Lab 2

Due Oct 14, 2016 by 5pm **Points** 100 **Submitting** a file upload

CS-546 Lab 2

Asynchronous Code, Files, and Promises

This week will be interesting -- we're going to be working with files; particularly, reading them, creating metrics on them, and storing them using promises.

You'll need to write two modules this week.

Your file module, fileData.js

This module will export four methods:

getFileAsString(path)

This method will, when given a path, return a promise that resolves to a string with the contents of the files.

If no path is provided, it will return a rejected promise.

If there are any errors reading the file, the returned promise will reject rather than resolve, passing the error to the rejection callback.

getFileAsJSON(path)

This method will, when given a path, return a promise that resolves to a JavaScript object. You can use the `JSON.parse` function to convert a string to a JavaScript object (if it's valid!).

If no path is provided, it will return a rejected promise.

If there are any errors reading the file or parsing the file, the returned promise will reject rather than resolve, passing the error to the rejection callback.

Hint: this function can be accomplished in approximately 3-4 lines. Don't overcomplicate it!

saveStringToFile(path, text)

This method will take the `text` supplied, and store it in the file specified by `path`. The function should return a promise that will resolve to `true` when saving is completed.

If no path or text is provided, it will return a rejected promise.

If there are any errors writing the file, the returned promise will reject rather than resolve, passing the error to the rejection callback.

saveJSONToFile(path, obj)

This method will take the `obj` supplied and convert it into a string so that it may stored as in a file. The function should return a promise that will resolve to `true` when saving is completed.

If no path or obj is provided, it will return a rejected promise.

If there are any errors writing the file, the returned promise will reject rather than resolve, passing the error to the rejection callback.

Your metric module, textMetrics.js

This module will export one method, `createMetrics(text)` which will scan through the text **ignoring case** and return an object with the following information:

```
{
    totalLetters: total number of letters in the text,
    totalWords: total number of words in the text,
    uniqueWords: total number of unique words that appear in the text,
    longWords: number of words in the text that are 6 or more letters long,
    averageWordLength: the average number of letters in a word in the text,
    numberOfSentences: total number of sentences in the text,
    textComplexity: totalWords/numberOfSentences + (longWords x 100)/totalWords
    wordOccurrences: {
        word1: number of times that word appears in the text,
        word2: number of times that word appears in the text,
        etc...
    }
}
```

So running:

`createMetrics("Hello, my friends! This is a great day to say hello.")`

Will return:

```
    totalLetters: 39,
    totalWords: 11,
    uniqueWords: 10,
    longWords: 1,
    averageWordLength: 3.55,
    textComplexity: 14.59
    wordOccurrences: {
        hello: 2,
        my: 1,
        friends: 1,
        this: 1,
        is: 1,
        a: 1,
        great: 1,
        day: 1,
        to: 1,
```

```
        say: 1
    }
}
```

Important note:

You will ignore any character that is punctuation. Ignore all spaces. Ignore all commas. Ignore all question marks, exclamation marks, periods, single quotes, double quotes, etc. You will only care about letters and spaces.

app.js

Write a file, app.js, which will print out the results from the following test files:

- chapter1.txt (https://sit.instructure.com/courses/14567/files/1799204/download?wrap=1) (https://sit.instructure.com/courses/14567/files/1799204/download?wrap=1) (https://sit.instructure.com/courses/14567/files/1799204/download?wrap=1)
- chapter2.txt (https://sit.instructure.com/courses/14567/files/1799203/download?wrap=1) (https://sit.instructure.com/courses/14567/files/1799203/download?wrap=1) (https://sit.instructure.com/courses/14567/files/1799203/download?wrap=1)
- chapter3.txt (https://sit.instructure.com/courses/14567/files/1799202/download?wrap=1) (https://sit.instructure.com/courses/14567/files/1799202/download?wrap=1) (https://sit.instructure.com/courses/14567/files/1799202/download?wrap=1)

Requirements

1. You **must not submit** your node_modules folder
2. You **must remember** to save your dependencies to your package.json folder
3. You must do basic error checking in each function
    1. Check for arguments existing and of proper type.
    2. Throw if anything is out of bounds (ie, trying to perform an incalculable math operation or accessing data that does not exist)
    3. If a function should return a promise, instead of throwing you should return a rejected promise.