**Closures** and **callbacks** are both important concepts in JavaScript, but they serve different purposes. Let's look at what each of these is, how they differ, and where they are used.

**Closure**

A **closure** is a function that "remembers" the environment or scope in which it was created, even after that scope is gone. Closures allow you to access variables from an outer function after the outer function has finished executing. Essentially, closures provide a way to create private data or to keep references to variables alive.

(**Closure** ek function hota hai jo apne create hone wale environment ya scope ko "yaad" rakhta hai, chahe woh scope khatam ho chuka ho. Closure aapko outer function ke variables ko access karne ki suvidha deta hai, even jab outer function execute ho chuka ho. Closure ka use private data banane ya kisi variable ko zinda rakhne ke liye kiya jata hai.)

function outerFunction() {

let count = 0; // Outer function variable

return function() {

count++;

console.log(count);

};

}

const counter = outerFunction(); // `counter` is now a closure

counter(); // Output: 1

counter(); // Output: 2

Is example mein, return kiya gaya function count variable ko access kar pa raha hai, chahe outerFunction ka execution khatam ho chuka ho. Yehi **closure** hai.

#### Key Points:

* **Keeps State**: Closure outer scope ke variables ko yaad rakh sakta hai.
* **Encapsulation**: Private variables aur functions create karne mein useful