Make Counter II

Premium Completed

</> Medium 10mins 2.19k done

Implement a function `makeCounter` that accepts an optional integer value (defaults to 0) and returns an object that contains the following methods:

- `get()`: returns the current value.
- `increment()`: increments the current value and returns it.
- `decrement()`: decrements the current value and returns it.
- `reset()`: resets the current value to the initial value.

Examples

```
const counter = makeCounter();
counter.get(); // 0
counter.increment(); // 1
counter.increment(); // 2
counter.get(); // 2
counter.reset(); // 0
counter.decrement(); // -1
```

With a custom initial value:

```
const counter = makeCounter(5);
counter.get(); // 5
counter.decrement(); // 4
counter.decrement(); // 3
counter.get(); // 3
counter.reset(); // 5
counter.increment(); // 6
```

Solution

Approach 1: Directly return an object

1. The `makeCounter` function accepts an optional parameter `initialValue`, which is set to 0 by default.
2. Inside the `makeCounter` function, we declare a variable `count` and initialize it with the provided `initialValue`.
3. Return an object with four methods: `get`, `increment`, `decrement`, and `reset`:
    - The `get` method simply returns the current value of count.
    - The `increment` method increments the count by 1 using the prefix increment operator (`++x`) and returns the updated value.
    - The `decrement` method decrements the count by 1 using the prefix decrement operator (`--x`) and returns the updated value.
    - The `reset` method resets the `count` to the initial value provided in the `makeCounter` function and returns it.

JavaScript TypeScript

```javascript
/**
 * @param {number} initialValue
 * @return {{get: Function, increment: Function, decrement: Function, reset: Function }}
 */
export default function makeCounter(initialValue = 0) {
  let count = initialValue;

  return {
    get: () => count,
    increment: () => ++count,
    decrement: () => --count,
    reset: () => (count = initialValue),
  };
}
```

Approach 2: Return a `Counter` class

We can also create a separate `Counter` class that contains the 4 required methods

We can also create a separate `Counter` class that contains the 4 required methods.

The `makeCounter` function will simply be a wrapper than returns an instance of `Counter` initialized with the `initialValue`. The main difference here is that we need to use instance variables to keep track of the current value and the `initialValue` since we cannot rely on closures.

JavaScript TypeScript

```
class Counter {
  constructor(initialValue = 0) {
    this.initialValue = initialValue;
    this.value = initialValue;
  }
  get() {
    return this.value;
  }
  increment() {
    return ++this.value;
  }
  decrement() {
    return --this.value;
  }
  reset() {
    this.value = this.initialValue;
    return this.value;
  }
}

/**
 * @param {number} initialValue
 * @return {{get: Function, increment: Function, decrement: Function, reset: Function }}
 */
export default function makeCounter(initialValue = 0) {
  return new Counter(initialValue);
}
```