**Writing asynchronous JavaScript**

**Build a Web Page with Asynchronous JavaScript**

**Scenario:** You are building a small demo web page to understand asynchronous programming principles. Instead of fetching real data from an API, you'll simulate an asynchronous task like fetching data locally after a delay. This activity helps you learn callbacks, Promises, and async/await using concepts you already know.

**Step 1: create a new html file**

1. Select File > New File.
2. Press Enter and select OK.
3. Name it index.html.

**Step 2: build the html structure**

1. In index.html, set up a basic HTML5 document.
2. Add the following elements inside the <body>:

* A <h1> heading with the text "Async JavaScript Lab".
* A <button> with the ID fetch-data and the text "Fetch Data".
* A <div> with the ID data-container to display results.

**Step 3: create a new JavaScript file**

1. Select File > New File.
2. Name it script.js.
3. Press Enter and select OK.

**Step 4: write asynchronous JavaScript Code**

You will simulate data fetching and use callbacks, Promises, and async/await with local data and a delay using **setTimeout**.

**1. Callback Function:**

* Create a function that takes a callback.
* Use **setTimeout** to simulate a delay.
* Once the delay is complete, execute the callback with some mock data (for example, an array of names).

**2. Promises:**

* Convert the callback function into a function that returns a Promise.
* Resolve the Promise with the same mock data.

**3. Async/Await:**

* Use **async** and **await** to handle the Promise from the previous step.
* Update the DOM with the mock data using string concatenation and **.innerHTML**.

**4. Error Handling:**

* Wrap your **async/await** logic in a **try/catch** block.
* Display a console message if an error occurs.

**Step 5: Run your code**

* Click Go Live in the lower-right corner of Visual Studio Code.
* A new browser tab should open, displaying your page.
* Click the Fetch Data button to test your asynchronous logic.

**HTML:**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8" />

<meta name="viewport" content="width=device-width, initial-scale=1.0" />

<title>Async JavaScript Lab</title>

</head>

<body>

<h1>Async JavaScript Lab</h1>

<button id="fetch-data">Fetch Data</button>

<div id="data-container"></div>

</body>

<script src="script.js"></script>

</html>

**JavaScript:**

// --- Mock Data ---

**const** mockData = ['Alice', 'Bob', 'Charlie'];

// --- 1. Callback-based Fetch ---

**function** fetchDataWithCallback(callback) {

setTimeout(() => {

callback(mockData);

}, **1000**);

}

// --- 2. Promise-based Fetch ---

**function** fetchDataWithPromise() {

**return** **new** Promise((resolve, reject) => {

setTimeout(() => {

// Simulate success

resolve(mockData);

// To simulate error, uncomment the line below:

// reject('Failed to fetch data');

}, **1000**);

});

}

// --- 3 & 4. Async/Await + Error Handling ---

**async** **function** fetchAndDisplayData() {

**try** {

**const** data = **await** fetchDataWithPromise();

**const** container = document.getElementById('data-container');

container.innerHTML = '';

**let** html = '<ul>';

**for** (**let** name **of** data) {

html += '<li>' + name + '</li>';

}

html += '</ul>';

container.innerHTML = html;

} **catch** (error) {

console.error('Error fetching data:', error);

}

}

// --- Button Click Handler ---

document.getElementById('fetch-data').addEventListener('click', () => {

fetchAndDisplayData();

});

// Uncomment to test callback version

/\*

fetchDataWithCallback((data) => {

console.log('Callback Data:', data);

});

\*/