Closure in Javascript

@_codevalley

Zuhaib Asif

What is Closure?

A Closure is a combination of a function and the lexical environment within which that function was declared. This lexical environment consists of the variables that were in scope at the time the closure was created. Closures allow functions to retain access to variables from their outer (enclosing) scopes even after those outer functions have finished executing.

What is Closure?

```javascript
function outerFunction() {
  // Variable declared in the outer function
  let outerVariable = 'I am from the outer function';

  // Inner function (closure) declared within the outer function
  function innerFunction() {
    console.log(outerVariable); // Accessing outerVariable from the outer scope
  }

  // Returning the inner function (closure)
  return innerFunction;
}

// Creating a closure by calling outerFunction and assigning the result to a variable
let closureExample = outerFunction();

// Invoking the closure, which still has access to the outerVariable
closureExample(); // Output: "I am from the outer function"
```

Explanation Of Previous Code:

1. innerFunction is a closure because it is declared inside outerFunction and has access to outerVariable, even though outerFunction has finished executing. When outerFunction is called and assigned to closureExample, it essentially "captures" the environment in which it was created, including outerVariable. Later, when closureExample is invoked as a function, it can still access and use outerVariable.

Conclusion:

Closures are powerful in JavaScript and are commonly used to create private variables, implement data hiding, and achieve various other programming patterns. They play a crucial role in creating modular and maintainable code.