Software Developer Course Assessment

Quantitative Assessment Practice #1

Course Name: Full Stack JavaScript (Node.js)

Current Week: 13 May 2024

Introduction:

The purpose of this assessment is twofold; first, to help us understand how the class is doing in terms of the course material that we have covered during the previous couple of weeks. The **main** purpose of this assessment is for us to improve our approach to review and ensure that what we're currently doing is an effective teaching strategy for the students. Completion of this assessment is **mandatory - if you don't submit a solution, it will be marked as incomplete. You must complete a minimum of 75% of your assigned QAPs per course – otherwise you will be marked as incomplete for this course.** When you do submit a solution, it will be marked against the provided rubric.

Again, the goal here is to help you all in the best way that we can, so please do be honest when answering the questions related to how long it took, which resources you used, etc. And please ensure that you do your **own** work – don't just copy off a friend to get it done, earnestly do your best with it. If you can't get it completely working, give us what you have. While it will be graded, the grade will not count against you, it's just a way for us to see where everybody is, and to know which concepts, if any, we, as a class, may be struggling with.

Deadline: Your instructor will determine the deadline for submission for the assessment of your solutions. Please ensure you answer all the questions outlined in the instructions portion of this document as well in your submission.

Marking: The core evaluation of this program is marked with one of three possible marks: *Incomplete, Pass, Pass Outstanding.* For QAPs, the individual learning outcome marks are more important for our information as faculty as well as for the information provided to the student. Therefore, QAPs are marked on a scale of 1-5. The details of this marking system are summarized in the table below.

Grade	**Meaning**
1	*Incomplete.* Student shows severe lack of understanding of the material – solution is heavily incomplete, non-functional, or completely off base of what the assignment was asking for.
2	*Partially Complete.* Students show some understanding of the material. Solution may be non-functional or partially functional, but the approach is correct, albeit with some major bugs or missing features.
3	*Mostly Complete.* Student demonstrates understanding of the major ideas of the assignment. Solution is mostly working, albeit with a few small bugs or significant edge cases which were not considered. Shows a good understanding of the correct approach, and is either nearly a feature-complete solution, or is a feature-complete solution with some bugs.
4	*Complete (Equivalent to: Pass.)* Student shows complete understanding of assigned work and implemented all necessary features. Any bugs that are present are insignificant (for example aesthetic bugs when testing the functionality of code) and do not impact the core functionality in a significant way. All necessary objectives for the assignment are completed, and the student has delivered something roughly equivalent to the canonical solution in terms of features and approach.
5	*Complete with Distinction (Equivalent to: Pass Outstanding)* The student demonstrates a clear mastery of the subject matter tested by the QAP. The solution goes above and beyond in some way, makes improvements on the canonical solution, or otherwise demonstrates the student's mastery of the subject matter in some way. A solution in this category would consider all reasonable edge cases and implement more than the necessary functionality required by the assignment.

Instructions:

You are allowed to complete the assessment problems below in whatever way you can but please answer the following questions/points as part of your submission:

1. How many hours did it take you to complete this assessment? (Please keep try to keep track of how many hours you have spent working on each individual part of this assessment as best you can - an estimation is fine; we just want a rough idea.)

2. What online resources you have used? (My lectures, YouTube, Stack overflow etc.)

3. Did you need to ask any of your friends in solving the problems. (If yes, please mention name of the friend. They must be amongst your class fellows.)

4. Did you need to ask questions to any of your instructors? If so, how many questions did you ask (or how many help sessions did you require)?

5. Rate (subjectively) the difficulty of each question from your own perspective, and whether you feel confident that you can solve a similar but different problem requiring some of the same techniques in the future now that you've completed this one.

Tasks #1: Understanding core global objects

Becoming familiar with all the node.js core global objects is an important part of becoming a node full-stack developer. Review the most important node.js core global objects and **select three** to use for your learning within this QAP. Consider the following eleven core global objects as most important as you begin your node mastery journey.

http, events, filesystem, console, buffer, globals, stream, url, path, os, process

Step 1: review all 11 core objects suggested by searching online with "node" and the [global object name] as the search terms.

Step 2: Choose three of the 11 global objects and do a deep dive to improve your understanding of node and these essential objects. Do the following for each object.

> **2.a.** Write a few sentences or paragraph describing each object and how it would be used.
>
> ***Note 1:*** *use a single folder for all the sample files you create for this QAP. Use npm init to prepare the new folder for node.js.*
>
> **2.b.** Write some sample code, in node, to exercise a few of the features of each chosen object. Be sure your sample code writes events to the terminal using console.log();
>
> ***Note 2:*** *each object's sample code should be in its own .js file.*
>
> **2.c.** Be verbose with your inline comments, show your understanding for each line of code.

Step 3: Confirm all of you sample node files can be run, without error, using the terminal. The person marking should be able to open the folder with all your solution files in a terminal and type "node [solution file name]". The marker should see the results written by the console.log() statements from within the solution files.

Tasks #2: Understanding NPM

Understanding how to find and use node modules from the NPM (Node Package Manager) is an essential skill as a node.js programmer. Invest some time in searching the npm site for the features you may need when developing a web site. Be creative when doing NPM searches, think broadly of things / features you would like to see in a website.

Choose a NPM package and install it into a new node project and find an online tutorial (video, blog, o'reilly, etc) to assist you in installing and writing code using the package. Some packages are listed below, feel free to find your own. **Choose a simple package or framework**. This task is mostly about finding and installing a NPM package. So do yourself a favor and find a simple package with an easy sample to implement.

Moment	Parse, validate, manipulate, and display dates and times in JavaScript.	$ npm install moment --save	https://momentjs.com/
Validator	A library of string validators and sanitizers.	$ npm install validator --save	https://www.npmjs.com/package/validator
Lodash	A modern JavaScript utility library delivering modularity, performance & extras.	$ npm install --save lodash	https://lodash.com/
Express	Fast, unopinionated, minimalist web framework for Node.js	$ npm install express --save	https://expressjs.com/
Bcrypt	a library to hash and verify passwords with sync, callbacks, and promise interface.	$ npm install bcrypt --save	https://www.npmjs.com/package/bcrypt
Jest	A complete and ready to set-up JavaScript testing solution	$ npm install --save jest	https://jestjs.io/
Winston	A logger for just about everything.	$ npm install winston --save	https://www.npmjs.com/package/winston
Nodemon	Restart the node application when file changes in the directory are detected.	$ npm install --save --g nodemon	https://nodemon.io/
~~**Colors**~~	~~A library to use colors and styles in the node.js console.~~ Note: this package was corrupted by the main developer in January 2022	~~$ npm install --save colors~~	https://www.npmjs.com/package/colors https://www.bleepingcomputer.com/news/security/dev-corrupts-npm-libs-colors-and-faker-breaking-thousands-of-apps/
Cookie	Basic HTTP cookie parser and serializer for HTTP	npm install cookie --save	https://www.npmjs.com/package/cookie

Step 1: Write some simple node.js code to show you have successfully installed a package from NPM. The code should write some events to the terminal using console.log(). Comment the code to help with understanding of what the software is doing.

Step 2: Confirm your sample node file can be run, without error, using the terminal. The person marking should be able to open the folder with your solution file in a terminal and type "node [solution file name]". The marker should see the results written by the console.log() statements from within the solution files.

Task #3 Using Github for the Project Deliverables:

All your project files should be saved to a single github repository with an easily understood file naming convention. DO NOT include other NPM files or folders as part of your project. **Do not include** the **node_modules** folder (Reminder: use .gitignore). Your QAP will **not** be considered if it contains any additional files as this can make the project zip file > 30 MB in size!

QAP Project Submission:

Submit the completed questionnaire with your project submission. The project submission should include the URL to your github repository for this project. This should be submitted to the assignments in the MS-Team.