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.
    b. To identify a device you can use browser detection- Modernizr is a javascript library that detects for html5 and css3 features (prefer to UA Sniffing). One example of why you may care what type of device your page is being displayed on is so that you can create an app that is easily viewed on both mobile devices and a desktop. Based on what kind of device you have and the size of its display, your html/css may flex or change differently. this is why it is important to use bootstrap and code to account for sizing with different screen types. Another reason that you may care is if you want to provide different html depending on the browser that is being used. Or, if you are trying to check for the existence of a specific feature- which you can do with Modernizer. If your site needs to use a certain web feature that some browser do not support, you would probably want to send the user to a different site with less features or accommodate for the features that are unable to run on that device.

    c. (5) What is a package-lock.json file? What is it used for? Is it required?
    d. package-lock.json records the exact version you install so that you can reproduce the exact environment later. You cannot ship an app with only the package-lock.json file. It is required if you want all installations to remain identical. Package-lock.json is always generated for operations where npm modifies package.json

e. (5) What is npm? How does it work? Why is it used?
f. NPM is a package manager for Node JS libraries. it works by being asynchronous event driven (http closes after every event but node stays open) it is single thread, non blocking, and it has even loops and call backs. Non blocking means that two files can load at the same time. NPM is often bundled with node, and it is used to install node applications, modules and packages. this is the npm install command. It helps to manage dependencies, and it has a ton of packages.

g. (10) Describe **in detail** the sequence(s) of transaction(s) for a frontend to request data from some external entity via Node.
h. you can make http requests for node js, and you can use JSON APIs to show this. the default http module is in the standard library. with this module, you can plug and go without having to worry about installing other external packages or dependencies. this is the const http= require(https) and httpg.et("link")... lines. that code will send a get request to the api link stored in the "link" portion of the http get request. Once the dtaa is received you can use resp.on('data',(chunk)... to retrieve that data that has been received from the api call. you also need to parse the response data manually. you need to required the https module if the api is communicating over https. You have to run npm install in the terminal to download the request library, and install it as a dependency from npm. you can write the lines onst request= require('request') and request("link",{josn:true}... and this will retrieve all of the data.

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.
    a. it has no errors when i run it

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.Server(function(socket) {
   sockets.push(socket);

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

s.listen(8080);
```

1. the net module is providing an asynchronous network api for TCP or ICP servers. it can be accessed in the first line of code. the second line of code is creating an empty array called sockets. when s.listen is called on the last line to listen on the server, the net.server function pulls that data from the req header from the server, and if sockets[i] is equal to that data that is being pulled, then you continue through the loop for the length of the socket data. Otherwise, socket[i] becomes the data that is pulled from socket (denoted as d). At the end of the connection, sockets is spliced.
2. the server is connected to the node js code, and there is an array created that is blank. the .net function pulls information from the socket which is a connection with the server and user side, and the data from the socket is loaded into the array if it is not the same (if it is the same it has no contents and it continues through the loop). At the end of the loop the connection is cut.

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