

Ace your JavaScript Interview. [Get my ebook](#). 100 solved Javascript, 20 solved React, & 2 frontend system design questions (**1160+ copies sold**). Get a [Free preview](#).



Find the correct position to insert an element in the array

Posted on [January 26, 2019](#) | by [Prashant Yadav](#)

Posted in [Algorithms](#), [Arrays](#) | Tagged [Easy](#)

An algorithm to find the correct position to insert a given element in the [array](#).

We will implement a simple algorithm in javascript to find the correct position to insert an element in the given array.

Assume the given array will be sorted and it will not contain any duplicates.

Example

Input:

[1, 3, 5, 6]

5

[1, 3, 5, 6]

2

[1, 3, 5, 6]

7

[1, 3, 5, 6]

0

Output:

2 //5 is present at index 2 in [1, 3, 5, 6]

1 //2 can be inserted after 1 and before 3 at position 1 in [1, 3, 5, 6]

4 //7 can be inserted after 6 at position 4

0 //0 can be inserted before 1 at position 0

[Copy](#)

Practically
prepare for
your
JavaScript
interview

[JavaScript
Revision](#)[JavaScript-
Concept Based
Problems](#)[Data Structures](#)[Algorithms](#)[Machine
Coding](#)[Web
Fundamentals](#)

Implementation

- We will check if element is already present in the array or not and if it is present then return its position.
- If it is not present then we will find the position of the element greater than itself and return its position.



- If there is no element less than the input element in the array then we will return the position after the last element.
- Everything will be written in [ES6](#).

```
let searchInsert = (nums, target) => {  
  
    //keep track of the element  
    let found = 0;  
    let isFound = false;  
  
    //check if element is already present  
    for(let i = 0; i < nums.length; i++){  
  
        //if element is found then return its position  
        if(target === nums[i]){  
            found = i;  
            isFound = true;  
            break;  
        }  
    }  
  
    //if an element is not found then find the position where it can be inserted  
    if(!isFound){  
        for(let i = 0; i < nums.length; i++){  
  
            //if the target element is less than the element in the array then add it  
            //before  
            if(target < nums[i]){  
                found = i;  
                isFound = true;  
                break;  
            }  
        }  
    }  
  
    //if position is found then return it else return the position after the last  
    //element in the array  
    return isFound ? found : nums.length;  
};
```

```
Input:  
console.log(searchInsert([1,3,5,6], 5));  
console.log(searchInsert([1,3,5,6], 2));  
console.log(searchInsert([1,3,5,6], 7));  
console.log(searchInsert([1,3,5,6], 0));
```

```
Output:  
2  
1  
4  
0
```

Time complexity: $O(n)$.

Space complexity: $O(1)$.

Time and Space complexity

- We are iterating through the given array twice one after another, so Time complexity is $O(n + n) = O(n)$.
- We are using constant space, so Space complexity is $O(1)$.

Prepare for your **JavaScript Interview** [practically on each Interview rounds and grab that job.](#)

BEGIN LEARNING

Recommended Posts:

- [Program to check the prime number](#)
- [Algorithm to merge two sorted array](#)
- [Find longest palindrome in a string](#)
- [Longest Consecutive Sequence](#)
- [Selection sort in javascript](#)
- [Program to print the diamond pattern](#)
- [Minimum characters to delete to make string anagram](#)
- [Iterative heap sort in Javascript](#)
- [Palindrome string](#)
- [Find the maximum sum of products of two arrays.](#)

[Prev](#)

[Next](#)

Leave a Reply

Your email address will not be published. Required fields are marked *

Comment

Start typing...

Name*

Name

Email*

Email

POST COMMENT

[About Us](#) [Contact Us](#) [Privacy Policy](#) [Advertise](#)



Handcrafted with ♥ somewhere in **Mumbai**

© 2023 [LearnersBucket](#) | [Prashant Yadav](#)

