Web Science

Quiz 1: February 28, 2023

Enter your answers directly into this document (with the exception of #2 and #3). All answers should be In Your Own Words, using complete sentences with proper spelling and grammar.

Save this document as either a Word docx or a PDF. For all questions other than #2 and #3, you will not receive any credit for answers not placed in this document.

When finished with the quiz, put everything you wrote (this document, all code, etc.) in your personal GitHub repo in a folder named **quiz1**. You must test your code on your VM, served with Node. No Apache!

1. **Short answers** (25 points): (Answer in complete sentences, explain your answers)

    a. (5) What are the potential advantages of using TypeScript's type system as opposed to typeless JavaScript?

    Developers, especially in big teams can really benefit from TypeScript because it provides type safety as well as code readability to the project. Readability is important because it makes the code you write self-expressive and people reading or code reviewing your code will have a much easier time knowing what's going on because type safety makes variables predictable as opposed to in JavaScript where a type can be easily converted to another type carelessly and cause big headaches.

    b. (7) What is a package.json file? What is it used for? How is it created?

    A package.json is a JSON file that exists at the root of a Javascript/Node project. It holds metadata relevant to the project and it is used for managing the project's dependencies, scripts, version, etc. It can be created via npm init or equivalent commands in other package managers.

    c. (3) Of the following permutations, identify which are valid and which are invalid for shipping/installing a Node application: 1. (package.json && package-lock.json); 2. package.json only; 3. package-lock.json only

Shipping/installing a Node application with package.json only, package.json && package-lock.json are valid, with package-lock.json only is invalid.

d. (10) Describe **in detail** the sequence(s) of transaction(s) for a frontend to request data from some external entity via Node.

1. You'll have to set up a Node server.
2. You need to specify the source of the external entity, e.g OpenWeatherMap API.
3. You'll need to specify the request(method, parameters, query strings, API key, request header, etc.).
4. You'll have to specify the specific endpoint your frontend needs to access this data.
5. Your Node server must be running.
6. With that, your frontend can make a request to the endpoint you specified in Node via the fetch API or equivalent tools.
7. Node communicates with the external entity, if successful, returns the data requested and sends to the endpoint in which the frontend had made a request. Some sort of error handling might be useful in this step.

2. **Coding question**: (60 points) Here is a free API that does not require any API keys: http://universities.hipolabs.com/ – the documentation for which can be found here: https://github.com/Hipo/university-domains-list

   Create a new input box (or extend an already created input box) that can accept the name of a university and returns the API's information about that university. If the user fails to input a valid university, return the information for RPI. You might want to extend a recently due lab…

   Creativity matters; you need to really integrate this new information into your app. Make it feel like it is a meaningful, conscious, intentioned feature of your app. How you do that is up to you. Don't make it look like some random afterthought. Go beyond the minimum (but remember that creativity doesn't have to be visual). If you need to, write a short README.md file that tells me what I should consider for creativity. (creativity: 30 points of the 60 available for this question)

   You may use any and all open source libraries you want for this coding question, so long as you cite them in a README.md file.

3. (15) Ensure the package.json file for Q2 has no errors when I run npm install.
   It is ensured.
   :)

4. **Extra credit (+5)**: Why do your URLs need /node/ in order for Node to serve web pages on your VM?

   Because by default everything in the VM is being served with Apache and to tell it that we're using Node server we must specify /node/ in the URL.