Web Science

Quiz 1: February 26, 2018

100 points max

Place your name on the top of the document in the header

Enter your answers directly into this document (with the exception of #2 and #3)

All answers should be in be in Your Own Words, and use proper grammar

Make sure your answers use an alternative font and/or color

Save the document as

ITWS4500-S18-Quiz1-*yourname*-quiz1.docx

Place all documents/files including this one in a folder named

ITWS4500-S18-Quiz1-*yourname*-*yourRCSID*

When finished with the quiz, zip your folder and all related files into a file named

ITWS4500-S18-Quiz1-*yourname*-*yourRCSID*.zip

And submit it to LMS

1. **Frameworks** (25 points): (Answer in complete sentences, explain your answers)
   1. (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 of the easiest ways, especially when working with CSS, is to use media queries. Media queries allow you to load certain things in the styles when the page loads based on properties of the window the page is being displayed on. The first example of why you should care is for styling; if a web page isn’t written to work on multiple devices, the page might be confusing or hard to use for a user trying to use the page. Second, some devices, particularly devices with touch screens, are limited as to what capabilities you can use; for example, touch screens do not interact well with the hover feature of CSS, so you may find alternative ways to include that hover feature for mobile.

* 1. (5) What is a package.json file? How is it used? What it used for? Is it required?

A package.json file is the configuration for a certain node application. It includes things such as the application’s dependencies (packages), command shortcuts (i.e. “npm start” versus “node server.js”), the application’s name and description, and the application’s version. This is not required, but it is required if you are using any dependencies (i.e. any node packages). You can create it either by doing it yourself or using the configuration by typing “npm init” in the goal directory.

* 1. (5) What is nvm? How does it work? Why is it used?

NVM is the Node Version Manager and it helps the user manage multiple versions of Node on one computer / server.

* 1. (10) Describe the difference between Front-end and Back-end frameworks. Provide at least 2 examples for each in your answer. (Be clear in your descriptions, ie ‘why is it back/front-end?’)

Front-end frameworks are generally used for styling pages and modifying the user’s view, and they are rendered / run by the client or browser. Two examples of this are Bootstrap and Angular. Back-end frameworks are used for serving requests for resources from the client, and it runs on the server-side. This is advantageous because the program’s code is not visible to the client, so you can store passwords and keys securely. Two examples of this are ExpressJS and Ruby-On-Rails.

1. **Node.js** : (40 points) Create a webserver in node.js, name your server *yourRCSid-Quiz1Server.js)* you may use express, but you *may not use a generator* – (ie NOT express-generator), which will serve a simple HTML page with an input field for zipcode and a button labeled ‘Run’ when a GET request is received on http://localhost:3000. Upon entering a zipcode and clicking the button, the server should get the current temperature for that zipcode and output a sentence that says the name of the location and whether it is Freezing (<=0C), Cold (btw 0 and 10), Warm (btw 11 and 25) or Hot (>25) – display the corresponding message in a unique color for each category. Allow the user to enter additional zipcodes for weather in other locations. Display the new sentence above the previous. Include a button that allows the user to refresh the page and start over.

Complete

1. (15) Build a package.json file for Q2. If we run it, there should be no errors or warnings when we try to install & run your code from #2 above. (You should make your application name : *yourRCSid-Quiz1Server.js*)

Complete

1. (20) Explain *in detail* what the following code does; (also add *stylized* comments to the code explaining what each line does, and highlight and correct any errors)

var net = require('net'); //- Loads the node package ‘net’ into the variable named ‘net’.

var sockets==[]; //- Assuming ‘==’ is a typo and is actually supposed to be ‘=’, this initializes and empty array called ‘sockets’. If it’s not a typo, this will be an error.

var s = net.Server(function(sock) { //- Using the net.Socket function, this connects to the server using a socket named ‘sock’.

sockets.push(socket); //- This adds the socket that was opened in the previous line to the array ‘sockets’ (assuming ‘socket’ is supposed to be ‘sock’. If not, this will throw an error)

socket.on('data', function(d) { //- This listens for a new socket

for(var i=0; i<sockets.length;i++) { //- Loops through the sockets

if (sockets[i]==socket) continue; //- If the socket is the current socket, we don’t print data (again assuming ‘socket’ should be ‘sock’)

sockets[i].write(d); //- Writes the data received on that socket to the other sockets in this code

}

});

socket.on('end', function() { //- When the socket doesn’t have anything left to read, do the following

var i=sockets.indexOf(socket); //- Again assuming ‘socket’ should be ‘sock’, get the index of ‘socket’ in ‘sockets’

sockets.splice(i,1); //- Remove the socket from the sockets array.

});

});

s.listen(8080); //- Listen on port 8080 for new sockets

This uses the net.Socket class to allow users to connect to a sort of chat server (it looks like) running on 8080. What it looks like is that data is transferred from one socket to all others in the array.