**Scope**

**Defines the accessibility of variables.**

Global scope\*\*: variables accessible anywhere.

Local scope\*\*: variables accessible only inside functions or blocks.

Example:

```js

function example() {

let localVar = 10; // local scope

console.log(localVar);

}

**Closures**

A closure is a function that retains access to variables from its outer (enclosing) scope even after that outer function has returned.

Example:

```js

function counter() {

let count = 0;

return function() {

count++;

return count;

};

}

const increment = counter();

console.log(increment()); // 1

console.log(increment()); // 2

```

**Closures enable powerful patterns like data encapsulation and function factories.**

**🔹 A. Global Scope**

Variables declared outside any function or block.

let globalVar = "I am global";

function test() {

console.log(globalVar); // ✅ Accessible

}

**🔹 B. Function Scope**

Variables declared inside a function are **not accessible outside**.

function myFunc() {

let localVar = "Inside Function";

console.log(localVar); // ✅

}

console.log(localVar); // ❌ ReferenceError

🔹 C. **Block Scope (let and const)**

if (true) {

let blockVar = "Inside block";

console.log(blockVar); // ✅

}

console.log(blockVar); // ❌

**🔸 Example: Closure in Action**

function outer() {

let counter = 0;

function inner() {

counter++;

console.log(counter);

}

return inner;

}

const count = outer(); // outer() runs once, returns inner()

count(); // 1

count(); // 2

count(); // 3