CSE186 Summer 2023

Assignment 2

In this assignment you will write a simple JavaScript class and tests to confirm it functions as expected. You will also gain some familiarity with well-known development tools for building, testing, documenting, and analyzing JavaScript programs outside the browser.

This assignment is worth 10% of your final grade.

Installation

Download and install node.js LTS for your operating system: https://nodejs.org/en/download/.

Do NOT install the "Current" version.

Note that node.js includes the Node Package Manager `npm`.

Once installed, open a terminal / console session and run the following command:

```
$ node --version
```

Which should return this version number:

```
v18.16.1
```

Now run the following command:

```
$ npm --version
```

Which should return this version number:

```
9.5.1
```

If you run into problems getting `node` and `npm` installed, come along to an discussion section or an office hours session as soon as possible so we can help you get up and running.

Setup

Download the starter code archive from Canvas and expand into an empty folder. I recommend, if you have not already done so, creating a folder for the class and individual folders beneath that for each assignment.

The starter code archive contains the following files:

```
.eslintrc.js
README
package.json
templater.js
templater.test.js
```

You will modify `templater.js` and `templater.test.js` but under no circumstances modify `.eslintrc.js` or `package.json`.

To setup the node environment, ***navigate to the folder where you extracted the starter code*** and run the following command:

```
$ npm install
```


This will take some time to execute as `npm` downloads and installs all the packages required to build and test the assignment.

To execute the tests, run the following command:

```
$ npm test
```

To check the code quality against Google standards by running `eslint`, run the following command:

```
$ npm run lint
```

Requirements

Basic:

Implement a JavaScript class that when supplied with a template including tags enclosed in double curly braces (e.g. `{{sometag}}`) produces a string with the tags replaced by values found in a name-value-pair map i.e. a simple JavaScript object.

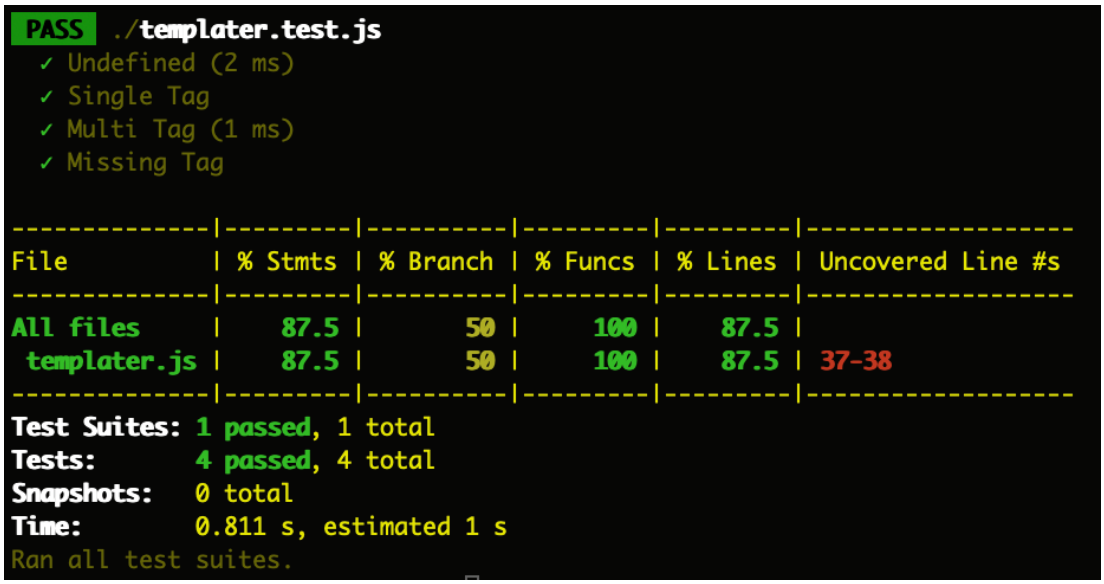
For example, after running the following code:

```
const t = new Templater('Hello {{tag}}');
let s = t.apply({tag: 'World'});
```

The variable `s` will have the value `'Hello World'`.

To get this working, complete the implementation of the `Templater` class in `templater.js` so it passes the tests provided in `templater.test.js`.

Once complete, running the tests (see “Setup” section) should produce output something like this:



Note that the special cases of an undefined template or missing tag values in the map supplied to `apply()` have to handled correctly.

Do not modify `template.test.js`.

For missing tag values, any extra white space created by *not* replacing a tag has to be removed.

For example, running the following code:

```
const t = new Templater('Mary {{had}} a {{little}} {{lamb}}');
let s = t.apply({had: 'had', lamb: 'lamb'});
```

Should give `s` the value `'Mary had a lamb'` with exactly one space between each word.

Advanced:

Modify `template.js` to ensure the `Templater` class has the following features:

When the `strict` parameter to `apply()` is `true` and one or more tags are missing from `map`, an `Error` should be thrown. For example, when the following code is executed an `Error` should be thrown by the second line:

```
let t = new Templater('Mary {{had}} a {{little}} {{lamb}}');
let s = t.apply({had: 'had', lamb: 'lamb'}, true));
```

In addition, if a tag in the template has whitespace in it, it should be ignored. For example, when the following code is executed (note the whitespace in the '`had`' tag):

```
let t = new Templater('Mary {{had }} a {{little}} {{lamb}}');
let s = t.apply({had: 'had', little: 'little', lamb: 'lamb'}));
```

The variable `s` will have the value `'Mary a little lamb'`.

The following additional scenarios should be tested:

- There are no spaces between tags in the template
  `'Mary {{had}}{{little}}' => 'Mary hadlittle'`
- Tags appear more than once in the template
  `'Mary {{had}} {{had}}' => 'Mary had had'`
- Tags are separated by characters other than spaces in the template
  `'Mary {{had}}-{{little}}' => 'Mary had-little'`

We strongly suggest you consult the existing test code and the documentation at https://jestjs.io then modify `template.test.js` to include this new functionality, but you do not have to do that, you can test manually if you like.

Your code will be graded against known good tests so take care to ensure your implementation does what is expected of it.

Stretch:

Your code and tests have no lint errors or warnings, and your implementation exhibits 100% statement, branch, function, and line coverage.

Note that the intent is for you to tackle this requirement once Advanced has been met but failing part or all of Advanced does not mean you automatically fail this requirement. You must, however, implement automated tests for the Advanced requirement to pass Stretch.

What steps should I take to tackle this?

Review the lecture handouts and consult any on-line resources you find useful. If you get stuck, come along to office hours to ask questions and get some help.

How much code will I need to write?

A model solution that satisfies all requirements adds approximately 20 lines of code to `templater.js` and around 60 lines to `templater.test.js`.

Grading scheme

The following aspects will be assessed:

1. (100%) **Does it work?**

    a. Basic Requirement (40%)
    b. Advanced Requirement (40%)
    c. Stretch Requirement (20%)

2. (-100%) **Did you give credit where credit is due?**

    a. Your submission is found to contain code segments copied from on-line resources and you failed to give clear and unambiguous credit to the original author(s) in your source code (-100%). You will also be subject to the university academic misconduct procedure as stated in the class academic integrity policy.

    b. Your submission is determined to be a copy of a past or present student's submission (-100%)

    c. Your submission is found to contain code segments copied from on-line resources that you did give a clear an unambiguous credit to in your source code, but the copied code constitutes too significant a percentage of your submission:

        - < 25% copied code No deduction
        - 25% to 50% copied code (-50%)
        - > 50% copied code (-100%)

What to submit

Run the following command to create the submission archive:

```
$ npm run zip
```

**** UPLOAD** `CSE186.Assignment2.Submission.zip` **TO THE CANVAS ASSIGNMENT AND SUBMIT ****

