

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](#).



Advertisements

Check if two string are anagram of each other

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

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

An algorithm to check if two [string](#) are anagram of each other.

We will implement a simple algorithm in javascript to check if the given two string are anagram of each other or not. Everything will be written in [ES6](#).

Anagram: An anagram of a string is another string that contains same characters, only the order of characters can be different.

Example

Input:

```
'prashant'
'tnahsarp'

'learnersbucket'
'tekcubsrenraes'
```

Output:

```
true
false
```

Copy

We will use different methods to solve this anagram strings problem.

- Using sorting $O(n \log n)$.
- By counting the letters of strings $O(n)$.

Using sorting.

Implementation

- We will sort both strings in ascending order and check if they are equal.
- If they are 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 anagramStrings = (str1, str2) => {

    //split the string to character array
    //sort the character array
    //then join the sorted array to form the string
    let sortedStr1 = str1.split('').sort().join('');
    let sortedStr2 = str2.split('').sort().join('');

    //return true if equal else return false
    return sortedStr1 === sortedStr2;
}
```

Copy

```
Input:
console.log(anagramStrings('prashant', 'tnahsarp'));
console.log(anagramStrings('learnersbucket', 'tekcubsrenraes'));

Output:
true
false
```

Copy

We have used string `split()` method to create the characters `array`.
Then using `sort()` method we have sorted the character array.
After that with `join()` we have joined the characters array to create the string.

Time complexity: $O(n \log n)$.

Space complexity: $O(n)$.

Time and Space complexity

- We are using `split()` method to create the characters array which will take $O(n)$. Then sorting the array will take $O(n \log n)$ and to form the string again with `join()` will take $O(n)$, so Time complexity is $O(n + n \log n + n) = O(n \log n)$.
- We are creating the character array from the given string which will take $O(n + n)$, so Space complexity is $O(n)$.

By counting the letters of the string.

Implementation

- If both the strings are not equal then return `false`.
- We will keep track of the characters in both the string and count their occurrences.
- If all the characters count is equal then return `true` else return `false`.

[Copy](#)

```
let anagramStrings = (str1, str2) => {  
  //if both the strings are not equal then return false  
  if(str1.length !== str2.length){  
    return false;  
  }  
  
  //create two objects to keep track  
  let track = {};  
  let track2 = {};  
  
  //count the character occurrences of first string  
  for(let i = 0; i < str1.length; i++){  
    if(!track[str1[i]]){  
      track[str1[i]] = 1;  
    }else{  
      track[str1[i]]++;  
    }  
  }  
  
  //count the character occurrences of second string  
  for(let i = 0; i < str2.length; i++){  
    if(!track2[str2[i]]){  
      track2[str2[i]] = 1;  
    }else{  
      track2[str2[i]]++;  
    }  
  }  
  
  //check if the character occurrences in both the string are not equal then  
  return false;  
  for(let i = 0; i < str1.length; i++){  
    if(track1[str1[i]] !== track2[str2[i]]){  
      return false;  
    }  
  }  
  
  return true;  
}
```

[Copy](#)

Input:
console.log(anagramStrings('prashant', 'tnahsarp'));
console.log(anagramStrings('learnersbucket', 'tekubsrenraes'));

Output:
true
false

Time complexity: $O(n)$.

Space complexity: $O(n)$.

Advertisements



Time and Space complexity

- We are counting the character occurrences of both the strings which will take $O(n + n)$, Then check if count is equal or not in $O(n)$, so Time complexity is $O(n + n + n) = O(n)$.
- We are keeping track of characters count, so Space complexity is $O(n)$.

Prepare for your **JavaScript Interview** practically on each Interview rounds and grab that job.

BEGIN LEARNING

Recommended Posts:

[Find kth smallest and largest element in BST.](#)

[Longest Common Subsequence](#)

[Print all subarrays with a given sum k in an array](#)

[Convert string to array in javascript](#)

[Number of subarrays with given sum k](#)

[Recursively reverse a doubly linked list](#)

[Find all the armstrong number between two numbers](#)

[Program to print the chess board pattern in javascript](#)

[Next permutation problem](#)

[6 ways to convert string to a number in javascript](#)

[Prev](#)

[Next](#)

Comments

Mix-Movie.com says:

[October 21, 2019 At 4:14 Pm](#)

Method 4 (Bit Manipulation) The above implementation can be further optimized by using bit manipulation. If we start at a value of 0 and XOR all the characters of both strings, we should return an end value of 0 if they are anagrams because there would be an even occurrence of all characters in the anagram. Done forget to defend the code by validating inputs.

[Reply](#)

Leave a Reply

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

Comment

Start typing...



Name*

Name

Email*

Email

POST COMMENT

Advertisements



[About Us](#)

[Contact Us](#)

[Privacy Policy](#)

[Advertise](#)



Handcrafted with ♥ somewhere in **Mumbai**

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

