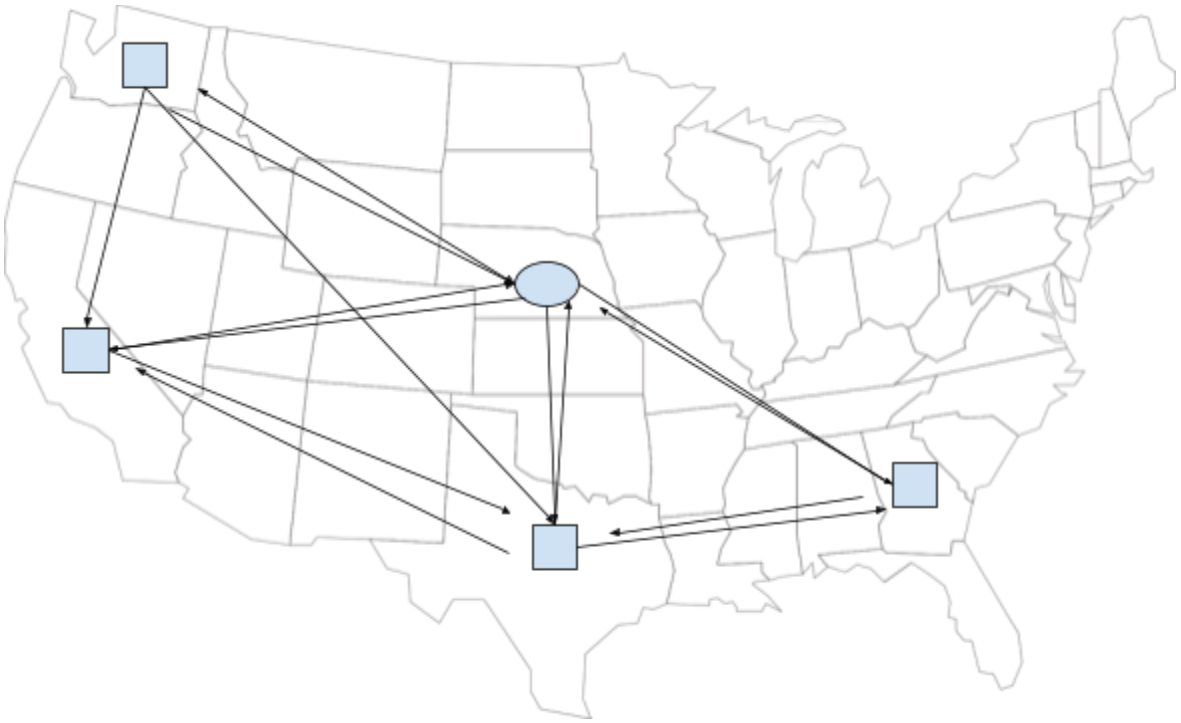
How the Web Works

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 large network of computers which communicate all together.
2) What is the world wide web? (hint: here)
   is an interconnected system of public webpages 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?
      2 or more computers linked/connected either physically or wirelessly to communicate
   b) What are servers?
      computers that store webpages, sites, or apps.
   c) What are routers?
      it makes sure that a message sent from a given computer arrives at the right destination computer
   d) What are packets?
      Data that is sent in thousands of small chunks.
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)
   Like railroads connecting different parts of the country in order to send stuff/mail(data).
   Highways too.

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)



The boxes represent the computers/people that would send the data/mail/goods
The Arrows represent the railroads/highways/cables to connect the destinations
The circle can be a router to connect all the people/ computer and make sure its sent to the right place

Topic 2: IP Addresses and Domains

1) What is the difference between an IP address and a domain name?
   The IP address is a set of numbers that represent a unique location on the web.
   Domain name is a name that is easy to remember, that you can type and a DNS will match it to the Ip address.
2) What's devmountain.com's IP address? (Hint: use 'ping' in the terminal)
   192.168.0.205
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?
   You can get some general info. From them and they can disable certain aspects of the website
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)
   With DNS which are special servers that match up a web address you type into your browser (like "mozilla.org") to the website's real (IP) address.

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	1st step because you have to make the request 1st
HTML processing finishes	Request reaches App server	2nd step because the request will then have to reach the app server
App code finishes execution	Browser receives Html,begins processing	3rd step because then the browser will start to receive the html/data and start processing it
Initial request (link clicked, URL visited)	HTML processing finishes	4th step because the html has to be processed after receiving it
Page rendered in browser	App code finishes execution	5th step because the app will need to finish its execution
Browser receives HTML, begins processing	Page rendered in Browser	Last because the page being rendered is the endgame

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:

- Open a terminal window and run `curl -i http:localhost:4500`
  Jurrni Journaling your journeys

3) Were you correct about the body? If yes, how/why did you make your prediction? If not, what was it and why?
   Yes and no. Yes because i saw what was on server.js but no because i put both the content and header
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 and no. Yes because i saw what was on server.js but no because i put both the content and header

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 will see the entries with their id,date, and content
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
  It will be "Hello world", "Two days in a row!", and "Whoops"

3) Were you correct about the body? If yes, how/why did you make your prediction? If not, what was it and why?
4) Yes because i saw in in the server.js file
5) 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 because i saw in in the server.js file

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)
   Requests that web server accepts the data enclosed in the body of the request message
   Or to upload a file
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,date, and content will be the properties
   Number data and string data
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" : "3", "date" : "09/13/2021", "content" : "I hope this is correct" }'
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:
   {Id:3, date:9/13/2021, content:i hoper this is correct}
6) Predict what the content-type of the response will be:

- 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?
   Sort of. Everything posted except the content part of the entry(It did, there was a typo)
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?
   No, the content did not show up.(Nevermind it worked, just a typo)

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)