Lab 3: Node Setup
Due: 2/13/2020 11:55pm
25 Points

Purpose: We are going to get a hello world node.js up and running. After we have successfully tested our hello world application, we will build a very basic web server using Node.js. The web server will simply listen for requests and try to find files in the file system that match the resources requested. If one is found, it will be returned and correctly displayed to the client based on it's type (HTML, image file), if the file is not found, the user should get a friendly error. This will serve 3 main purposes at this stage of the class.

1. It will help solidify some of the linux terminal knowledge you have just learned while it is fresh and help you get to know your server environment better. As we will do some work in the terminal
2. It will introduce you to the basics of getting a Node.js service up and running, you will be working with Node on your local development machine (your laptop or lab computer).
3. Because Node is so lightweight on it's own, our Hello World application will actually require you to write code to handle the incoming request and create the response. Normally applications that are deployed on the web use an existing service for this, such as Apache or Tomcat. But with Node we get the cool experience of working directly with the HTTP.

Preparatory: You will just need an internet connected computer for this Lab, but you will want to either have a computer with a Linux / UNIX (mac) terminal (GitBash for windows). This lab goes through the setup process in great detail, but does not touch on how to install GitBash.

Submission: When your lab is complete, you will submit the lab assignment in iLearn, submit the following in your iLearn assignment.

The URL of your github repository for the lab

Step 1: Procure Node

Visit https://nodejs.org/en/ and go to the 'downloads' page

- Look for the 64-bit download for your platform

Step 2: Install Node on your machine

Follow the documentation on the Node website to install node on your development machine

Step 3: Create a github repository

This will be done for Lab 3 just like you did for Lab 2. The name should be formatted as: bgormanly-lab3 where bgormanly is replaced by your first initial, last name. The repository should be created with a .gitignore file for node and the repository should be private.

Once you have created the lab3 repository in github, clone it down to your computer in the directory you wish to work in.

Step 3: Create Our Web Server

Your cloned project directory will serve as the directory we will work in and since we are using a non-privileged port, you can (and should) run your Node.js application server as a non-privileged user (your default user will work, you do not need to change to root or run with root privileges). In my examples, I will assume my cloned folder is a child of my home directory and it is called bgormanly-lab3/

```
~/bgormanly-lab3
```

For now to keep things simple you will have 2 files in your folder, named **server.js** and **index.html** you can create them with your favorite editor on the server or on your local dev machine and transfer them when done, your choice.

Hello World:
You can create this right on the server if you are familiar with editors like vim or nano or you can open the files with a text editor of your choice.

- Open the server.js we just created and put the following code in it:

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

http.createServer(function (request, response) {
    response.writeHead(200, {
        'Content-Type': 'text/plain',
        'Access-Control-Allow-Origin' : '*'
    });
    response.end('Hello World\n');
}).listen(1337);
console.log("Server running on port 1337!");
```

- Save and close the file.
- Now we have to learn a little bit around running processes in Linux, etc.

- First, we will just run the Node.js application.

```
$ node server.js
```

- You should see your log message at the end of the file: “Server running on port 1337!”.
- Open your browser and go to http://localhost:1337
- If everything went well you should see the ‘Hello World!’ message in your browser, if so congratulations! You can keep going, if not you have to fix this before you can go on, go back through the lab (especially the part about opening the firewall).
- You can end the node process by going back to your terminal where it is running and using ctrl-c to end the process.
- Congratulations on getting this far but we are not done yet.

Step 4: Expanding our Web Server

The next (**and largest**) step is going to be your own, and is the code you will turn in for this lab. You are going to use Node.js to build a basic web server. It will be able to receive an HTTP request, take appropriate action by finding a file and return an HTTP response to the client! I am going to give an introduction during lab, but much of your success will depend on your ability to research the problem and find solutions.

I recommend starting by reading and trying code from the link below. Please note: the information you require is not all on the first page of the document, use the next at the bottom to progress through different parts: https://www.w3schools.com/nodejs/nodejs_http.asp

Important! When researching this piece you must use solutions that are using the http library

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

You may not use express or another web application framework.

1. **Your basic node server will be able to handle both a request for a page that exists (your index.html file) and one that does not** (I am going to try typing in the name of a random resource, like brian.html).
    a. To do this you must be able to **Parse the URL** of the requested resource (if you go to localhost:1337/somepage.html, you need to try retrieve somepage.html!).
    b. Check to see **what type of file** they are requesting (by looking at the extension of the file)
    c. Return the file to the user by setting the correct **Content-Type** for the file and writing it to the response
    d. If the page requested does not exist (another page is requested besides index.html) you should display a “Not Found” message it can be in the server.js code or you can **include a 404.html file** that the server.js returns, your choice.

2. You should also be **able to return 1 image**, and to get full credit your server must return the image in the response with the **correct content-type in the header**! (even if it works without it!)
    a. Add the image in the same folder as your index.html file
    b. Include an anchor in the HTML file that shows your image on the page

You can also find more detail in this Node.js document: https://nodejs.org/api/http.html#http_http_createserver_requestlistener It has a great deal of code and information about creating a basic Node.js web server.

Note 1: For this Lab (and this lab only unless I specify otherwise in the future) I am promoting you sharpening your google skills to **find existing code** on the net that handles the **server.js** part of this assignment (note I did not say I was promoting getting it from other students). The index.html page must be your own! We have not spent any time yet on JavaScript, but there is no reason that your existing Software Dev 1 skills combined with some skillful Googling and some changes to make it work with your port, file names, etc. cannot yield a working lab!
Note 2: Remember you must **restart your web server after you make changes** to your files for the changes to show!

When complete please add, commit and push your code in git and **paste the link to your github lab url into the iLearn assignment!**