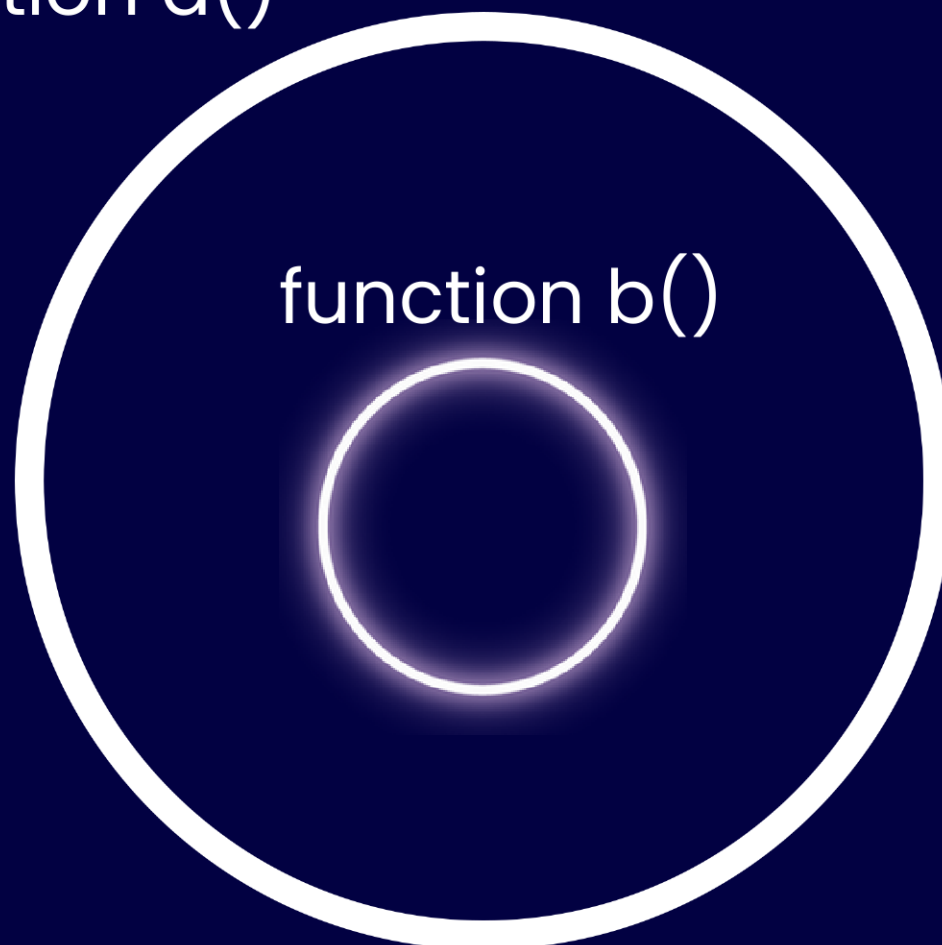


# Closures

javascript

A closure is a powerful feature in JavaScript where an inner function has access to the outer (enclosing) function's variables even after the outer function has executed.

function a()



function b()

## How Do Closures Work?

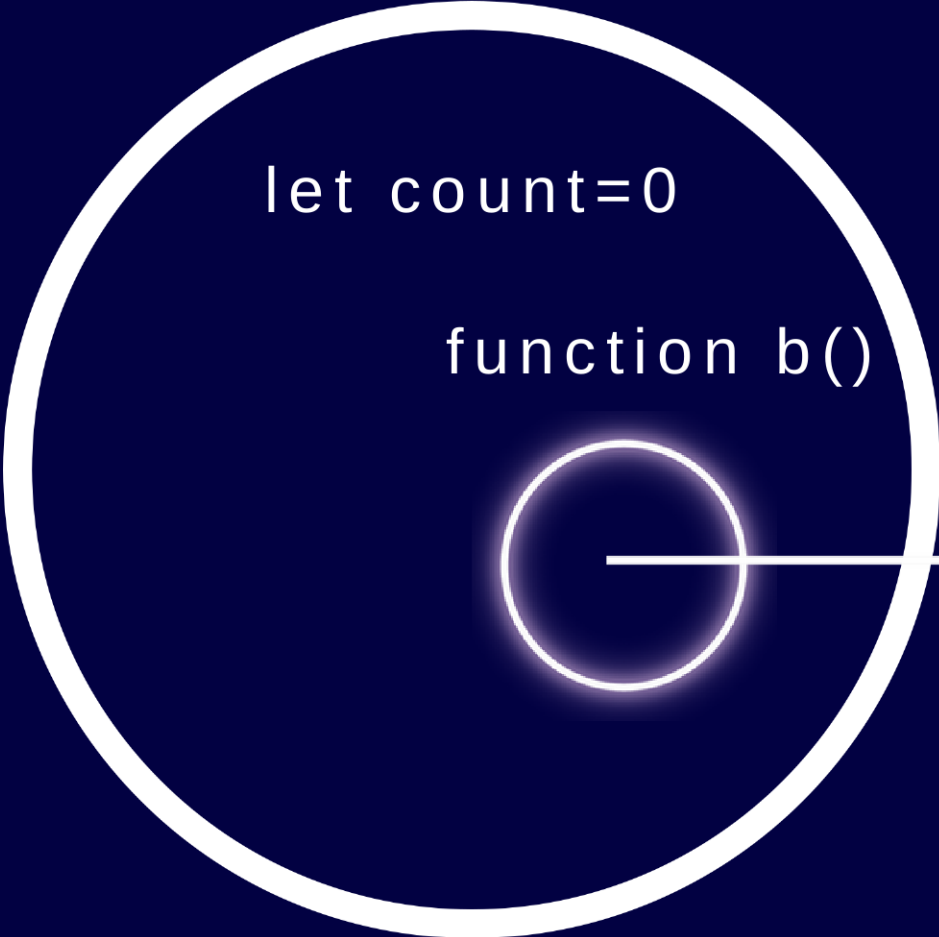
```
function a() {  
    let count = 0; // This variable is enclosed  
  
    function b() {  
        count += 1;  
        return count;  
    }  
  
    return b;  
}  
  
const increment = a();  
console.log(increment()); // Output: 1
```

function a()

let count=0

function b()

**count -(0+1) initially**



The diagram consists of two concentric circles. The outer circle is defined by a thick black border and contains the text 'function a()' above it and 'let count=0' inside it. The inner circle is defined by a thin black border and contains the text 'function b()' above it. A horizontal arrow points from the center of the inner circle to the right, passing through the boundary of the outer circle and pointing towards the text 'count -(0+1) initially'.

**Explanation:**

**Outer Function (function a()):** This function defines a variable count and returns an inner function.

**Inner Function:** This function increments the count variable and returns it.

**Closure:** The inner function forms a closure, retaining access to count even after createCounter has finished executing.

## **Key Point:**

**The inner function has access to variables in its own scope, the scope of the outer function, and the global scope.**