COMP 1537 – Assignment 4

Due: The week of October 31st, before the beginning of your scheduled lab class, for marks! This assignment is out of **10**.

This assignment must be done individually.

Objective

For this lab, you are going to get acquainted with Node.js, or simply known as Node. Node is a server environment that allows you to run JS on the server-side. This can streamline your development process since you are using the same language on both the client as well as the server. As well, Node.js comes with a plethora of modules for various features. Windows users, use this link to install Node.js:

https://nodesource.com/blog/installing-nodejs-tutorial-windows/

Mac OS users, follow this link to install Node.js:

https://treehouse.github.io/installation-guides/mac/node-mac.html

You might be asking why you need to already start using Node this early. After all, you haven't even completed one JS assignment and we are already moving to the server side. The most important reason is that web browsers have this policy of not allowing Cross Site Scripting (XSS). If you write some JS code for the HTML web document and attempt to refer to files (e.g., an image, more JS files, another HTML document, etc.), the browser will block you from doing this. This is good since it protects our computers from malicious scripts that may attempt to scan through our file system and perform actions that we are not aware of or want.

So, in order to successfully retrieve files programmatically with JS, we need to be running from a server environment. This requires us to be running Node – which acts as a web server for us and allows our JS to refer to the files that are on the server.

! **Note:** when you load your HTML documents in the web browser, do not refer to them from the file system directly. So for example, this would be the incorrect way to refer to the file:

file:///C:/apps/myapps/node-template/public/html/index.html

While this will load the file for you, this file will be considered locally loaded and therefore will block your JS from accessing files, such as images, from the file system. Refer to your HTML document from the server, such as this:

http://localhost:8000/

! **Note:** Code editors such as Microsoft Visual Studio Code, the late Adobe Brackets.io, and the rest do have launcher plugins. **Avoid using these** and get yourselves acquainted with the console/command line/terminal so as to avoid odd behaviors and tool misconfigurations.

! **Note:** On some Macintosh computer configurations, the alias "localhost" may not work. What you can do instead is to refer to the actual IP address given which is http://127.0.0.1:8000

Once you've set up Node, you'll be able to run your apps in the console (Windows) or terminal (Mac OS).

Play-By-Play Action in Node

Acquaint yourself with the Node example that is provided (called express-routing.zip). It has several files:

- index.js: the server-side code that Node runs as an application
- index.html: the HTML document that acts as the main page for your application, and is sent to the web browser by index.js
- style.css: the style for your index.html HTML document
- client.js: the JS file that acts as the programmatic part of the client-side of your application, and that also contacts the server asking for more content (the newsfeed.xml document – for now)
- newsfeed.xml: a snippet of HTML code, in this case an unordered list

This basic template has very little code in it but so much is going on within it. It demonstrates the basics of what happens with a client-server application which is depicted in figure 1.

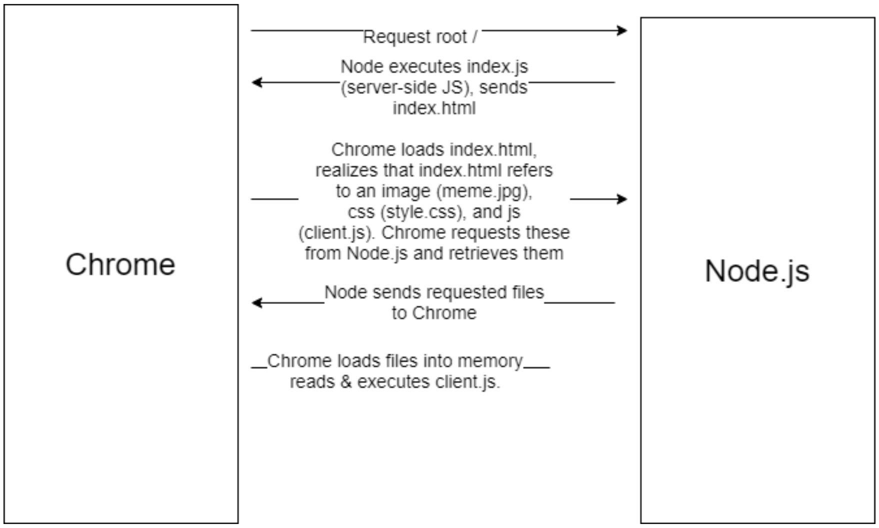
Chrome
Node.js
Request root /
Node executes index.js (server-side JS), sends index.html
Chrome loads index.html, realizes that index.html refers to an image (meme.jpg), css (style.css), and js (client.js). Chrome requests these from Node.js and retrieves them
Node sends requested files to Chrome
Chrome loads files into memory reads & executes client.js.

Figure 1 - Basic Node Template, Client-Server Interaction

A lot just happened there! And this is just a basic template! But it demonstrates the goings on within a web app. Study this example and understand this process. Ask questions in lecture and lab because it is imperative for you to know how this process works.

This assignment looks easy and once you understand both virtual paths as well as actual paths, configuration, and how files are served up, you will realize that it is an easy assignment! Your job is to take your previous assignment (assignment 3) and serve it up using Node.js

> **! Note:** You will need one module from the node package manager: express. You can get that module by typing at the console/command line/terminal the following:
>
> ```
> npm install -g express
> ```
>
> That should install express in your Node install directory. If for some reason this fails, you may have to remove the “-g”. This usually happens on Microsoft Windows with Home versions of Windows 10. The cause is due to your account not having administrative privileges.

Requirements

You will need to adhere to the following requirements for this submission:

- Serve up on port 8000 only
- Use the same directory structure that the example uses
- Your Node.js server app file must be called assignment4.js
- Your main page is called index.html
- Your web template/design must contain at least:
    - One CSS stylesheet that is external and referenced by your html document
    - One JS document that simply prints to the command line that it was loaded
    - Two images that are in your app and not hot linked to other sites (i.e., all your image resources are local)
    - At least one local font via @font-face
    - A favicon.ico – do not give me back mine! Change this to something else that is yours and specific to your design!

! Note: You won't need any of the files found in the data sub-directory … for now, that will come up in the next assignment. You also won't need the profile.html document – or any other HTML documents either. That will be the next assignment.

Once you have completed, ensure that your HTML document is valid. It is suggested that you use the HTML 5 validator online at:

https://html5.validator.nu/

To ensure that you don't lose marks. Also use the CSS validator in Visual Studio Code (CSS Formatter & HTML CSS Support)

As well, include a readme.txt file that follows this format:

```
[Student Name], [Student ID], [Set], [Date]


This assignment is [enter percent]% complete.


[explanation if not complete, what is working/not
working]
```

Figure 2: readme.txt

Submission

Create a zip archive of any and all text files that are part of your assignment submission. Your HTML files will have the extension “.html”, your CSS files will have the extension “.css”, and your JavaScript files will have the extension “.js”. If you have images, be sure to include them in this archive that you create.

Once you create your zip archive file, rename it to follow the format SurnameFirstnameCOMP1537Assignment4. My zip file would be FergusonArronCOMP1537Assignment4, for instance.

Finally, submit it to the learning hub in the folder labeled “assignment 4”.