Learn how to reverse words in a string

Posted on September 21, 2020 | by Prashant Yadav

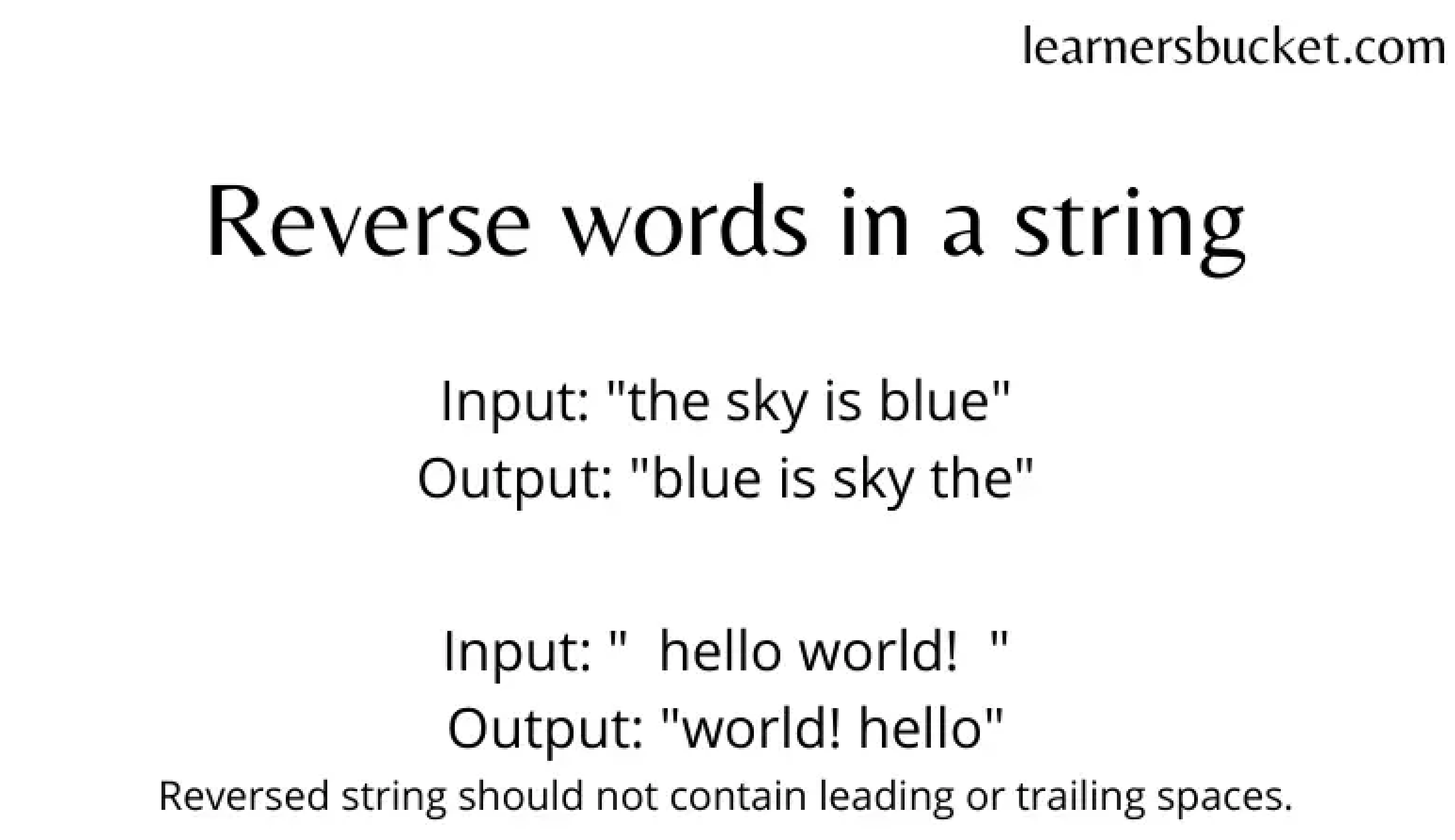
Posted in Algorithms, String | Tagged Easy

Given a string reverse each words in it.

Example

```
Input: "the sky is blue"
Output: "blue is sky the"

Input: "  hello world!  "
Output: "world! hello"
Explanation: Reversed string should not contain leading or trailing spaces.
```



learnersbucket.com

Reverse words in a string

Input: "the sky is blue"
Output: "blue is sky the"

Input: " hello world! "
Output: "world! hello"
Reversed string should not contain leading or trailing spaces.

There are multiple ways through which this problem can be solved, but we will categorize it in two parts.
1. Using extra space.
2. Using constant space.

Practically prepare for your JavaScript interview

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

Reverse words in a string using constant space.

Conceptually this is how it works

- Use two variables to track the white spaces on both the ends.
- Then iterate the string without the trailing spaces and keep adding each character to form a word. If we encounter a empty space then add the formed word to the final result and reset the word.
- Keep doing this for each character in the string.
- In the end after the iteration if there is character in the word then add that to the result.

Copy

```
const reverseWords = (s) => {
    let left = 0;
    let right = s.length-1;

    //remove the trailing spaces from both end
    while (s[left] === " ") left++;
    while (s[right] === " ") right--;

    let word = "";
    let result = "";

    //Iterate the string and keep on adding to form a word
    //If empty space is encountered then add the formed word to the result
    while (left <= right) {
        const char = s[left];
        if (char !== " ") {
            word += char;
        } else if (word && char === " ") {
            if (result) result = word + " " + result;
            else result = word;
            word = "";
        }
        left++;
    }

    //If not empty string then add to the result
    if (word) {
        if (result) result = word + " " + result;
        else result = word;
    }

    return result;
}
```

Copy

```
Input:
console.log(reverseWords("learnersbucket practice code"));

Output:
"code practice learnersbucket"
```

Time complexity: O(n).

Space complexity: O(4).

Advertisements

Reverse words in a string by utilizing extra space.

Basically in this approach we break the string in the array of words and then reverse the array and join it again to form the string.

There are multiple ways to do that.

Advertisements

Method 1: Using stack.

Copy

```
const reverseWords = (s) => {
    //Split into array of words
    const words = s.split(' ');

    //Use stack to reverse the words
    const stack = [];

    //Add each from array to the stack
    for(let i of words) {
        stack.push(i);
    }

    let fS = "";

    //Get each word from stack and form the reversed string
    while(stack.length) {
        const cS = stack.pop();
        if(cS) {
          fS += " " + cS;
        }
    }

    //Return the string
    return fS.trim();
};
```

Method 2: Simply iterating the words array in reverse direction

```
const reverseWords = (str) => {
  const words = str.split(" ");
  let ans = "";
  for(let i = words.length - 1; i >= 0; i--){
    ans = `${ans} ${words[i]}`;
  }

  return ans.trim();
}
```

Method 3: Filtering the empty space from the array of words so that we don't have to trim it at the end.

```
const reverseWords = (s) => {
    return s.split(" ").filter(Boolean).reverse().join(" ");
};
```

Method 4: Properly splitting the string on each word.

```
const reverseWords = s =>  s.trim().split(/\s+/).reverse().join(' ');
```

For each of these methods

Time complexity: O(n).

Space complexity: O(n).

Recommended Posts:

Find distinct ways to climb the stairs in javascript.

Longest Consecutive Sequence

Alternatively merge two different arrays

Reverse a sublist of linked list

Implement queue using two stack

Find all the armstrong number between two numbers

Count number of sub string recursively

Convert a string to uppercase in javascript

Iterative heap sort in Javascript

Fractional knapsack problem

Prev Next

Advertisements