Lesson 05 Demo 08

Working with Asynchronous JavaScript

Objective: To demonstrate the implementation of asynchronous JavaScript using Promises, async/await, for improving responsiveness and reliability in web applications

Tools required: Visual Studio Code and Node.js

Prerequisites: None

Steps to be followed:

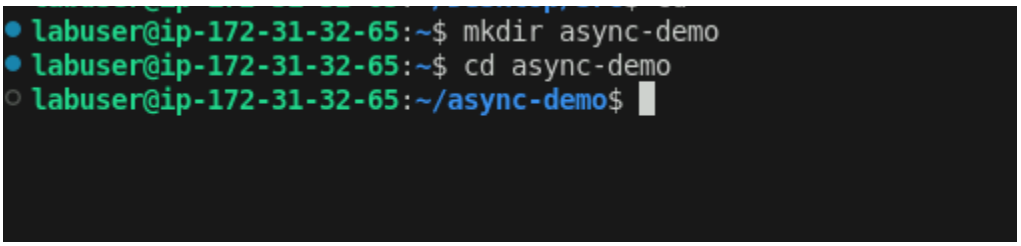
1. Create and set up the project
2. Develop the webpage structure
3. Implement JavaScript for asynchronous operations
4. Execute and verify the project

Step 1: Create and set up the project

1.1 Create a new project folder and navigate into it using the following command:

mkdir async-demo

cd async-demo



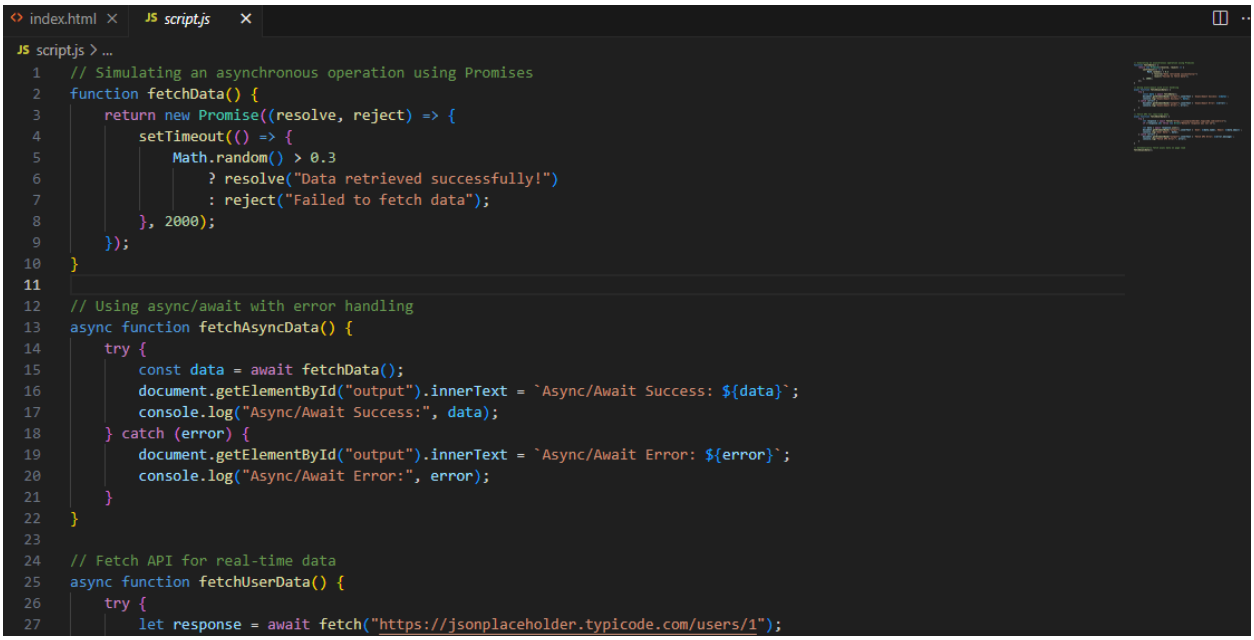
Step 2: Develop the webpage structure

2.1 Create an HTML file named **index.html** and add the following code:

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>Async JavaScript Demo</title>
</head>
<body>
   <h1>Working with Asynchronous JavaScript</h1>
   <button onclick="fetchUserData()">Fetch User Data</button>
   <p id="output"></p>

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

Step 3: Implement JavaScript for asynchronous operations

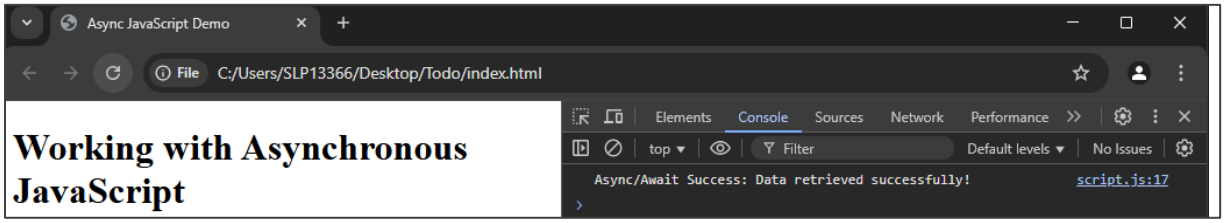
3.1 Create a JavaScript file named **script.js** and add the following code:



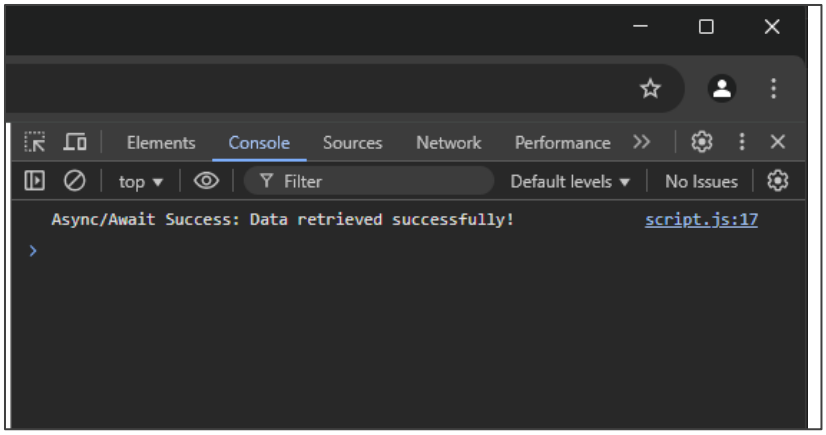
```
// Simulating an asynchronous operation using Promises
function fetchData() {
    return new Promise((resolve, reject) => {
        setTimeout(() => {
            Math.random() > 0.3
                ? resolve("Data retrieved successfully!")
                : reject("Failed to fetch data");
        }, 2000);
    });
}

// Using async/await with error handling
async function fetchAsyncData() {
    try {
        const data = await fetchData();
        document.getElementById("output").innerText = `Async/Await Success: ${data}`;
        console.log("Async/Await Success:", data);
    } catch (error) {
        document.getElementById("output").innerText = `Async/Await Error: ${error}`;
        console.log("Async/Await Error:", error);
    }
}

// Fetch API for real-time data
async function fetchUserData() {
    try {
        let response = await fetch("https://jsonplaceholder.typicode.com/users/1");
```

Step 4: Execute and verify the project

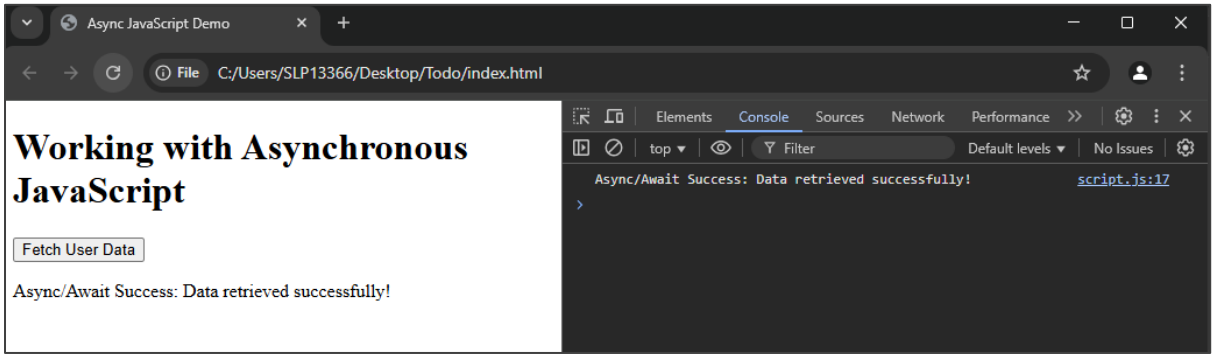
4.1 Open **index.html** in a web browser



4.2 Check the console (F12 > Console tab) for logs



4.3 Click on the **Fetch User Data** button to fetch data from the API



By following the above steps, you have successfully implemented asynchronous JavaScript using Promises, async/await, and the Fetch API in separate HTML and JavaScript files, ensuring better code organization and real-time data retrieval with error handling.