DMIT2008 Assignment 3

Task: Stock Price Application Update

Due: October 31st 2020

Weight: 15%

Overview

You must add some functional updates to the existing stock price application from the previous assignment. Drawing on what you have learned over the past several weeks, you will make additions to the existing codebase that satisfy the new requirements of the application.

Expectations

Download and extract the included application zip starter package (assignment-03-starter.zip). You will find the interface has been updated slightly (you are not tasked with making any updates to the user interface code), please review the js/src directory to see the updated interface elements.

Your updated application must demonstrate the following:

- You must complete the implementation (to a suitable degree) of the test stubs that are present in the test/stock.test.js file
    - The required base packages for testing (e.g. jest, coverage etc.) have already been added to package.json; you may have to add additional packages to properly define additional tests/assertions
    - Please see the existing comments in the stock.test.js file for direction
    - Please NOTE if there is "BONUS" in the name (that isn't added by yourself) that this is only for bonus marks and isn't
- Stock constructor function (You can also create a stock Class with a stock constructor function):
    - You must separate the functionality of the stock API into a separate module with a constructor function. The spec for the Stock module is presented later in this document.
    - Use the stock.test.js file as a guide for what to do in the stock.js module
- You are solely responsible for all parts of this assignment (all JS code).
- As for bonus points:

    - Create a stock history chart that displays on the page. This should be triggered when a user clicks the "View History" button.
    - Include a script in your package.json that will show your test coverage.
    - Clearly indicate the percentage of test coverage of the stock.js in a comment within your "stock.test.js" file.
- **Any other requirements as laid out by your instructor in class**

Stock Module Definition

Your definition of the Stock module must meet the following requirements:

- The Stock module must export a named Stock constructor function (i.e. do not export using 'default').
- The Stock constructor function may be called in one of two ways (Note: Arrow function notations can't be used as constructor function you need to use the traditional "function something() {}" for this.):
    - With no parameter (i.e. the default call)
        - The constructor should define a default *symbol* property, which will be assigned an empty string as an initial value
        - The constructor should define a default *stockData* property, which will be assigned an empty object literal as an initial value
    - With an object of instance attributes
        - Each attribute of the passed object must be applied to the new Stock instance
- The Stock constructor should have all methods defined on its prototype
    - No functions should be defined inside of the constructor function itself.
- The functions "getStockPrice" and "getStockFiveDayHistory" need to be either be added to the prototype, or if you're using a class they need to be added as class methods. (Note: this is either or not both)
- Update the index.html and main.js to make proper use of this new module once it is completed and successfully tested.
- Please note, that you shouldn't need to change any functions with "display" in the function name in the main.js file.

Delivery

Zip your project folder and submit it to Moodle by the deadline.

- **Do not include the node_modules/ directory in your zip package**.

Seek help if you need it. **Late submissions will not be graded.**

Grading Key

Tasks	Grade	Marks	Total
Testing Jest used to build test suite • Assertions for Stock and constructor suites provided test stubs are complete and adequately test for the desired outcomes (see spec) • Assertions for method suites provided test stubs are complete and adequately test for the desired outcomes (see spec) • Final undefined suite is correctly implemented and assertion(s) adequately test for the desired outcome(s) (see spec)	3 5 5	3 5 5	13
Design Patterns Create a Stock constructor • Separate file, exported and loaded as a named module • Three required methods are correctly defined and function as expected (see spec) • Makes use of prototype for functions (see spec) • Constructor is properly defined and all required properties are present for both call types (see spec) • Stock module correctly implemented in the index.html, and main.js is updated to correctly make use of the new module • Application behaves just as it did prior to the implementation of the Stock module	3 5 3 3 5 1	3 5 3 3 5 1	17

Bonus Marks Create a displayPriceHistory chart method • Must display the price of a stock on each day (this can be a bar chart or a line chart) days need to be the x axis, and one of the prices be the y axis. Install and use test coverage on our file. • Make a script in you package.json to run test coverage. • At the top of your stock.test.js include the "test coverage percentage", needs to be exact.	***[1]*** ***[1]*** ***[1]***	***[1]*** ***[1]*** ***[1]***	***[3]***
Code Readability and Efficiency • Demonstrates best practice as demonstrated in class lectures and examples • Efficient and maintainable techniques • Code format • Well documented • Utilizes your own written code (i.e. not gathered from online sources such as StackOverflow)	-	-5	-

TOTAL MARKS		30 ***[33]***

Marking Rubric

Marks	5 Marks Criteria [minus]
5 [0]	Task was completed with the highest of proficiency, adhering to best practices, and followed subject matter guidelines. The task was completed to a professional standard.
4 [-1]	Task was completed well, with some minor mistakes. Well above average work, shows good understanding of the task, and a high degree of competence.
3 [-2]	Task was completed satisfactorily. Some features are missing or incorrectly implemented. Shows a moderate level of understanding in the task with room for improvement.
2 [-3]	Task completion is below average, the task was poorly completed. Shows understanding of the task and the requirements to implement, but implementation was poorly executed.
1 [-4]	Some of the task was completed. Shows a lack of understanding in the subject matter and very poor execution.
0 [-5]	Not completed.

Marks	3 Marks Criteria [minus]
3 [0]	Task was completed well, adhering to best practices, and followed subject matter guidelines.
2 [-1]	Task was completed satisfactorily. Some features are missing or incorrectly implemented. Shows a moderate level of understanding in the task with room for improvement.
1 [-2]	Some of the task was completed. Shows a lack of understanding in the subject matter and very poor execution.
0 [-3]	Shows a little to no degree of competence in completing the task; not completed.

Marks	1 Marks Criteria
1	Task completed satisfactorily
0	Task was not completed