# Async Await in Js

Async and await are part of the asynchronous programming paradigm in JavaScript, introduced in ECMAScript 2017 (ES8). They provide a way to handle asynchronous operations more comfortably, making the code look and behave more like synchronous code.

## Asynchronous Programming in JavaScript

JavaScript is single-threaded, meaning it can only do one thing at a time. However, it can perform asynchronous operations, like fetching data from a server, reading files, or waiting for a timer, without blocking the main thread. This is crucial for maintaining a responsive user interface.

## Promises

Before async/await, asynchronous operations were often handled with promises. A promise represents a value that might be available now, in the future, or never. It allows chaining of asynchronous operations using .then() and .catch().

Example

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

## Async/Await

Async and await provide a more readable and concise way to work with promises.

- `async`: The async keyword is used to declare an asynchronous function. It implicitly returns a promise, and any value returned from the function is wrapped in a resolved promise.
- `await`: The await keyword is used to pause the execution of the async function until the promise is resolved. It can only be used inside an async function.

Example with Async/Await:

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

fetchData();
```

## Why Use Async/Await?

- *Readability*: Async/await makes the code easier to read and understand. It resembles synchronous code, making it easier to follow the flow of execution.
- *Error Handling*: With async/await, error handling can be done using try/catch blocks, which are more familiar and easier to manage compared to handling errors with promises.
- *Debugging*: Debugging async/await code is often easier because the stack traces are more straightforward, and you can use standard debugging techniques like breakpoints and step-through debugging.

---