Web Science

Quiz 1: March 1, 2022

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: answers.docx (or .odf or .pdf) (-5 if wrong name). 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.) on GitHub into a branch in your lab repo named: quiz1 (-5 if submitted incorrectly). **Do not submit your node_modules folder! (-15 if you submitted the node_modules folder)**

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

    a. (5) How can I determine the type of device that my page is being displayed on? Give two examples of why I might care.

    One can determine the type of device that a webpage is being displayed on by using CSS media queries. One reason why one may care what device the webpage is being displayed on is that the front-end can be displayed quite differently on different screen sizes, say, a smartphone or a computer monitor. By knowing what type of device the webpage is on, we can cater to those differences. Another reason would be to specify that certain styles are only for printed documents or for the use of screen readers, other forms the media can be viewed/consumed.

    b. (5) What is a package-lock.json file? What is it used for? Is it required?

A package-lock.json file is a JSON file created upon installing a package. It is used to record the exact version of every package you installed so you can replicate the environment later. It is not required, but package.json is.

    c. (5) What is npm? How does it work? Why is it used?

    NPM stands for Node Package Manager, and it is used to install node applications, modules, and packages. It is where Node projects are published, and we use it in the command line to keep these Node projects up to date and make installations easier.

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

2. **Coding question**: (40 points) Create a webserver in node.js, name your server: server.js. You may use Express, but you *may not use a generator* – (i.e., NOT express-generator), which will serve a simple frontend (in the technologies of your choosing). The frontend will provide an input field for ZIP code and a series of buttons that issue GET and/or POST requests when clicked to the Node server. (frontend: 10 points)

   Upon entering a ZIP code and clicking the "Temperature" button, your application should send a POST request to http://localhost:3000/temperature. Node should then get the current temperature for that ZIP code (I bet you have an API for that!) and send the frontend back that information. The frontend should then output a sentence that says the name of the location and whether it is Freezing (<33F), Cold (between 33 and 50), Warm (between 51 and 80) or Hot (>80) – display the corresponding message in a unique color for each category. (temperature sequence: 10 points)

   Upon clicking the "Is RPI windy?" button, your application should send a GET request to http://localhost:3000/wind. Node should get wind speed information for Troy, NY, via that API and send that information back to the frontend. Have the frontend display this information in a unique color. (wind sequence: 10 points)

   Creativity matters; don't just give me an empty white page with a text entry form box and two buttons. Go beyond the minimum (but remember that creativity doesn't have to be visual). If you need to, write a short README file that tells me what I should consider for creativity. (creativity: 10 points)

   You may use any and all libraries you want for this coding question.

3. (15) Ensure the package.json file for Q2 has no errors when I run npm install & run your code.

4. (20) Provide **two** different explanations of the code below. The first explanation should be a high-level explanation (no less than four complete sentences) outlining what this code does to someone who has no coding experience. The second explanation should be a *detailed* one explaining line-by-line what the code does. If there are any errors in the code, fix them.

```
var net = require('net');

var sockets=[];

var s = net.createServer(function(socket) {
   sockets.push(socket);

   sockets.on('data', function(d) {
      for(var i=0; i<sockets.length;i++) {
         if (sockets[i]==socket) continue;
         sockets[i].write(d);
      }
   });
   sockets.on('end', function() {
      var i=sockets.indexOf(socket);
      sockets.splice(i,1);
   });
});

s.listen(8080);
```

1. The general use of the code above creates a node server on port 8080 using the net package. When a user, or a socket, connects, it adds them to an array of connected users. When data is entered by a user, it gets written to their position in the array. When the user disconnects, they are removed from the array.
2. Line 1: Defines the variable 'net' (also ensures net module is installed).
   Line 2: Creates an array called 'sockets'.
   Line 3: Sets up a net server, as well as opens a socket.
   Line 4: The new socket gets added ("pushed") to the 'sockets' array.
   Line 5-8: When the socket receives data, pass it in as the variable 'd'. Upon receiving data, iterate through the 'sockets' array. If we are currently at the same socket in the array, go on (hence the 'continue'). Otherwise, write the data 'd' into the array (at socket[i]).
   Lines 11-13: Upon the socket being closed, find the specific socket in the array and close/remove it.
   Line 16: Indicates that the server is listening in on Port 8080.

5. (+5) What is the name of the RPI-developed chat protocol popular in the 1990s?
Naim