CS275 Web and Mobile App Development
Assignment 3

As with this and all future assignments, you must work individually.

Objective:

The purpose of our initial server side CS275 assignment is two-fold:

1. Set up and test your server side environment
2. Get acquainted with Nodejs basics, including interactions between clients and server along with the development of a simple dynamic web service.

Part 1: Selection and Configuration of Server Side Development Environment

We will be using Nodejs as our server side JavaScript engine and need to install it along with its companion Node Package Manager (NPM) on your laptop (the latter of which should be automatically installed alongside Nodejs by default).

Nodejs installers for a variety of operating systems can be found at
www.nodejs.org (Downloads)

Be sure to install the version specific for your operating system (Windows, Macintosh, Ubuntu, others). Although hopefully the installer will walk you through the process, issues particular to each OS can be found and resolved via a quick (Google) literature search. Feel free to share your issues and solutions on Piazza to help others!

Once the installer has been downloaded and run, it is a good idea to test the setup. Note that these installers should provide both Node and NPM.

1. First, it will (likely) be necessary to restart your computer.
2. Test Node - open a command line and type
   ```
   node –v
   ```
   This should print the version number
3. Test NPM – from the command line, type
   ```
   npm –v
   ```
   This should print NPM's version number
4. Create a simple test file (perhaps the introductory example on page 7 of the Node – Part 1 slide set) to test your environment
   - Create a JavaScript file (someFileName.js)
   - Make sure an appropriate `console.log` command is included
   - From the command line, type:
     ```
     node someFileName.js
     ```
   - You should see the content of the console.log command as output

Part 2: Developing an Initial Web Service App

For this assignment, we will develop a client– server app to request, process and display results for a variety of simple mathematical calculations, in particular:

- n-Factorial
- Summation series (add all integers from 1 to n)

Your client side web page will need the following input elements:

- An input box to specify a value for the seed (n)
- A drop down box to enable selection of either of the 2 calculation options (please research the literature if this feature is new to you)
- A button to launch the overall process once inputs have been made

Client side processing should include the following steps:

- Read in the seed value and option and then set up the URL (use localhost:8080) along with the selected option
- Make an Ajax call :
  - `type = “GET”`
  - URL should include both the localhost front end along with a reference to the option chosen
    - eg. http://localhost:8080/sum or http://localhost:8080/fact
  - `data` ==> the seed value n (as a JSON object)
  - `dataType` = html
  - etc.
- Upon a successful return, populate a `div` with the associated answer
  - eg. The factorial for 5 is 120.

For the **server side** processing:

- “`require`” the necessary modules – `express, http`, others?
- Create a pair (one for each calculation) of `app.get` actions to process each calculation option. Each will include a function with parameters `(req,res)` to perform the calculation and send back the result `res.send(text for the result)` to the client
- Make sure that your web service is set to listen for traffic on port (we’ve been using 8080).

Behaviors to include:

- Test of empty or negative values for `n`

Development Hints

- **Make sure you're loading your pages from the server and not just clicking on them to open them as files (you should type in something like http://localhost:8080/hw3.html into your browser not click on a file called hw3.html from a file explorer).**
- You could first develop the client and server side independently
  - Write your server-side endpoint handler functions and just test them by putting in a browser something like
    http://localhost:8080/sum?n=5
  - Write your client-side code so that it just "`alerts`" you (or prints to the console) as to what URL is constructed.
- Then ultimately integrate the two entities to complete the request / response "loop"
- To debug on the server side, include `console.log` statements as necessary
- It is necessary to "kill" the Node process after making revisions to your server script since the "old" process may still be attached to the port. Depending on your OS you might try things like:
  - Ctrl+C
  - Task Manager

What to Submit

- A screen cast video to detailing a thorough (both client and server side) code walk through, followed by a demo of all cases (both calculations + empty and negative inputs for n)
- Your source code, well internally documented
- README file on how to run your code
- All of the above **ZIPPED** together and that single compressed file uploaded to BBlearn.

Grading (50) Points

- 40 points: program correctness along with adherence to the stated requirements
- 5 points: quality of internal documentation and code style
- 5 points: README file