How the Web Works

In this lab, you'll be working with a partner to explore a little more about the internet, the web, requests, responses and more. You'll be reading and writing about concepts as well as practicing some of the commands that we saw during the lecture earlier.

Topic 1: The Internet and the World Wide Web

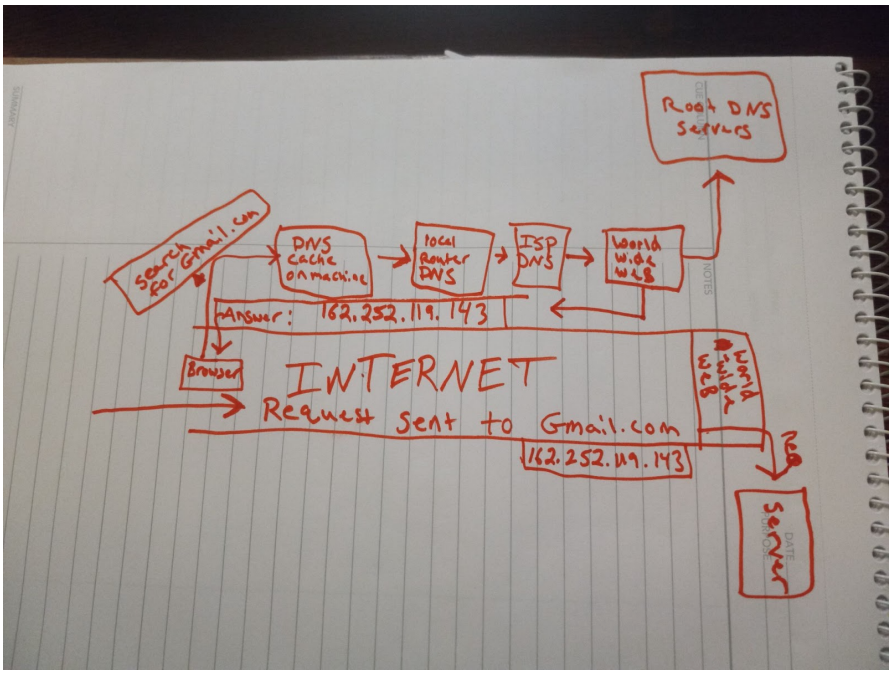
1) What is the internet? (hint: here)
   The Internet is a worldwide network of networks that uses the Internet protocol suite (also named TCP/IP from its two most important protocols).

2) What is the world wide web? (hint: here)
   The *World Wide Web*—commonly referred to as **WWW**, **W3**, or **the Web**—is an interconnected system of public webpages accessible through the Internet. The Web is not the same as the Internet: the Web is one of many applications built on top of the Internet.

3) Partner One: read this page on how the internet works, Partner Two: read this page on how the world wide web works. When you're done reading, come back together and and answer the following questions
   a) What are networks? Two or more computers connected to each other either physically or wirelessly.
   b) What are servers? Servers are computers that store webpages, sites, or apps. When a client device wants to access a webpage, a copy of the webpage is downloaded from the server onto the client machine to be displayed in the user's web browser.
   c) What are routers? Special tiny computer that acts as a signaler making sure that a message sent arrives at the right destination.
   d) What are packets? Series of a small chuck of website files.

4) Come up with a metaphor for the internet and the web, you can do a single one if you think of one that puts them together or two separate ones (feel free to use one you've heard today or read about if you can't think of a new one, but spend at least 10 minutes trying to think of something different before you resort to that) The internet is the national Highway system and the world wide web is the roads and highways almanac.

5) Draw out a diagram of the infrastructure of the internet and how a request and response travel using your metaphor (like the map and letters we saw during the lecture). Insert the drawing into this document (can be a picture of a physical drawing, a Google Drawing, a Figma drawing, etc)



Topic 2: IP Addresses and Domains

1. What is the difference between an IP address and a domain name? Ip address is an actual set of numerical instructions. It is a unique address to find that computer on the network.
   The domain name functions as a link to the IP address. Link do not contain the actual information, but they do point to the place where the IP address information resides.

2. What's devmountain.com's IP address? (Hint: use 'ping' in the terminal)
   104.22.12.35

3. Try to access devmountain.com by its IP address. It shouldn't work because we have our sites protected by a service called CloudFlare. Why might it be important to not let users access your site directly at the IP address? To protect our site from hackers and other criminals (dns attacks, dos attacks,brute force attacks.)

4. How do our browsers know the IP address of a website when we type in its domain name? (If you need a refresher, go read this comic linked in the handout from this lecture)
   IP addresses are used to distinguish one connected computer from others on the tcp/ip connection.
   URL's represent the address for a certain file on the tcp/ip network globally.
   <the method of protocol to be used://the domain name/the directory name/the file name>

Topic 3: How a web page loads into a browser

The steps of how a web page is requested and sent are in the table below. However, **they are out of order**. Unscramble them and explain your thinking/reasoning in the second two columns of the table.

Steps Scrambled	Steps in Correct Order	Why did you put this step in this position?
Example: Here is an example step	*Here is an example step*	*- I put this step first because ____* *- I put this step before/after ____ because ____*
Request reaches app server	2	I put this before app code finishes because the requests initiate the app process
HTML processing finishes	5	I put this after “browser receives html, begins processing” because this finishes the page rendering process.
App code finishes execution	3	I put this after “request app server” because this executes the request provided to the app
Initial request (link clicked, URL visited)	1	Starts the process. Original action
Page rendered in browser	6	This is the end result thus I put it last
Browser receives HTML, begins processing	4	I put this after “app code finishes execution” because

Topic 4: Requests and Responses

Setup

- Download the folder for this exercise from Frodo.
- Make sure you unzip it.
- Open it in VS Code
- Run `npm i` in the terminal (make sure you're in the web-works folder you just downloaded).
    - You'll know it was successful if you see a node_modules folder in the web-works folder.
- Run `node server.js` in the terminal (also in the web-works folder) and you should see a log to the terminal saying 'serving up port 4500'
- You'll be using this file to figure out what will happen when you make requests to this server, so read it over to see what's going on. We'll be getting into the two GET functions and the POST function.

Part A: GET /

- You'll start by looking at the function that runs when we make a get request to /, which looks like this: http://localhost:4500 or http://localhost:4500/
- You'll use the curl command to make a request and read the response in your terminal

1) Predict what you'll see as the body of the response: Jurrni, journaling your journeys.
2) Predict what the content-type of the response will be: HTML h1 and h2

- Open a terminal window and run `curl -i http:localhost:4500`

3) Were you correct about the body? If yes, how/why did you make your prediction? If not, what was it and why? Yes. I saw it is written in the code as h1 and h2.
4) Were you correct about the content-type of the response? If yes, how/why did you make your prediction? If not, what was it and why? yes. I knew it would be an HTML data.

Part B: GET /entries

- Now look at the next function, the one that runs on get requests to /entries.
- You'll use the curl command again. This time, you'll need to figure out how to modify it to get the response that you need. curl -i http://localhost:4500/entries

1) Predict what you'll see as the body of the response: the entries object – an array of ids.
2) Predict what the content-type of the response will be:

- In your terminal, run a curl command to get request this server for /entries

3) Were you correct about the body? If yes, how/why did you make your prediction? If not, what was it and why? yes.
4) Were you correct about the content-type of the response? If yes, how/why did you make your prediction? If not, what was it and why? yes.

Part C: POST /entry

- Last, read over the function that runs a post request.

1) At a base level, what is this function doing? (There are four parts to this)
2) To get this function to work, we need to send a body object with our request. Looking at the function in server.js, what properties do you know you'll need to include on that body object? And what data types will they be (hint: look at the objects in the entries array)? Id (number), date (string), content (string).
3) Plan the object that you'll send with your request. Remember that it needs to be written as a JSON object inside strings. JSON objects properties/keys and values need to be in **double quotes** and separated by commas.
4) What URL will you be making this request to? localhost:4500/entry
5) Predict what you'll see as the body of the response: an array of entries and the new entry

6) Predict what the content-type of the response will be: an array with code formatting.
- In your terminal, enter the curl command to make this request. It should look something like the example below, with the information you decided on in steps 3 and 4 instead of the ALL CAPS WORDS.
    - curl -i -X POST -H 'Content-type: application/json' -d JSONOBJECT URL
- `curl -i -X POST -H 'content-type: application/json' -d '{"date": “August 16", "content": "This is my entry"}' http://localhost:4500/entry`
    -
7) Were you correct about the body? If yes, how/why did you make your prediction? If not, what was it and why? yes
8) Were you correct about the content-type of the response? If yes, how/why did you make your prediction? If not, what was it and why? yes

Submission

1. Save this document as a PDF
2. Go to Github and create a new repository. (Click the little + in the upper right hand corner.)
3. Name your repository “web-works” (or something like that).
4. Click “uploading an existing file” under the “Quick setup heading”.
5. Choose your web works PDF document to upload.
6. Add “commit message” under the heading “Commit changes”. A good commit message would be something like “Adding web works problems.”
7. Click commit changes.

Further Study: More curl

Visit this link and do the exercises using the website provided. Keep track of the commands you used in this document. (Don’t forget to resubmit to GitHub when you complete this section)