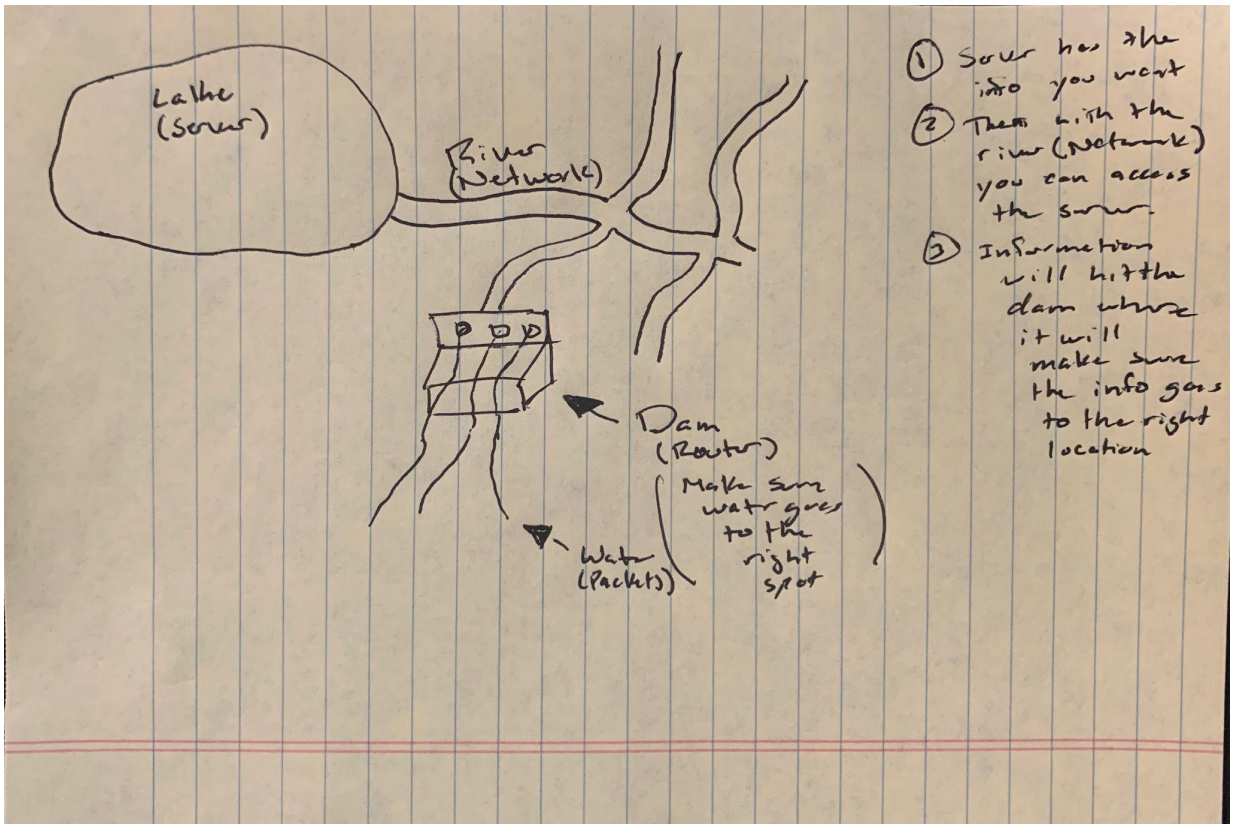
How the Web Works

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) a global network that connects with other networks.
2. What is the world wide web? (hint: here) an interconnected system of web pages accessible through 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? Computers that are linked and can send information
    b. What are servers? Computers that store webpages, sites, or apps.
    c. What are routers? Computers that ensure the right information is being sent from one location to the proper recipient.
    d. What are packets? Data chunks being sent from the server to the client.
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)
    a. The internet is like a river and the web is like the water that flows through it. Lakes are like servers that hold water using dams.
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)
    a.



Topic 2: IP Addresses and Domains

1. What is the difference between an IP address and a domain name?
    a. Domain is a simple nickname given to a website that substitutes the IP address numbers.
2. What's devmountain.com's IP address? (Hint: use 'ping' in the terminal)
    a. 172.66.40.149
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?
    a. It adds a layer of protection against known and unknown risks.
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)
    a. Links everything together in the DNS and stores for future use.

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	Everything has to connect to the server before you can get any information.
HTML processing finishes	5	HTML can't finish if the HTML process hasn't started
App code finishes execution	3	Has to come after the initial server connection
Initial request (link clicked, URL visited)	1	Because an initial request is required.
Page rendered in browser	6	Because at the end of the process the page loads in.
Browser receives HTML, begins processing	4	Before a page is displayed it needs to request download.

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: shows information about the websites
2. Predict what the content-type of the response will be: everything will be json

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

1. Were you correct about the body? If yes, how/why did you make your prediction? If not, what was it and why?
    a. No, it was what appeared on the screen as a user.
2. 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?
    a. Yes, and No: I didn't think it would print out all the extra information like the date I accessed it

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.

1. Predict what you'll see as the body of the response: I think we'll see all the journal entries that have been made.
2. Predict what the content-type of the response will be: I think it will be text/html

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

1. Were you correct about the body? If yes, how/why did you make your prediction? If not, what was it and why?
    a. Yes, looking up in the code and saw the "let entries" section
2. 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?
    a. No, application/json: because the computer is compiling everything together and taking out all the un needed spaces.

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)
    a. Starts new entry with the the entry #, date, and content, adds to entries array
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)?
    a. You will need to enter in the date and content as strings.
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? /entry
5. Predict what you'll see as the body of the response: You'll see your new journal entry
6. Predict what the content-type of the response will be: application/json

- 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

1. Were you correct about the body? If yes, how/why did you make your prediction? If not, what was it and why?
    a. Yes, just by looking at the code listed above.
2. 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?
    a. Yes, I don't know why but its just how part B came in as.

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)