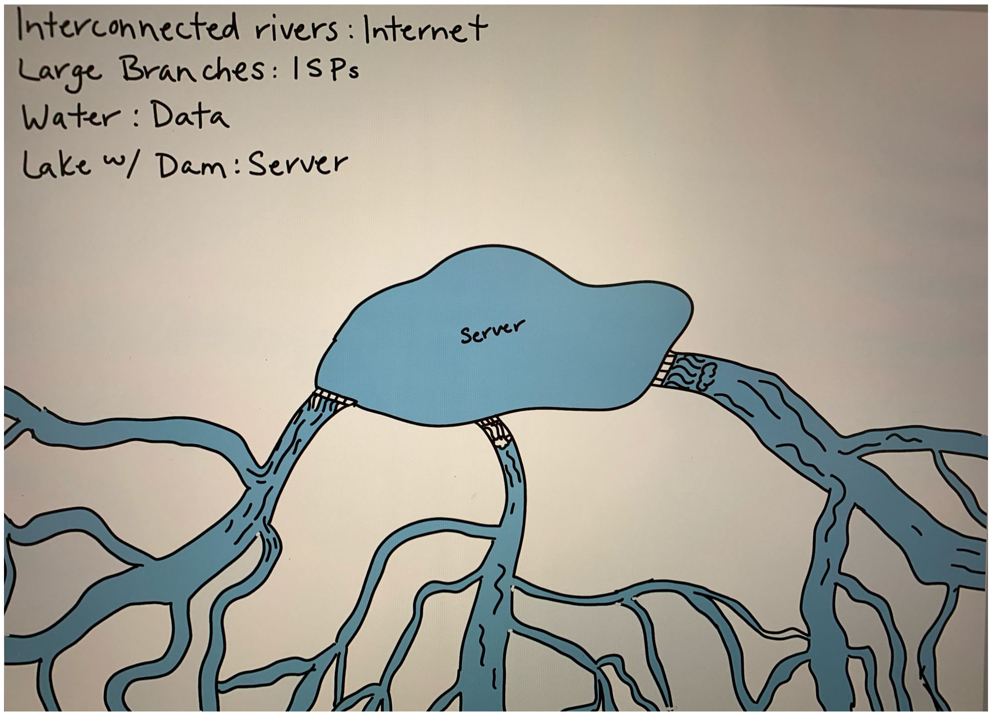
How the Web Works

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. A giant global network that connects networks of computers that are connected to the internet.
2. What is the world wide web? (hint: here)
   a. The world wide web is an interconnected system of webpages that is layered 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?
      i. Networks are computers that are linked and can share information.
   b. What are servers?
      i. Computers that store websites, apps, etc and can serve them up when requested.
   c. What are routers?
      i. A computer that connects computers and ensures the correct information is sent from one computer to the correct recipient.
   d. What are packets?
      i. Data chunks that are sent from a server to a client when they're requested.
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 network of interconnected rivers, and the web or data is like the water that travels through them. Lakes are like servers which store the data/water using dams, and then release it when it is requested or needed.
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?
    a. An IP address is a group of numbers that identifies a particular machine or server.
    b. A domain name is a nickname for a machine or web server that is in words so that it is easier for a human to remember and find.
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, adds privacy, protects your personal information, and prevents data tracking.
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. The browsers use a DNS to change the domain name into an IP address. It checks your local cache first, then the local network, and then your ISP to see if it can find a match before venturing into the internet/root host to find a match.

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	Initial request(link clicked, URL visited)	-I put this step first because it seems like nothing would happen without an initial request for information.
HTML processing finishes	Request reaches app server	-I put this step next because once the request is sent it seems logical that it would then be received by the server.
App code finishes execution	App code finishes execution	-We got this step next after reading an article. It seems like this is the server locating and serving the information.
Initial request (link clicked, URL visited)	Browser receives HTML, begins processing	-After the server sends the requested information it seems logical that it would be received by the browser.
Page rendered in browser	HTML processing finishes	-We put this one next because it would finish after beginning its processing.
Browser receives HTML, begins processing	Page rendered in browser	-We put this last because this seems to be the final product, a rendered page in the browser.

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:
    a. I think it will show us information about the website (IP address, server, etc.) and show us the HTML or at least a sample of it, for the site.
2. Predict what the content-type of the response will be:
    a. I think it 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. I was partially correct, I tried to remember what happened when we did it to google during the interactive lecture, but I didn't realize that the "body" was only the HTML part of it. I re-read back in my notes and clarified the anatomy of the response.
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. I was incorrect, the body was in a text/HTML content type. I thought that the information returned about the website was JSON, but I think I might be a bit confused about what JSON is.

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 it will pull up the entries that have been added
2. Predict what the content-type of the response will be: I think the content type 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.
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. I was wrong! I thought it would be in text/HTML but it was in JSON. I am a little confused

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. It is starts a new entry with the id, date, and content
    b. It pushes the new entry into the entries array
    c. It increments globalID by one

        d. Returns the new array of entries that includes the newest addition
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. We will have to include date and content.
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.
    a. Curl -i -X POST -H 'Content-type: application/json' -d '{"date": "September 27", "content":"The internet is intense."}' http://localhost:4500/entries
4. What URL will you be making this request to?
    a. http://localhost:4500/entries
5. Predict what you'll see as the body of the response:
    a. I think I will see the updated array of entries, including mine.
6. Predict what the content-type of the response will be:
    a. I think it will be in JSON since the last time I looked at the entries area it was in 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. I was! I saw that the last command in the function told it to send entries, and I thought that might mean that it would display the entries with the new additions.
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. I was! It was in JSON just like the last time I looked at the entries page.

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)