Web Development

Class No: 08

linkedin.com/in/m-munim1

github.com/M-Munim

+92 330 2021926

JavaScript Async/Await & API Handling

Understanding Asynchronous JavaScript

- JavaScript is Single-Threaded
- Imagine JavaScript as a chef in a small kitchen who can cook only one dish at a time (single-threaded).
- If the chef waits for water to boil before cutting vegetables, cooking will take forever!
- Instead, the chef multitasks—they put water to boil, then start chopping, and later check the water.

This is how asynchronous JavaScript works!

0330 2021926

Why Do We Need Asynchronous JavaScript?

JavaScript executes one operation at a time, but some tasks take time, like:

- Fetching data from a server
- Reading a file
- Waiting for a timer

If JavaScript waited for each task to finish, the whole program would freeze! Instead, it delegates long tasks and continues running other code.

03302021926

What is Async/Await?

Async/Await is a way to handle **asynchronous** code in JavaScript, making it easier to read and write.

Why do we need it?

JavaScript runs code line by line, but sometimes we need to wait for tasks like:

- Fetching data from a server (API calls)
- Reading a file
- Waiting for a timer

Earlier, we used callbacks (which led to "callback hell") or promises (.then()), but async/await makes it much cleaner!

0330 2021926

Breaking it down:

- async keyword

  When you add async before a function, it automatically returns a promise.

```
async function getData() {
    return "Hello, World!";
}
getData().then(console.log);                    // Output: Hello, World!
```

0330 2021926

await keyword

- await pauses the function until the promise is resolved.
- It must be used inside an async function.

```
async function fetchData() {
    let response = await fetch("https://api.example.com/data");
    let data = await response.json();
    console.log(data);              // This runs **only after** the data is fetched
}
fetchData();
```

03302021926

Example: Without and With Async/Await

- ❌ Without Async/Await (Using .then())

```
fetch("https://api.example.com/data")
.then(response => response.json())
.then(data => console.log(data))
.catch(error => console.log(error));
```

03302021926

With Async/Await (Cleaner & Easier)

```
async function getData() {
    try {
        let response = await fetch("https://api.example.com/data");
        let data = await response.json();
        console.log(data);
    } catch (error) {
        console.log(error);
    }
}
getData();
```

0330 2021926

Working with APIs using Fetch & Async/Await

What is an API?

An API (Application Programming Interface) allows different applications to communicate with each other.

For example:

- A weather app gets live weather updates from an API.
- A shopping app fetches product details from an API.

0330 2021926

What is fetch?

JavaScript provides the fetch() function to get data from APIs.

- It sends a request to the API.
- The API sends back data (usually in JSON format).
- We then process and use the data in our program.

0330 2021926

Fetching Data from an API

Example: Fetching Random Users from an API

```
async function fetchUser() {
        try {
            let response = await fetch("https://randomuser.me/api/"); // Fetch data from a public API

            let data = await response.json();              // Convert response to JSON format

            let user = data.results[0];                    // Extract user details
            console.log("User Name:", user.name.first, user.name.last);
            console.log("Email:", user.email);
            console.log("Country:", user.location.country);
        } catch (error) {
            console.log("Error fetching data:", error);
        }
    }

    fetchUser();
```

CSD 2021926

Breaking Down the Code:

- fetch("https://randomuser.me/api/") → Sends a request to the API.
- await response.json() → Converts the response into JSON format.
- Extracts user data (e.g., name, email, country).
- Handles errors using try...catch.

0330 2021926

How the API Response Looks (JSON Format)

When we call the API, it returns data like this:

JSON:

```
{
    "results": [
      {
        "name": { "first": "John", "last": "Doe" },
        "email": "johndoe@example.com",
        "location": { "country": "USA" }
      }
    ]
}
```

We extract name.first, name.last, email, and location.country from this JSON.

0330 2021926

Example: Fetching API Data and Displaying in HTML

```
<body>
  <button id="fetchBtn">Fetch User</button>
  <p id="userData"></p>

  <script>
    document.getElementById("fetchBtn").addEventListener("click", async function() {
      try {
        let response = await fetch("https://jsonplaceholder.typicode.com/users/1");
        let data = await response.json();
        document.getElementById("userData").textContent = `Name: ${data.name}, Email: ${data.email}`;
      } catch (error) {
        document.getElementById("userData").textContent = "Failed to fetch data";
      }
    });
  </script>
</body>
```

Submission Notes

Capture Evidence:

- Take screenshots or record a short video for each task.
- Ensure your evidence clearly shows the completed task.

Upload Tasks:

- Upload each task as a separate folder or file in your GitHub repository.

LinkedIn Post:

- Create a short post summarizing what you learned and how you completed the tasks.
- Highlight your key takeaways.
- Mention the following account in your post:
- **Muhammad Munim**

0330 2021926

Thank You

linkedin.com/in/m-munim1

github.com/M-Munim

+92 330 2021926