TECH Carrel
A Cubicle of Freelancers

Async and Await

In JavaScript, **async** is a keyword that is used to define asynchronous functions. An asynchronous function is a special type of function that allows you to work with asynchronous operations, such as making network requests, reading files, or performing other tasks that might take time to complete.

The async keyword is used before the function keyword when defining a function. It marks the function as asynchronous and **enables the use of the await keyword inside that function**.

The **await** keyword is used to pause the execution of an asynchronous function until a Promise is resolved, allowing you to write code that appears more synchronous and readable, even when dealing with asynchronous operations.

Handling promises using async and await

```
<script>
 async function fetchData() {
 try {
   const response = await
fetch('https://jsonplaceholder.typicode.com/users');
   const data = await response.json();
   console.log(data);
 } catch (error) {
   console.error('An error occurred:', error);
 }
}
fetchData();
</script>
```

Handling promises using .then & .catch

```
<script>
function fetchUserDataFromServer() {
        fetch(`https://jsonplaceholder.typicode.com/users`)
        .then(response => {
          if (!response.ok) {
            throw new Error(`HTTP error! Status: ${response.status}`);
          }
          return response.json();
        })
        .then(data => {
          console.log(data)
        })
        .catch(error => {
          console.error("Fetch error:", error);
        });
    }
    fetchUserDataFromServer()
```

In this example, the fetchData function is marked as async, allowing the use of await inside it. The await keyword is used when fetching data from an API. It waits for the promise returned by fetch to resolve before continuing with the execution of the function. This makes the code read as if it's executing synchronously, even though it's dealing with asynchronous operations.

Tips:

1. **async Functions:** Functions marked as async always return a Promise. If you return a non-Promise value from an async function, it will be automatically wrapped in a resolved Promise.

2. **await Keyword:** The await keyword can only be used inside an async function. It pauses the execution of the function until the awaited Promise is resolved or rejected.

3. **Error Handling:** You can use try and catch blocks to handle errors that might occur during asynchronous operations.

4. **Sequential Execution:** With await, you can write code that appears to execute sequentially, even when dealing with asynchronous tasks. This can lead to cleaner and more maintainable code.

async and await are powerful tools for managing asynchronous operations in a more readable and organized manner, allowing you to write asynchronous JavaScript code that is easier to understand and maintain.

Akhilesh Kumar Gupta
(Technical Training specialist)
TechCarrel LLP