Async/Await in JavaScript

@_codevalley

Zuhaib Asif

What is Async/Await?

Async/Await is a way to write asynchronous code that looks and behaves more like synchronous code. It simplifies working with promises, which are objects representing the eventual completion (or failure) of an asynchronous operation.

Real-Life Example

To understand async/await, let's consider a real-life example of making a sandwich. Imagine you're in a sandwich shop and you want to place an order. You give your order to the sandwich artist, and they start making it. However, making the sandwich takes some time, so you don't want to wait there doing nothing until it's ready. Instead, you'd like to do something else, like reading a book or checking your emails, and come back when the sandwich is done.

SWIPE

Code of Previous Example:

```
// The sandwich-making function
function makeSandwich() {
  return new Promise((resolve, reject) => {
   setTimeout(() => {
    resolve('Your sandwich is ready!');
   }, 2000); // Simulating 2 seconds of sandwich-
making time
});
}
// The ordering function using async/await
async function placeOrder() {
  console.log('Placing your sandwich order...');
  const result = await makeSandwich();
  console.log(result);
  console.log('Enjoy your sandwich!');
}
// Calling the ordering function
  placeOrder();
  console.log('Doing something else...');
```

Explanation of Previous Code

In the previous code, the makeSandwich() function returns a promise that resolves with the message 'Your sandwich is ready!' after a delay of 2 seconds (simulating the time it takes to make the sandwich).

The placeOrder( ) function is marked with the async keyword, indicating that it contains asynchronous operations. Inside this function, we can use the await keyword to pause the execution until the promise returned by makeSandwich() is resolved. During this time, the JavaScript engine is free to do other tasks, like executing the code after the await statement.
When the promise is resolved, the result is assigned to the result variable, and the execution continues. In this example, it logs the message 'Your sandwich is ready!' and then 'Enjoy your sandwich!'. Finally, outside the placeOrder() function, it logs 'Doing something else...'.

@_codevalley Zuhaib Asif

Output:

The output would be:

```
Placing your sandwich order...
Doing something else...
Your sandwich is ready!
Enjoy your sandwich!
```

This way, the program doesn't get blocked while waiting for the sandwich to be made. Instead, it moves on to perform other tasks until the awaited promise is resolved, making the code more readable and easier to reason about.

Conclusion:

In summary, async/await allows you to write asynchronous JavaScript code that looks and behaves more synchronously, making it easier to work with promises and manage asynchronous operations.