

Task 6

To be or Not to be:

The screenshot shows the '30 Days of JavaScript' submission interface. On the left, a table lists submissions with columns for Status, Language, Runtime, and Memory. The right panel shows a code editor with a JavaScript function and a test case.

Status	Language	Runtime	Memory
Accepted	JavaScript	29 ms	53.3 MB
Wrong Answer	JavaScript	N/A	N/A
Wrong Answer	JavaScript	N/A	N/A
Wrong Answer	JavaScript	N/A	N/A
Runtime Error	JavaScript	N/A	N/A
Runtime Error	JavaScript	N/A	N/A
Runtime Error	JavaScript	N/A	N/A
Runtime Error	JavaScript	N/A	N/A
Wrong Answer	JavaScript	N/A	N/A

```
function notToBe(val) {
  return true;
} else {
  throw new Error("Not Equal");
}

notToBe : function(notToBeVal) {
  if(notToBeVal !== val) {
    return true;
  } else {
    throw new Error("Equal");
  }
}
```

Testcase: Case 1 Case 2 Case 3 +

func =

() => expect(5).toBe(5)

Counter:

The screenshot shows the '30 Days of JavaScript' submission interface. On the left, a table lists submissions. The right panel shows a code editor with a JavaScript function and a test case.

Status	Language	Runtime	Memory
Accepted	JavaScript	53 ms	58.6 MB
Wrong Answer	JavaScript	N/A	N/A
Runtime Error	JavaScript	N/A	N/A
Runtime Error	JavaScript	N/A	N/A

```
function Counter() {
  return {
    increment: function() {
      return num++;
    },
    decrement: function() {
      return num--;
    },
    reset: function() {
      return num = init;
    }
  };
}
```

Testcase: Case 1 Case 2 +

init =

5

calls =

["increment", "reset", "decrement"]

Remove duplicates from sorted array:

The screenshot shows a coding editor interface with a 'Submissions' table on the left and a 'Code' editor on the right. The 'Submissions' table lists several attempts, with the most recent one being 'Accepted'. The 'Code' editor shows a JavaScript solution that uses a two-pointer technique to remove duplicates from a sorted array.

Status	Language	Runtime	Memory
Accepted	JavaScript	1 ms	58.2 MB
Runtime Error	JavaScript	N/A	N/A
Runtime Error	JavaScript	N/A	N/A
Runtime Error	JavaScript	N/A	N/A
Compile Error	C++	N/A	N/A
Compile Error	C++	N/A	N/A
Compile Error	C++	N/A	N/A

```
JavaScript
17
18
19
20
21
22
23
24
25
26
27
for(let m = 0; m < k; m++){
  nums[m] = expectedNums[m];
}
for(let n = k+1; n < expectedNums.length; n++){
  nums[n] = "-";
}
return k;
```

Testcase: Case 1, Case 2 +

nums = [1,1,2]

Best Time to buy and sell stock:

The screenshot shows a coding editor interface with a 'Submissions' table on the left and a 'Code' editor on the right. The 'Submissions' table lists several attempts, with the most recent one being 'Accepted'. The 'Code' editor shows a JavaScript solution that uses a single pass to find the maximum profit by tracking the minimum price seen so far.

Status	Language	Runtime	Memory
Accepted	JavaScript	1 ms	65.3 MB
Wrong Answer	JavaScript	N/A	N/A
Time Limit Exceeded	JavaScript	N/A	N/A
Time Limit Exceeded	JavaScript	N/A	N/A
Runtime Error	JavaScript	N/A	N/A
Time Limit Exceeded	JavaScript	N/A	N/A
Time Limit Exceeded	JavaScript	N/A	N/A
Runtime Error	JavaScript	N/A	N/A
Runtime Error	JavaScript	N/A	N/A
Wrong Answer	JavaScript	N/A	N/A
Wrong Answer	JavaScript	N/A	N/A
Wrong Answer	JavaScript	N/A	N/A

```
JavaScript
1 /**
2  * @param {number[]} prices
3  * @return {number}
4  */
5 var maxProfit = function(prices) {
6   let min = prices[0];
7   let max = 0;
8   for(let i = 1; i < prices.length; i++){
9     if(prices[i] < min){
10       min = prices[i];
11     }
12     let profit = prices[i] - min;
```

Testcase: Case 1, Case 2 +

prices = [7,1,5,3,6,4]

Majority Element:

The screenshot shows a coding editor interface with a 'Submissions' table on the left and a 'Code' editor on the right. The 'Submissions' table lists several attempts, with the most recent one being 'Accepted'. The 'Code' editor shows a JavaScript solution that uses a hash map to count the frequency of each element and returns the element with the highest frequency.

Status	Language	Runtime	Memory
Accepted	JavaScript	6 ms	56.7 MB
Wrong Answer	JavaScript	N/A	N/A

```
JavaScript
1 /**
2  * @param {number[]} nums
3  * @return {number}
4  */
5 var majorityElement = function(nums) {
6   let k = 1;
7   let uniqueNums = [nums[0]];
8   let comparison = nums[0];
9   for(let i = 1; i < nums.length; i++){
10    if(comparison !== nums[i] && !uniqueNums.includes(nums[i])){
11      comparison = nums[i];
12      uniqueNums[k] = nums[i];
```

Testcase: Case 1, Case 2 +

nums = [3,2,3]