String Replacement Documentation

**JavaScript Solution**

Since I am more familiar with JavaScript, I thought I would incorporate the function for replacing one portion of a string with another into both a front-end form and a back-end Node command.

To run with Node:

1. Navigate to GitHub repo (<https://github.com/sgliput/string-replacement>)

2. Clone locally in the command line using `git clone [https://github.com/sgliput/string-replacement.git`](https://github.com/sgliput/string-replacement.git%60)

3. Navigate into created folder with `cd string-replacement`

4. Run stringReplacement.js with `node stringReplacement.js` followed by the original string, the string to remove, and the string with which to replace it, all separated by spaces. (For example, `node stringReplacement.js “The bluebird is blue.” blue red` will return “The redbird is red.”)

The main JavaScript function (ReplaceString())used for this file is the same used for the front-end form that runs in the browser. Opening codingAssessment.html in a browser will yield a form where the user can type an original string, the string to remove from it, and the string with which to replace it, all in separate form fields. The old and new strings are then displayed with a button click.

Function Explanation:

The ReplaceString() function accepts three arguments: strOrig for the original string, strFind for the string to remove/replace, and strReplace for the string to insert in place of strFind. In lieu of JavaScript native replace function, I used the split() method to separate strOrig into an array separated on instances of strFind, thus replacing strFind with commas in the array. Using the join() method and joining the array with strReplace then changes the array back into a string with strReplace everywhere that strFind was before.

For the front-end form, I also capture the values of the input fields and tie the function call to a button click. The new and original strings are then inserted into two HTML <p> tags below the form.

While I don’t have much experience with testing harnesses, I did include a Karma test, which when configured right (start by typing `npm i` in the command line) returns four successful tests (type `npm test` in the command line). These tests can be found in the lib/codingAssessment.test.js file. These tests can be added to by changing the string values, and the test will determine if the resulting string matches the expected result.

**Java Solution**