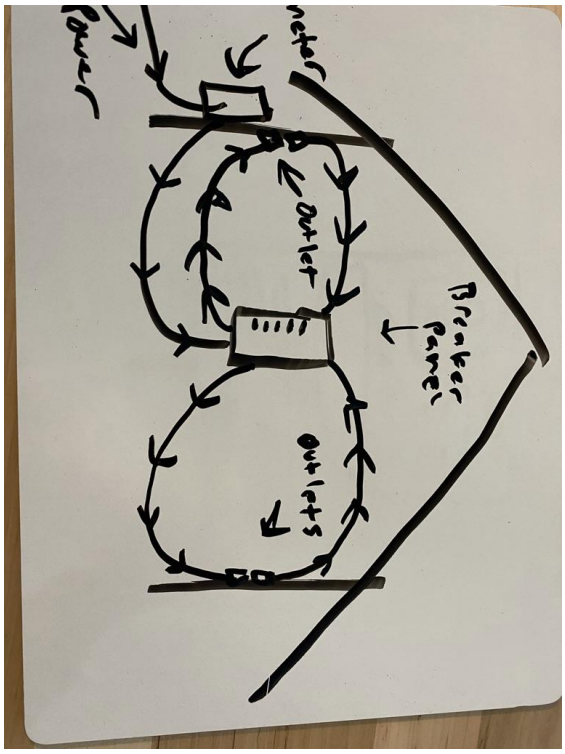
How the Web Works

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 global network composed of smaller local networks.**
2) What is the world wide web? (hint: here)
    - **The world wide web is a directory of web pages accessible through the internet. It is an application that is usable 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?
        - **A connection that links multiple computers so that they can communicate with each other.**
    b) What are servers?
        - **Servers are computers that store web pages, sites, and apps.**
    c) What are routers?
        - **A router is a tiny computer that ensures that computers are communicating with the proper destination computer.**
    d) What are packets?
        - **Packets are small chunks of data that are used to transmit data across the web.**
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 wiring and generators that make up the electric system in your house. The world wide web is the light switches that use this functionality to generate output for it.**
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?
    - **The IP address is the specific location of a website on the network. The domain name is a simplified name that makes it easier for people to connect to the website without remembering the IP address.**
2) What's devmountain.com's IP address? (Hint: use 'ping' in the terminal)
    - **176.66.40.149**
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?
    - **This is important for cybersecurity. This type of protection can help mitigate DDOS attacks and other cyber malware.**
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)
    - **The domain name is turned into the IP address by the DNS**

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 second, because once you call a web page it has to reach

		the web page.
HTML processing finishes	5	After the HTML processing begins, the next step needs to be for it to finish processing.
App code finishes execution	3	The web page's code needs to run in order for it to begin to handle your request.
Initial request (link clicked, URL visited)	1	I put this step first, because the first step has to be to call to the web page.
Page rendered in browser	6	The final step is for the web page to render it's final output.
Browser receives HTML, begins processing	4	Now the webpage is sending it's information to your browser in the form of html.

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:
    - **I predict we will see it return "Jurrni Journaling your journies"**
2) Predict what the content-type of the response will be:
   -The content type will be html/text

- 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?
    - **We were correct. We made this prediction because that was what the code was sending in this section.**
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?

    - **We were correct. We made this prediction because the output of this page was just a text string.**

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 predict we will see the entries array displayed as the body of the response.**
2) Predict what the content-type of the response will be:
    - **I predict the type will be html/text**

- 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?
    - **We were correct about the body. We made this prediction because it was the result that was shown at the /entries section of the code.**
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?
    - **We were not correct. The type was application/json. This is because the /entries section of the code stored json objects.**

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)
    - **This function posts to the server. This function also iterates the ID. This function also creates a new object called newEntry. This function also pushes the new object to the 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)?
    - **To send the body object we will need to send date and content properties. These properties will also be strings. We will also send over the id property. However we do not need to enter this ourselves as the function is already iterating.**
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.
    - **-d '{"id":"globalid", "date":"September 27th", "content":"dogs"}'**
4) What URL will you be making this request to?
    - http://localhost:4500/entry
5) Predict what you'll see as the body of the response:
    - **The body should show the entries array, however it will also show the new JSON object that we sent with our request added to the end of the array.**
6) Predict what the content-type of the response will be:
    - **The content type will be application/jsoc**

- 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

7) Were you correct about the body? If yes, how/why did you make your prediction? If not, what was it and why?
    - **We were correct about our body prediction. We made this prediction based on identifying the function's goals, and understanding what it was going to be posting.**
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?
    - **We were correct. We identified the type was going to be application/json because that was what we were going to be posting to the server.**

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)