Problem 2692: Make Object Immutable

Problem Information

Difficulty: Medium
Acceptance Rate: 0.00%
Paid Only: No

Problem Description

Write a function that takes an object

obj

and returns a new

immutable

version of this object.

An

immutable

object is an object that can't be altered and will throw an error if any attempt is made to alter it.

There are three types of error messages that can be produced from this new object.

Attempting to modify a key on the object will result in this error message:

`Error Modifying: ${key}`

.

Attempting to modify an index on an array will result in this error message:

`Error Modifying Index: ${index}`

.

Attempting to call a method that mutates an array will result in this error message:

`Error Calling Method: ${methodName}`

. You may assume the only methods that can mutate an array are

['pop', 'push', 'shift', 'unshift', 'splice', 'sort', 'reverse']

.

obj

is a valid JSON object or array, meaning it is the output of

JSON.parse()

.

Note that a string literal should be thrown, not an

Error

.

Example 1:

Input:

obj = {   "x": 5 } fn = (obj) => {   obj.x = 5;   return obj.x; }

Output:

{"value": null, "error": "Error Modifying: x"}

Explanation:

Attempting to modify a key on an object resuts in a thrown error. Note that it doesn't matter that the value was set to the same value as it was before.

Example 2:

Input:

```
obj = [1, 2, 3] fn = (arr) => {   arr[1] = {};   return arr[2]; }
```

Output:

```
{"value": null, "error": "Error Modifying Index: 1"}
```

Explanation:

Attempting to modify an array results in a thrown error.

Example 3:

Input:

```
obj = {   "arr": [1, 2, 3] } fn = (obj) => {   obj.arr.push(4);   return 42; }
```

Output:

```
{ "value": null, "error": "Error Calling Method: push"}
```

Explanation:

Calling a method that can result in a mutation results in a thrown error.

Example 4:

Input:

```
obj = {   "x": 2,   "y": 2 } fn = (obj) => {   return Object.keys(obj); }
```

Output:

{"value": ["x", "y"], "error": null}

Explanation:

No mutations were attempted so the function returns as normal.

Constraints:

obj

is a valid JSON object or array

2 <= JSON.stringify(obj).length <= 10

5

Code Snippets

JavaScript:

```
/**
* @param {Object|Array} obj
* @return {Object|Array} immutable obj
*/
var makeImmutable = function(obj) {

};

/**
* const obj = makeImmutable({x: 5});
* obj.x = 6; // throws "Error Modifying x"
*/
```

TypeScript:

```
type JSONValue = null | boolean | number | string | JSONValue[] | { [key:
string]: JSONValue };
type Obj = Array<JSONValue> | Record<string, JSONValue>;
```

```
function makeImmutable(obj: Obj): Obj {

};

/**
* const obj = makeImmutable({x: 5});
* obj.x = 6; // throws "Error Modifying x"
*/
```

Solutions

JavaScript Solution:

```
/**
* Problem: Make Object Immutable
* Difficulty: Medium
* Tags: array, string, sort
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

/**
* @param {Object|Array} obj
* @return {Object|Array} immutable obj
*/
var makeImmutable = function(obj) {

};

/**
* const obj = makeImmutable({x: 5});
* obj.x = 6; // throws "Error Modifying x"
*/
```

TypeScript Solution:

```
/**
* Problem: Make Object Immutable
* Difficulty: Medium
* Tags: array, string, sort
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

type JSONValue = null | boolean | number | string | JSONValue[] | { [key:
string]: JSONValue };
type Obj = Array<JSONValue> | Record<string, JSONValue>;

function makeImmutable(obj: Obj): Obj {

};

/**
* const obj = makeImmutable({x: 5});
* obj.x = 6; // throws "Error Modifying x"
*/
```