

1. Create Hello World Function

Write a function `createHelloWorld`. It should return a new function that always returns "Hello World".

Example 1:

Input: args = []

Output: "Hello World"

Explanation:

```
const f = createHelloWorld();
f(); // "Hello World"
```

The function returned by `createHelloWorld` should always return "Hello World".

Example 2:

Input: args = [{},null,42]

Output: "Hello World"

Explanation:

```
const f = createHelloWorld();
f({}, null, 42); // "Hello World"
```

Any arguments could be passed to the function but it should still always return "Hello World".

```
/**  
 * @return {Function}  
 */
```

```
var createHelloWorld = function() {  
  
    return function(...args) {  
        }  
    };  
  
/**  
 * const f = createHelloWorld();  
 * f(); // "Hello World"  
 */
```

2. Counter

Hint

Given an integer n, return a counter function. This counter function initially returns n and then returns 1 more than the previous value every subsequent time it is called (n, n + 1, n + 2, etc).

Example 1:

Input:

n = 10

["call","call","call"]

Output: [10,11,12]

Explanation:

counter() = 10 // The first time counter() is called, it returns n.

```
counter() = 11 // Returns 1 more than the previous time.
```

```
counter() = 12 // Returns 1 more than the previous time.
```

Example 2:

Input:

```
n = -2
```

```
["call","call","call","call","call"]
```

Output: [-2,-1,0,1,2]

Explanation: counter() initially returns -2. Then increases after each subsequent call.

```
/**  
 * @param {number} n  
 * @return {Function} counter  
 */  
  
var createCounter = function(n) {  
  
    return function() {  
  
    };  
};  
  
/**  
 * const counter = createCounter(10)  
 * counter() // 10  
 * counter() // 11  
 * counter() // 12  
 */
```

3. To Be Or Not To Be

Easy

Companies

Write a function expect that helps developers test their code. It should take in any value val and return an object with the following two functions.

- `toBe(val)` accepts another value and returns true if the two values === each other. If they are not equal, it should throw an error "Not Equal".
- `notToBe(val)` accepts another value and returns true if the two values !== each other. If they are equal, it should throw an error "Equal".

Example 1:

Input: `func = () => expect(5).toBe(5)`

Output: `{"value": true}`

Explanation: 5 === 5 so this expression returns true.

Example 2:

Input: `func = () => expect(5).toBe(null)`

Output: `{"error": "Not Equal"}`

Explanation: 5 !== null so this expression throw the error "Not Equal".

Example 3:

Input: `func = () => expect(5).notToBe(null)`

Output: `{"value": true}`

Explanation: 5 !== null so this expression returns true.

/**

```

* @param {string} val
* @return {Object}
*/
var expect = function(val) {

};

/** 
* expect(5).toBe(5); // true
* expect(5).notToBe(5); // throws "Equal"
*/

```

4. Counter II

Hint

Write a function `createCounter`. It should accept an initial integer `init`. It should return an object with three functions.

The three functions are:

- `increment()` increases the current value by 1 and then returns it.
- `decrement()` reduces the current value by 1 and then returns it.
- `reset()` sets the current value to `init` and then returns it.

Example 1:

Input: `init = 5, calls = ["increment","reset","decrement"]`

Output: `[6,5,4]`

Explanation:

```
const counter = createCounter(5);
```

```
counter.increment(); // 6  
counter.reset(); // 5  
counter.decrement(); // 4
```

Example 2:

Input: init = 0, calls = ["increment","increment","decrement","reset","reset"]

Output: [1,2,1,0,0]

Explanation:

```
const counter = createCounter(0);  
counter.increment(); // 1  
counter.increment(); // 2  
counter.decrement(); // 1  
counter.reset(); // 0  
counter.reset(); // 0
```

```
/**  
 * @param {integer} init  
 * @return { increment: Function, decrement: Function, reset: Function }  
 */
```

```
var createCounter = function(init) {
```

```
};
```

```
/**  
 * const counter = createCounter(5)
```

```
* counter.increment(); // 6  
* counter.reset(); // 5  
* counter.decrement(); // 4  
*/
```