Converting string to jadencase

Posted on December 30, 2018 | by Prashant Yadav

Posted in Algorithms, String | Tagged medium

An algorithm to convert the given string to jadenCase or capitalize the string.

JadenCase: Every first letter of each words of string should be uppercase.

Example

Copy

```
Input:
"How can mirrors be real if our eyes aren't real"

Output:
"How Can Mirrors Be Real If Our Eyes Aren't Real"
```

We will see three different methods to convert the string to jadencase.

- We are going to convert the string to an array of words and then convert the first letter of each word to uppercase which will work in O(n*m), Where n is the length of the string and m is the length of the longest word in the string.
- We will use the regular expression and `string.replace()` to achieve the same which will take O(n).
- Everything will be return in ES6.

Practically prepare for your JavaScript interview

JavaScript Revision

JavaScript-Concept Based Problems

Data Structures

Algorithms

Machine Coding

Web Fundamentals

Copy

```
let toJadenCase = (str) => {
  //create an array of words
  let x = str.split(" ");

  //create an empty array to store the converted words
  let y = [];

  //loop through each words
  for(let i = 0; i < x.length; i++){

    //splite each words to an array of characters
    let s = x[i].split("");

    //convert the first letter to uppercase
    let temp = s[0];
    s[0] = temp.toUpperCase();

    //push the converted word
    y.push(s.join(''));

 }
 //Join the converted words and return the string
 return y.join(" ");

};
```

Copy

```
Input:
console.log(toJadenCase("How can mirrors be real if our eyes aren't real"));

Output:
"How Can Mirrors Be Real If Our Eyes Aren't Real"
```

Time complexity: O(n * m);

space complexity: O(n);

Time and Space complexity

- We are splitting the string on whitespace and creating an array of words which will take O(n). After that, we will be converting an array of characters for each word that will take O(m) time. We will be doing it for each word so Time complexity is O(n * m).
- We will be converting an array of words and then the array of characters for that words, so Space complexity is O(n * m).

Above solution with inbuilt array methods

Copy

```
let toJadenCase = (str) => {
  return str.split(" ").map(function(word){
    return word.charAt(0).toUpperCase() + word.slice(1);
  }).join(" ");
}
```

```
Input:
console.log(toJadenCase("How can mirrors be real if our eyes aren't real"));

Output:
"How Can Mirrors Be Real If Our Eyes Aren't Real"
```

Using regular expression

Implementation

- We are going to use regular expressions to replace the first letter of each word and with its uppercase.
- We will use `string.replace()` method.

```
let toJadenCase = (str) => {
  return str.replace(/(^|\s)[a-z]/g, (x) => { return x.toUpperCase(); });
}
```

```
Input:
console.log(toJadenCase("How can mirrors be real if our eyes aren't real"));

Output:
"How Can Mirrors Be Real If Our Eyes Aren't Real"
```

Time complexity: O(n).
Space complexity: O(1).

Time and Space complexity

- We are using `string.replace()` to replace the word and regular expression to find the first letter of each word. As regular expression time depends upon it implementation, still we will be checking the first character of each word. so Time complexity O(n).
- We are using constant space, so Space complexity is O(1).

Recommended Posts:

- Longest repeated subsequence
- Top view of a binary tree
- Find square of a number without using *, / and pow()
- Best way to compare strings in javascript
- Print the last k nodes of the linked list in reverse.
- Invert a binary tree | Recursive and Iterative solutions
- Implement stack with max and min function

Absolute difference between diagonals of matrix

Reverse a string using stack

Sort a stack using another stack

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**