

Problem 2704: To Be Or Not To Be

Problem Information

Difficulty: Easy

Acceptance Rate: 0.00%

Paid Only: No

Problem Description

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.

Code Snippets

JavaScript:

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

};

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

TypeScript:

```
type ToBeOrNotToBe = {
  toBe: (val: any) => boolean;
```

```

notToBe: (val: any) => boolean;
};

function expect(val: any): ToBeOrNotToBe {

};

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

```

Solutions

JavaScript Solution:

```

/**
 * Problem: To Be Or Not To Be
 * Difficulty: Easy
 * Tags: general
 *
 * Approach: Optimized algorithm based on problem constraints
 * Time Complexity: O(n) to O(n^2) depending on approach
 * Space Complexity: O(1) to O(n) depending on approach
 */

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

};

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

```

TypeScript Solution:

```
/**
 * Problem: To Be Or Not To Be
 * Difficulty: Easy
 * Tags: general
 *
 * Approach: Optimized algorithm based on problem constraints
 * Time Complexity:  $O(n)$  to  $O(n^2)$  depending on approach
 * Space Complexity:  $O(1)$  to  $O(n)$  depending on approach
 */

type ToBeOrNotToBe = {
  toBe: (val: any) => boolean;
  notToBe: (val: any) => boolean;
};

function expect(val: any): ToBeOrNotToBe {

};

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