JavaScript Async and Await

Introduction

JavaScript's async and await keywords provide a more readable and concise syntax for writing asynchronous code, making it easier to work with promises. Async functions allow you to write promise-based code as if it were synchronous, without blocking the main thread.

Async Function

An async function is a function declared with the **async** keyword. It always returns a promise, which resolves with the value returned by the async function, or rejects with an error thrown from within the function.

```
async function getAllData(){
    await getData(1); //apke liye wait kiya jayga
    await getData(2);
    console.log('I am third statement');
}
```

Await Keyword

The **await** keyword can only be used within an async function. It pauses the execution of the async function until the promise is settled (either fulfilled or rejected), and then resumes the async function's execution.

```
async function getAllData(){
    await getData(1); //apke liye wait kiya jayga
    await getData(2);
    console.log('I am third statement');
}
```

IIFE (Immediately Invoked Function Expression)

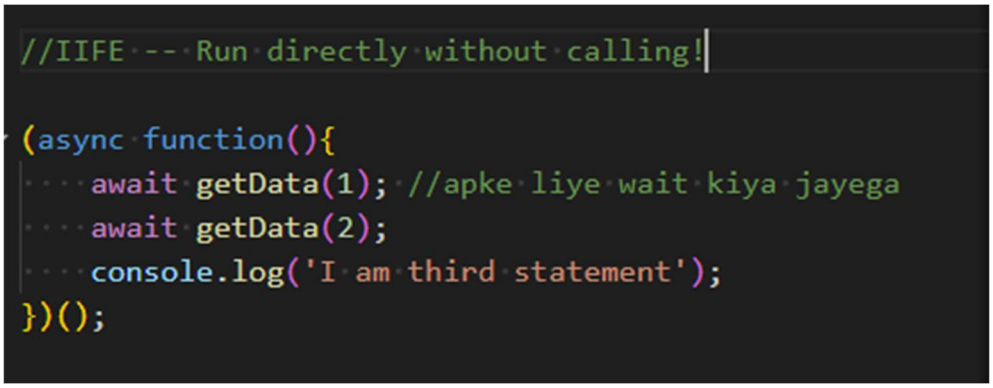
An Immediately Invoked Function Expression is a JavaScript design pattern which produces a lexical scope using a function expression that's immediately invoked. It's often used to encapsulate variables and functions to avoid polluting the global namespace.

```
(function() {
  // code here
})();
```

Benefits of IIFE

1. **Encapsulation**: Variables and functions declared within the IIFE are scoped to the function and do not pollute the global scope.
2. **Initialization**: IIFEs can be used to initialize variables or set up configurations before the rest of the code runs.
3. **Module Pattern**: IIFEs are often used as the building blocks of the module pattern in JavaScript, providing encapsulation and preventing variable leakage.

Example of IIFE



Conclusion

Async and await are powerful tools for handling asynchronous operations in JavaScript, making code easier to read and write. Meanwhile, IIFEs provide a way to encapsulate code and prevent global namespace pollution, contributing to better code organization and maintainability.