Check if an array is palindrome in javascript

Posted on January 14, 2019 | by Prashant Yadav

Posted in Algorithms, Arrays | Tagged Easy

An algorithm to check if a given array is palindrome or not.

We will use two different methods to check if a given array is a palindrome or not in javascript. Everything will be written in ES6.

> **Palindrome**: A word, sequence or number that reads same when reversed.

Example

```
Input:
[1, 2, 3, 2, 1]
[1, 2, 3, 3, 1]

Output:
true
false
```

Brute force method

Implementation

- We are going to loop through the half of the array.
- And we will check if first half of the array is equal to the other half or not.
- If it is equal then return `true` else return `false`.

Practically prepare for your JavaScript interview

JavaScript Revision
JavaScript-Concept Based Problems
Data Structures
Algorithms
Machine Coding
Web Fundamentals

```
let palindromeArray = (arr) => {
     //initialize to true
     let isPalindrome = true;

     //loop through half length of the array
     for(let i = 0; i < arr.length / 2; i++) {

         //check if first half is equal to the second half
         if(arr[i] !== arr[arr.length - i - 1]){
           isPalindrome = false;
           break;
         }
     }

     return isPalindrome;
}
```

```
Input:
console.log(palindromeArray([1,2,2,1]));
console.log(palindromeArray([1,2,2,2]));

Output:
true
false
```

Time complexity: O(n).

Space complexity: O(1).

Time and Space complexity

- We are looping through half of the array O(n/2), so Time complexity is O(n).
- We are using constant space, so Space complexity is O(1).

Using recursion to check the palindrome array.

We can use recursive functions as well to check the palindrome array in javascript.

Implementation

- We will create a function and check if first element of the given array is equal to the last element then call the function again with second element and second last element. Repeat this till there is only 1 element.
- If array contains only one element then return `true` else return `false`.

```
let palindromeArray = (arr, start = 0, end = arr.length - 1) => {
    //if array has only element
    if(start >= end){
      return true;
    }

    //if start is equal to end
    if(arr[start] === arr[end]){
      //call the same function
      return palindromeArray(arr, start + 1, end - 1);
    }else{
      //else not palindrome
      return false;
    }
}
```

```
Input:
console.log(palindromeArray([1,2,2,1]));
console.log(palindromeArray([1,2,2,2]));

Output:
true
false
```

We are using higher-order functions of ES6 in which we can assign the left parameter value to the right parameters `(arr, start = 0, end = arr.length - 1)`. Learn more about it here.

Time complexity: O(n).

Space complexity: O(n).

Time and Space complexity

- We are calling the same function for half of the array O(n/2). so Time complexity is O(n).
- We are storing each function in call stack, so Space complexity is O(n).

Recommended Posts:

- Convert string to array in javascript
- Program to print the floyd triangle
- Learn how to reverse words in a string
- Reverse a stack using recursion.
- Find the maximum depth of nested parentheses in a string
- Convert decimal to binary, octal or hexa in javascript
- Program to find the nth Fibonacci in javascript
- Find the most frequent element in an array
- Find missing alphabets to make a string panagram
- Reverse a sublist of linked list

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

Handcrafted with ♥somewhere in **Mumbai**