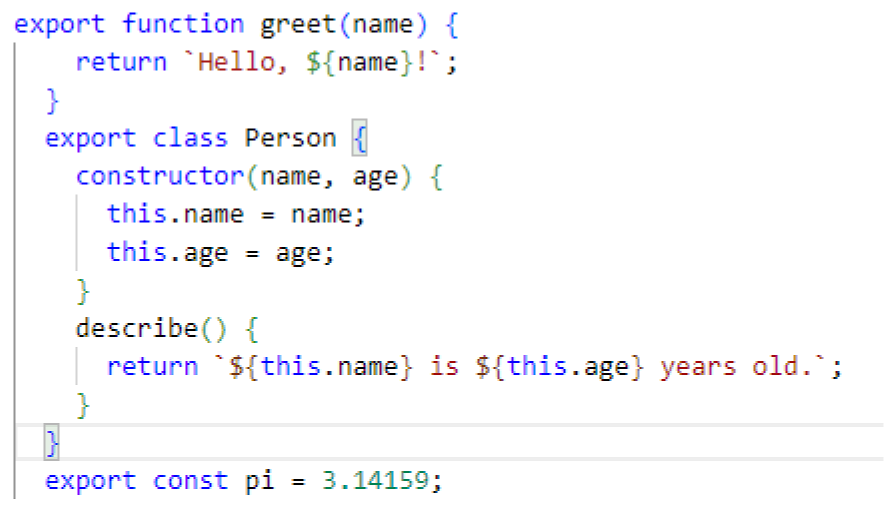
Operation One Completed, Operation Two Completed

6. Modules introduction, Export and Import

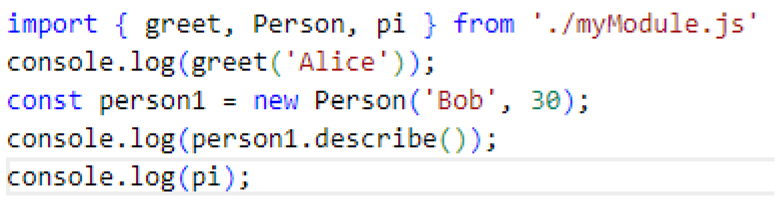
Task 1:

Create a module that exports a function, a class, and a variable.

Code:



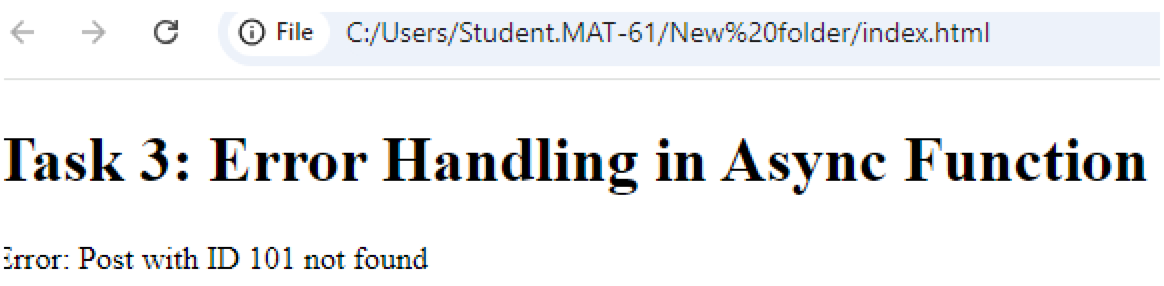
```
export function greet(name) {
  return `Hello, ${name}!`;
 }
 export class Person {
  constructor(name, age) {
   this.name = name;
   this.age = age;
  }
  describe() {
   return `${this.name} is ${this.age} years old.`;
  }
 }
 export const pi = 3.14159;
```



```
import { greet, Person, pi } from './myModule.js'
console.log(greet('Alice'));
const person1 = new Person('Bob', 30);
console.log(person1.describe());
console.log(pi);
```

```
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Async/Await Task 3</title>
</head>
<body>
  <h1>Task 3: Error Handling in Async Function</h1>
  <div id="error"></div>

  <script>
    async function getPostById(postId) {
      const url = `https://jsonplaceholder.typicode.com/posts/${postId}`;
      try {
        const response = await fetch(url);
        if (!response.ok) {
          throw new Error(`Post with ID ${postId} not found`);
        }
        const post = await response.json();
        return post;
      } catch (error) {
        document.getElementById('error').innerText = `Error: ${error.message}`;
      }
    }

    getPostById(101);
  </script>
</body>
</html>
```

Output:



File C:/Users/Student.MAT-61/New%20folder/index.html

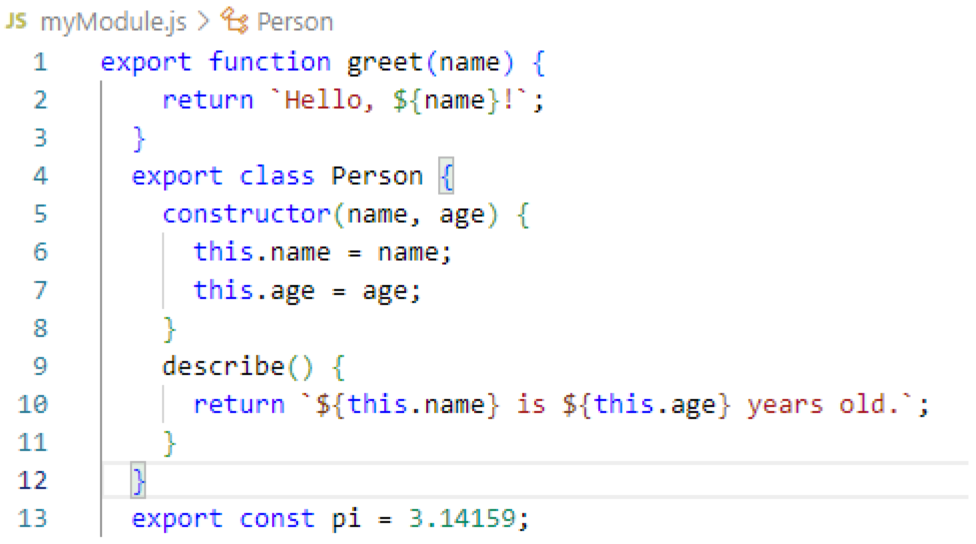
Task 3: Error Handling in Async Function

Error: Post with ID 101 not found

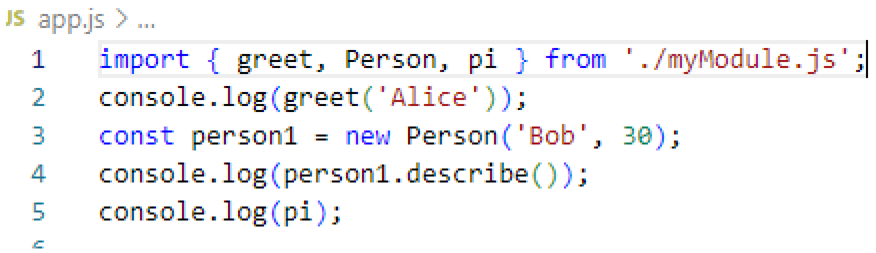
Task 2:

Import the module in another JavaScript file and use the exported entities.

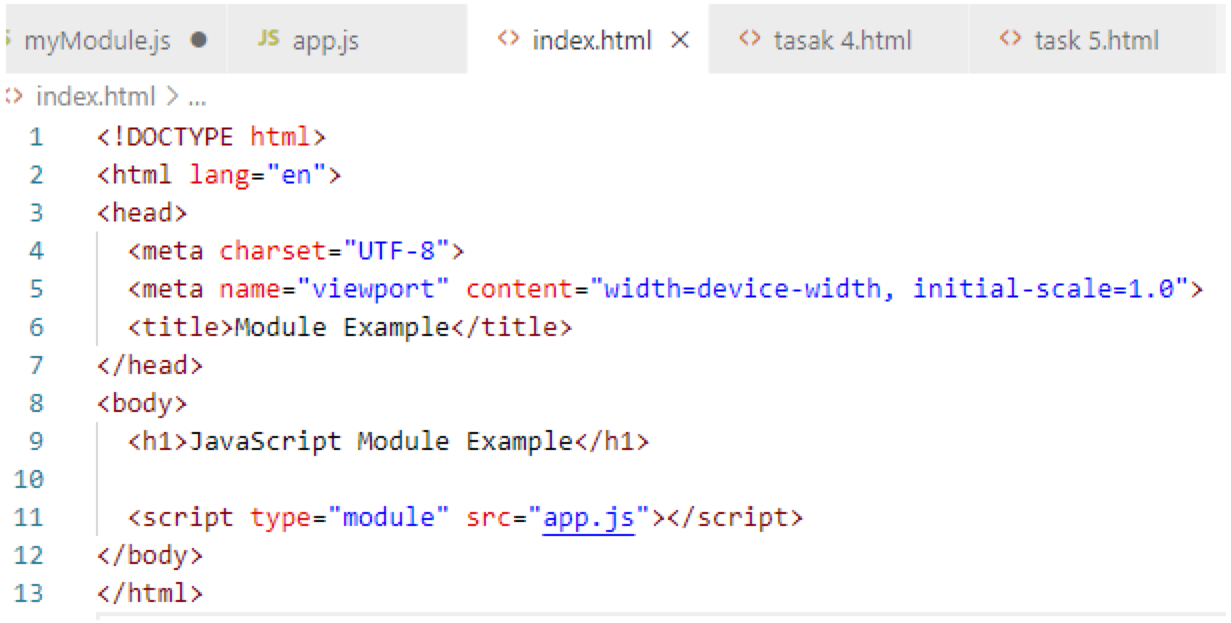
Code:


JS myModule.js > Person

```javascript
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 pi = 3.14159;
```


JS app.js > ...

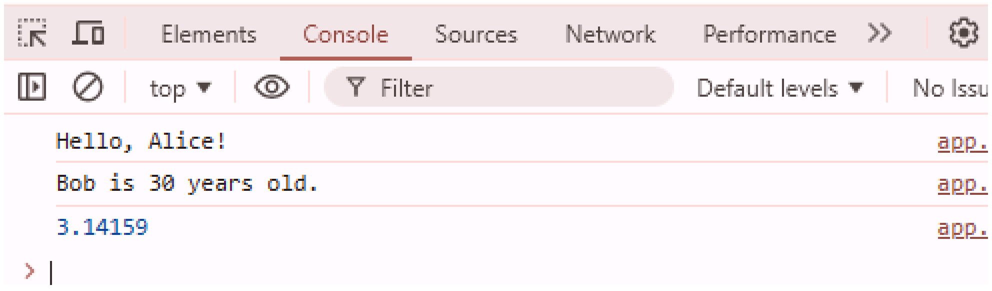
```javascript
import { greet, Person, pi } from './myModule.js';
console.log(greet('Alice'));
const person1 = new Person('Bob', 30);
console.log(person1.describe());
console.log(pi);
```


myModule.js ● | JS app.js | index.html × | tasak 4.html | task 5.html

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>Module Example</title>
</head>
<body>
  <h1>JavaScript Module Example</h1>

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

Output:



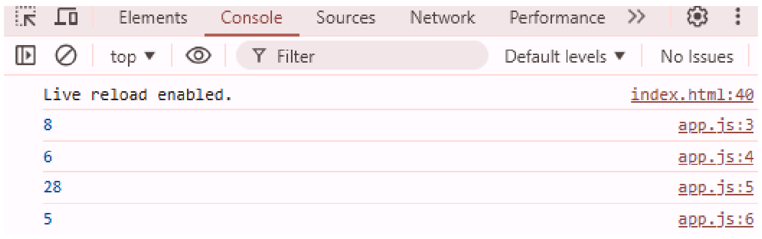
```
Hello, Alice!                                                    app.
Bob is 30 years old.                                             app.
3.14159                                                          app.
```

Task 3:

Use named exports to export multiple functions from a module.

Code:

Output:

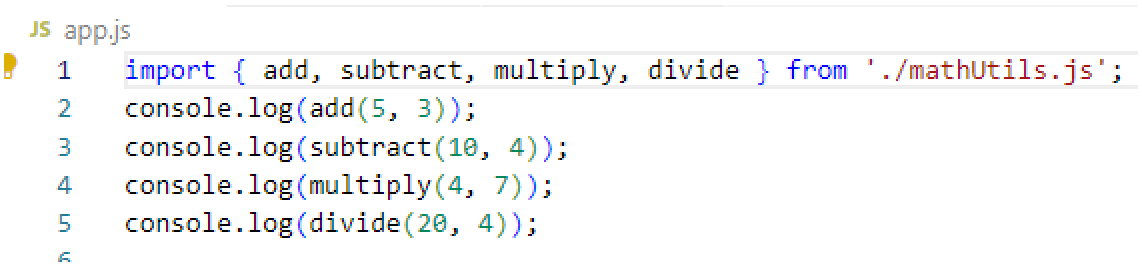


```
Live reload enabled.                                   index.html:40
8                                                          app.js:3
6                                                          app.js:4
28                                                         app.js:5
5                                                          app.js:6
```

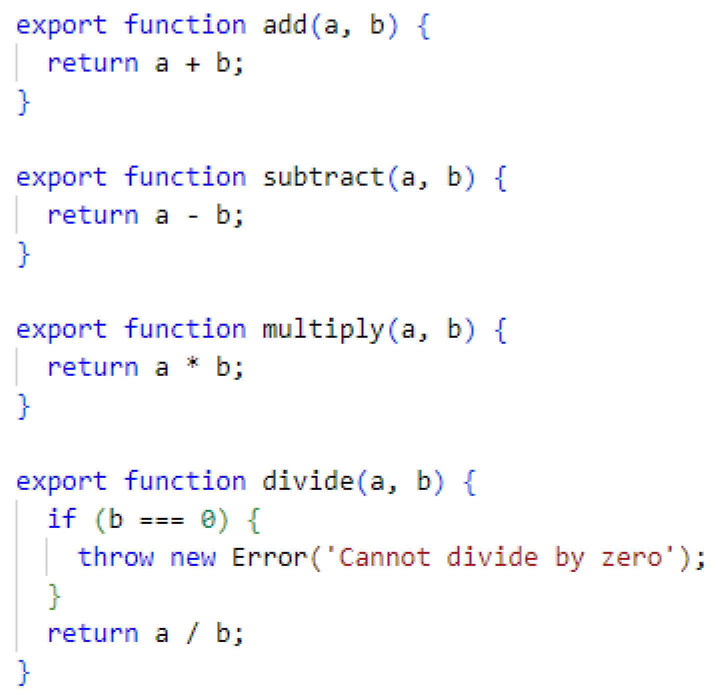
Task 4:

Use named imports to import specific functions from a module.

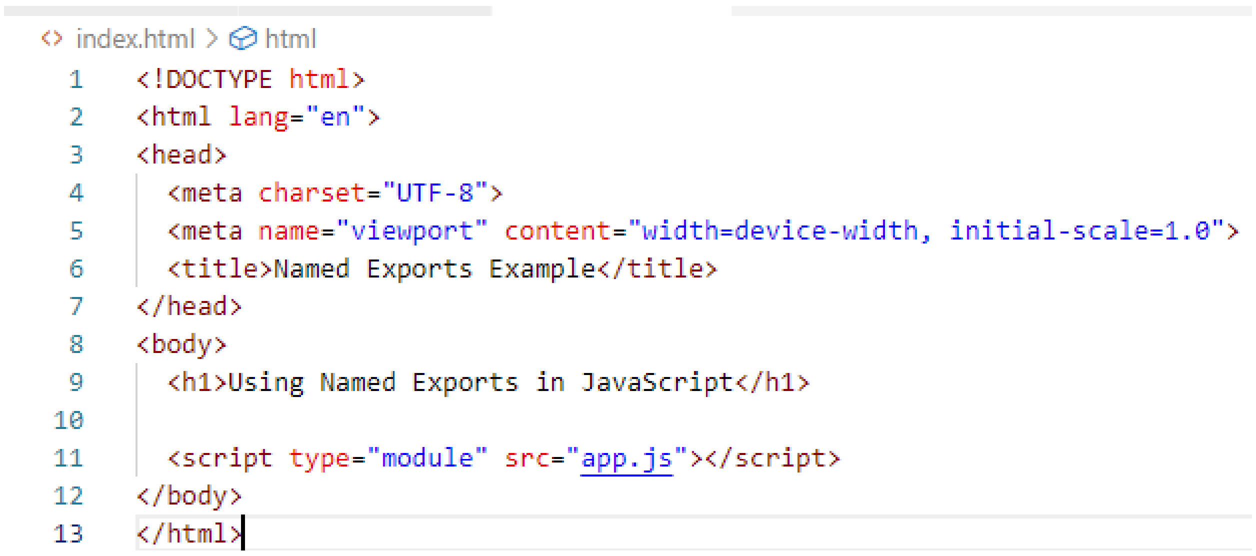
Code:



```
import { add, subtract, multiply, divide } from './mathUtils.js';
console.log(add(5, 3));
console.log(subtract(10, 4));
console.log(multiply(4, 7));
console.log(divide(20, 4));
```

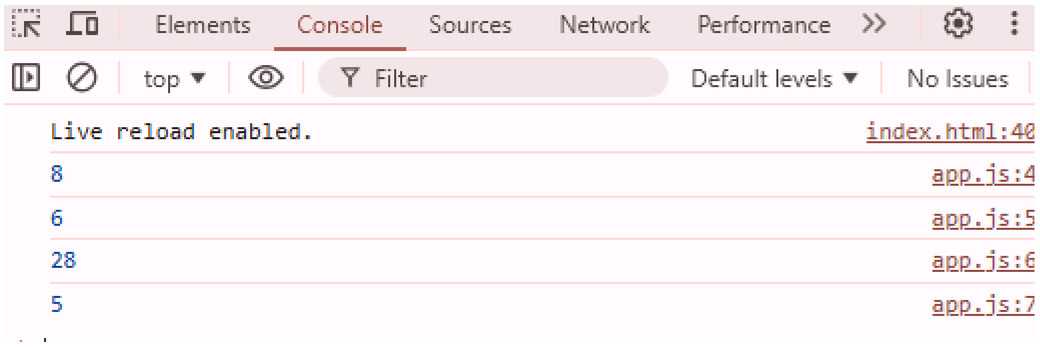


```
export function add(a, b) {
  return a + b;
}

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('Cannot divide by zero');
  }
  return a / b;
}
```



```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Named Exports Example</title>
</head>
<body>
  <h1>Using Named Exports in JavaScript</h1>

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

Output:

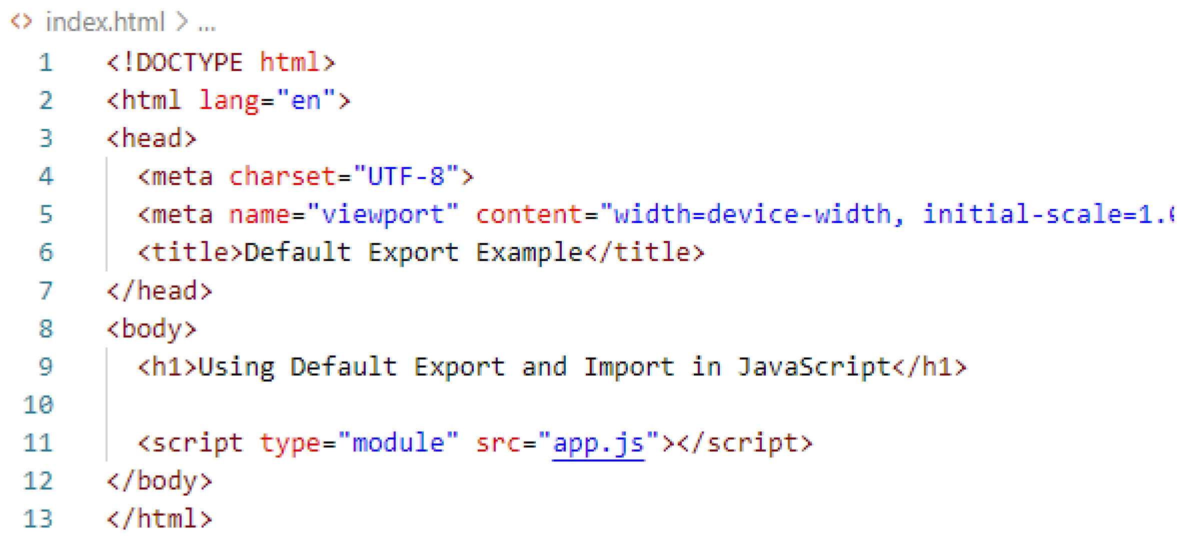


```
Live reload enabled.                                        index.html:40
8                                                                app.js:4
6                                                                app.js:5
28                                                               app.js:6
5                                                                app.js:7
```

Task 5:

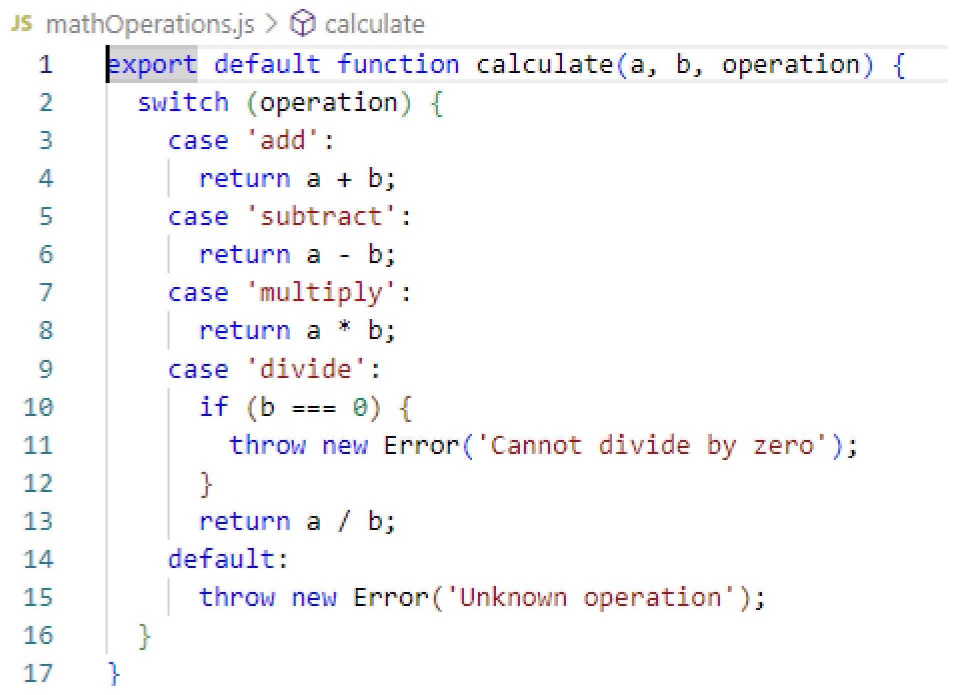
Use default export and import for a primary function of a module.

Code:



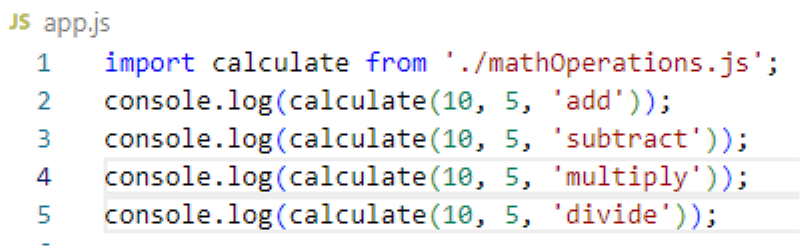
index.html > ...

```html
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.
  <title>Default Export Example</title>
</head>
<body>
  <h1>Using Default Export and Import in JavaScript</h1>

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



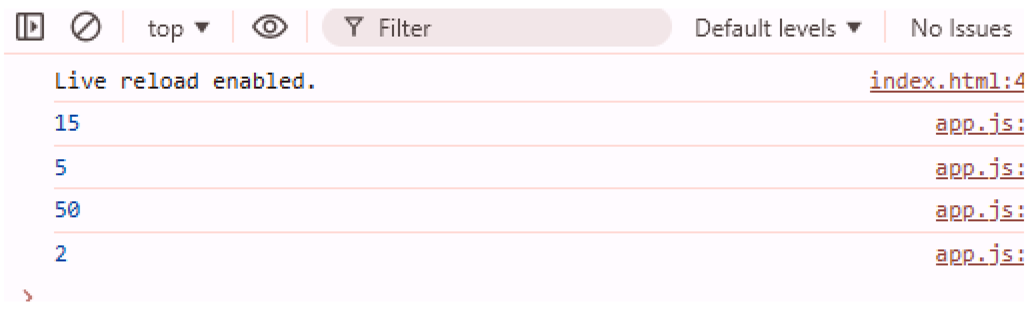
mathOperations.js > calculate

```javascript
export default function calculate(a, b, operation) {
  switch (operation) {
    case 'add':
      return a + b;
    case 'subtract':
      return a - b;
    case 'multiply':
      return a * b;
    case 'divide':
      if (b === 0) {
        throw new Error('Cannot divide by zero');
      }
      return a / b;
    default:
      throw new Error('Unknown operation');
  }
}
```



```
import calculate from './mathOperations.js';
console.log(calculate(10, 5, 'add'));
console.log(calculate(10, 5, 'subtract'));
console.log(calculate(10, 5, 'multiply'));
console.log(calculate(10, 5, 'divide'));
```

Output:



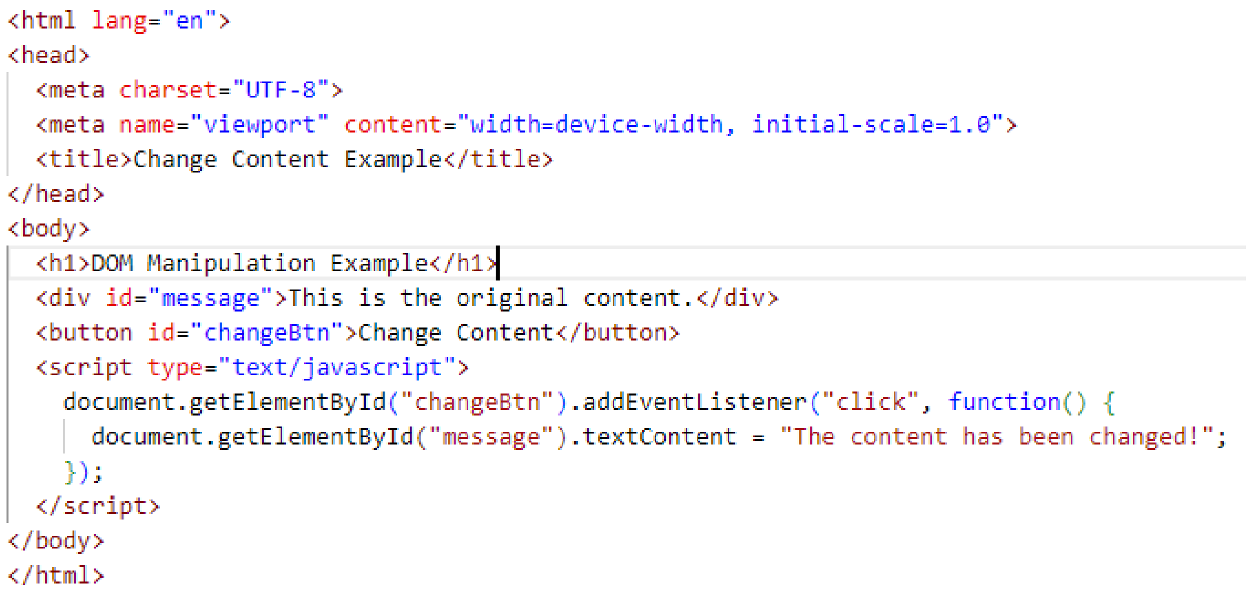
```
Live reload enabled.                                           index.html:4
15                                                                  app.js:
5                                                                   app.js:
50                                                                  app.js:
2                                                                   app.js:
>
```

7. Browser: DOM Basics

Task 1:

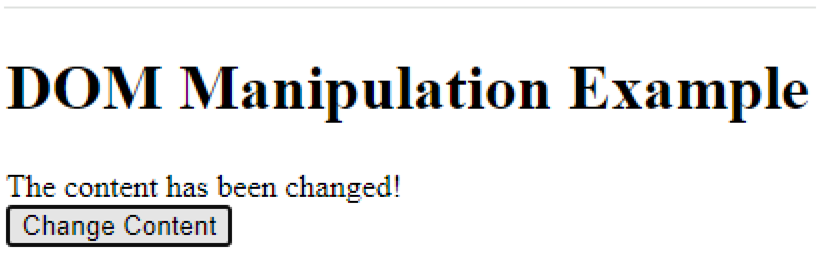
Select an HTML element by its ID and change its content using JavaScript.

Code:



```
<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>DOM Manipulation Example</h1>
  <div id="message">This is the original content.</div>
  <button id="changeBtn">Change Content</button>
  <script type="text/javascript">
    document.getElementById("changeBtn").addEventListener("click", function() {
      document.getElementById("message").textContent = "The content has been changed!";
    });
  </script>
</body>
</html>
```

Output:



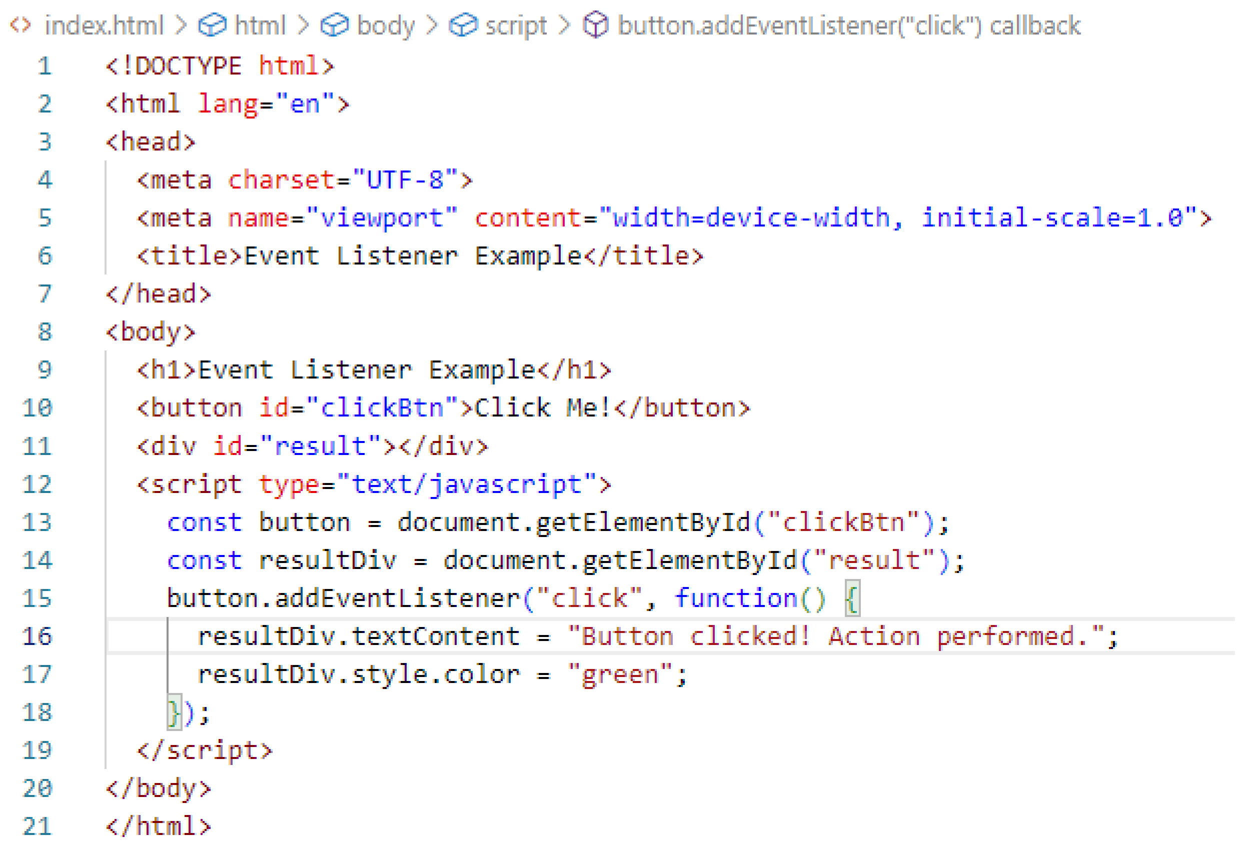
DOM Manipulation Example

The content has been changed!

Task 2:

Attach an event listener to a button, making it perform an action when clicked.

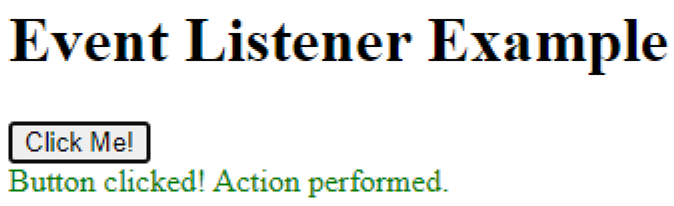
Code:



index.html > html > body > script > button.addEventListener("click") callback

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Event Listener Example</title>
</head>
<body>
  <h1>Event Listener Example</h1>
  <button id="clickBtn">Click Me!</button>
  <div id="result"></div>
  <script type="text/javascript">
    const button = document.getElementById("clickBtn");
    const resultDiv = document.getElementById("result");
    button.addEventListener("click", function() {
      resultDiv.textContent = "Button clicked! Action performed.";
      resultDiv.style.color = "green";
    });
  </script>
</body>
</html>
```

Output:



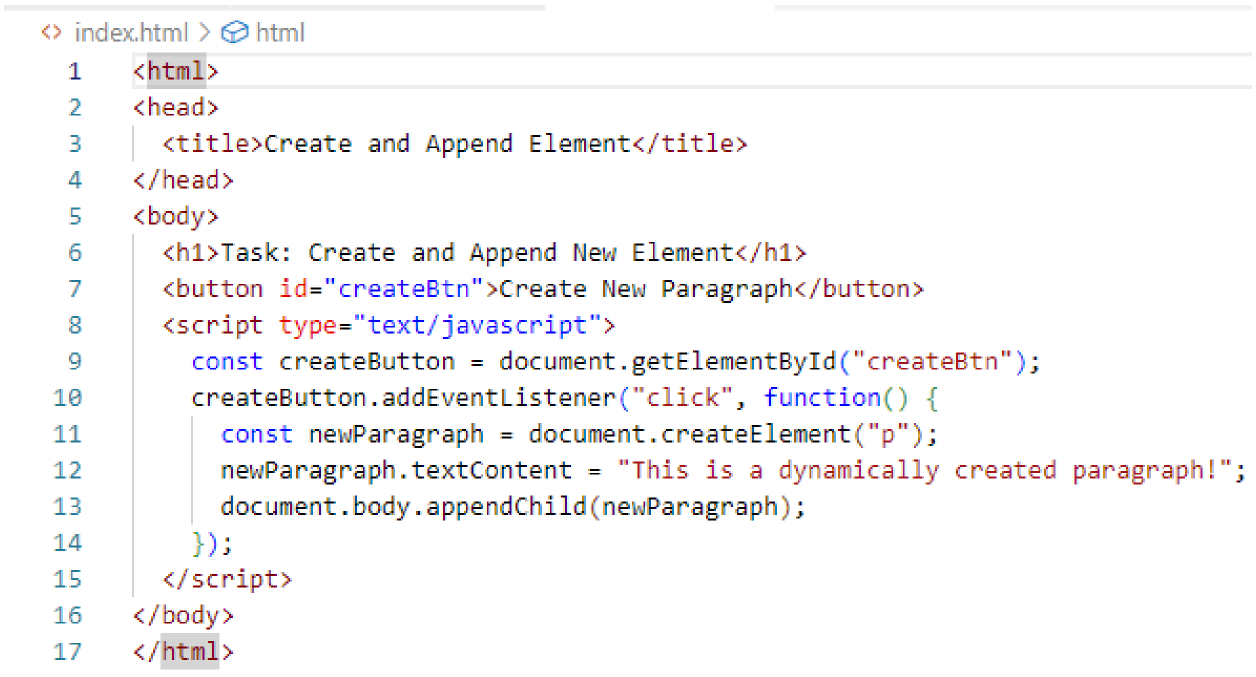
Event Listener Example

Button clicked! Action performed.

Task 3:

Create a new HTML element and append it to the DOM.

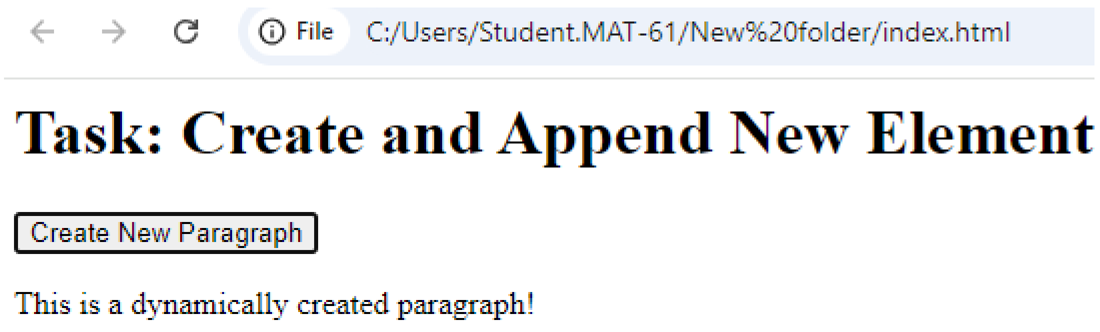
Code:



index.html > html

```
<html>
<head>
  <title>Create and Append Element</title>
</head>
<body>
  <h1>Task: Create and Append New Element</h1>
  <button id="createBtn">Create New Paragraph</button>
  <script type="text/javascript">
    const createButton = document.getElementById("createBtn");
    createButton.addEventListener("click", function() {
      const newParagraph = document.createElement("p");
      newParagraph.textContent = "This is a dynamically created paragraph!";
      document.body.appendChild(newParagraph);
    });
  </script>
</body>
</html>
```

Output:



File C:/Users/Student.MAT-61/New%20folder/index.html

Task: Create and Append New Element

Create New Paragraph

This is a dynamically created paragraph!

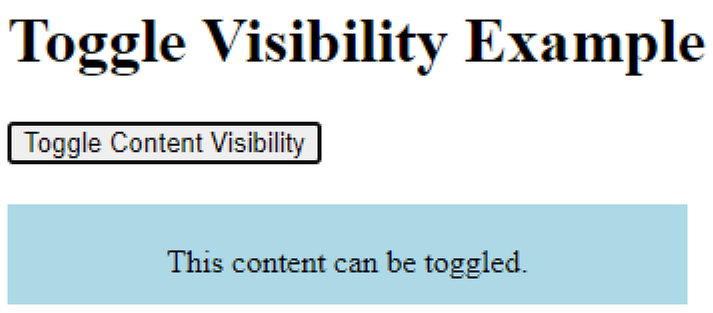
Task 4:

Implement a function to toggle the visibility of an element.

Code:

```
<!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 {
      width: 300px;
      height: 200px;
      background-color: lightblue;
      text-align: center;
      padding: 20px;
      margin-top: 20px;
      display: block;
    }
  </style>
</head>
<body>
  <h1>Toggle Visibility Example</h1>
  <button id="toggleBtn">Toggle Content Visibility</button>
  <div id="content">This content can be toggled.</div>
  <script type="text/javascript">
    function toggleVisibility() {
      const content = document.getElementById("content");
      if (content.style.display === "none") {
        content.style.display = "block";
      } else {
        content.style.display = "none";
      }
    }
    const toggleButton = document.getElementById("toggleBtn");
    toggleButton.addEventListener("click", toggleVisibility);
  </script>
</body>
</html>
```

Output:

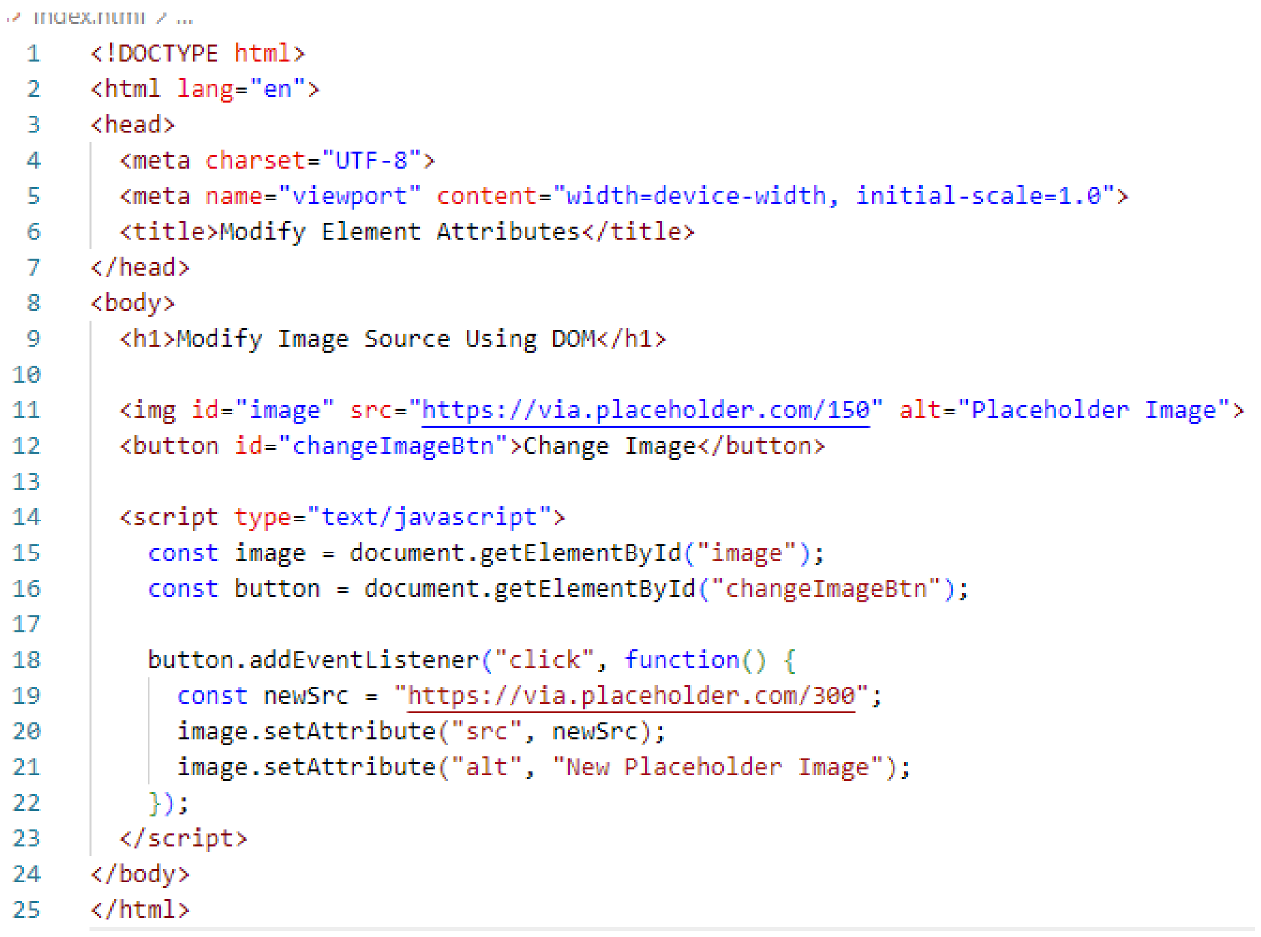

Toggle Visibility Example

This content can be toggled.

Task 5:

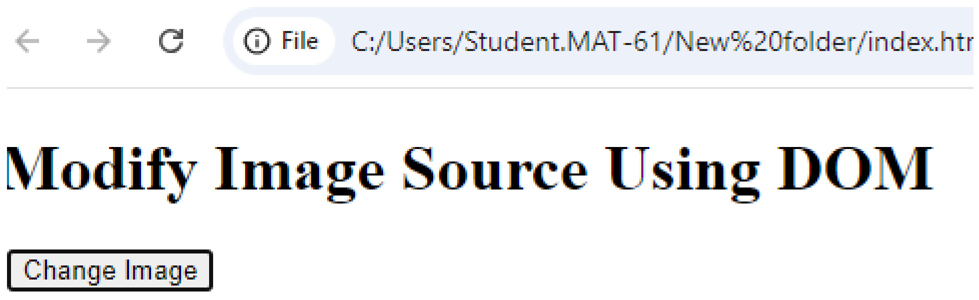
Use the DOM API to retrieve and modify the attributes of an element.

Code:



```html
<!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>
</head>
<body>
  <h1>Modify Image Source Using DOM</h1>

  <img id="image" src="https://via.placeholder.com/150" alt="Placeholder Image">
  <button id="changeImageBtn">Change Image</button>

  <script type="text/javascript">
    const image = document.getElementById("image");
    const button = document.getElementById("changeImageBtn");

    button.addEventListener("click", function() {
      const newSrc = "https://via.placeholder.com/300";
      image.setAttribute("src", newSrc);
      image.setAttribute("alt", "New Placeholder Image");
    });
  </script>
</body>
</html>
```

Output:



File C:/Users/Student.MAT-61/New%20folder/index.htr

Modify Image Source Using DOM

Change Image