Gmail Bikash Adhikari <virtual.bikash@gmail.com>

Task 2.1: Creating a Counter Using Closures Create...

1 message

Bikash <bikash.caim@gmail.com> Sat, Mar 8, 2025 at 6:52 PM
To: virtual.bikash@gmail.com

JavaScript

Task 1: Creating a Counter Using Closures

Create a function `createCounter()` that returns a function which increments and returns a counter value each time it is called.

```
function createCounter() {
  let count = 0;

  return function() {
    count++;
    return count;
  };
}

// Example usage:
const counter1 = createCounter();
console.log(counter1()); // Output: 1
console.log(counter1()); // Output: 2
console.log(counter1()); // Output: 3

const counter2 = createCounter(); //another counter
console.log(counter2()); //output: 1
console.log(counter1()); //output: 4, counter1 is not impacted by counter2
```

Explanation:

1. `createCounter()` **function:**
   - This function defines a local variable `count` and initializes it to 0.
   - It then returns an inner function. This inner function is the actual counter.
2. **Inner function (the counter):**
   - This function has access to the `count` variable from its outer scope (the `createCounter()` function) due to closure.
   - Each time it's called, it increments `count` and returns the updated value.
3. **Closure:**
   - The key concept here is closure. The inner function "closes over" the `count` variable, meaning it retains access to it even after the `createCounter()` function has finished executing.
   - Therefore, each time the inner function is called, it remembers the previous value of `count`.

- When you create a new counter by calling createCounter() again, a new count variable is created, and a new closure is created, so the counters don't interfere with each other.