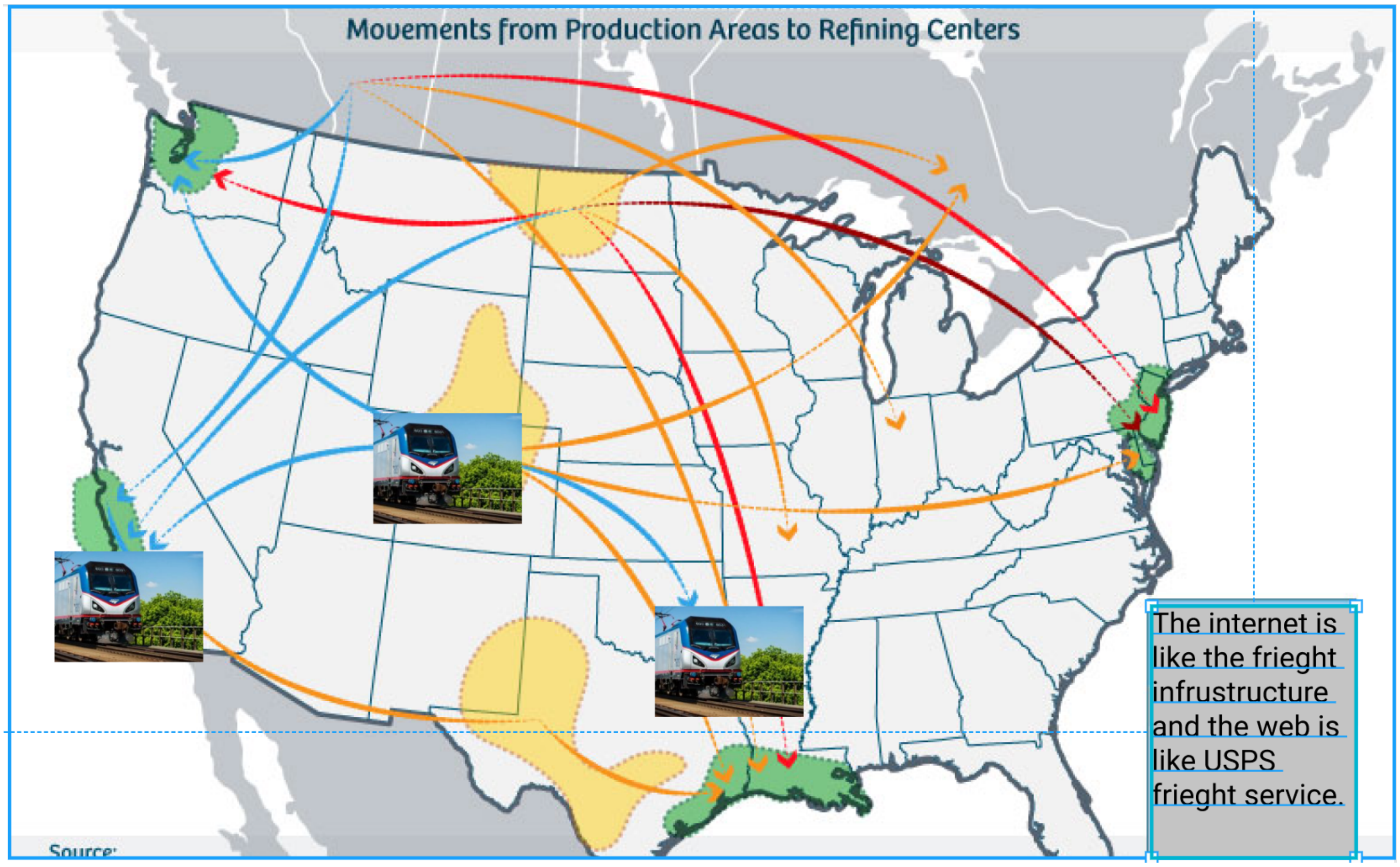
How the Web Works

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) The Internet is a worldwide network of networks that uses the Internet protocol suite (also named TCP/IP from its two most important protocols). Connects networks and servers together.
2) What is the world wide web? (hint: here)
    a) Interconnected system of public web pages. The HTTP protocol governs data transfer between a server and a client.
    b) To access a Web component, a client supplies a unique universal identifier, called a URL (uniform resource locator) or URI (uniform resource identifier) (formally called Universal Document Identifier (UDI)).
    c) HTML (hypertext markup language) is the most common format for publishing web documents.

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) A collection of computers connected together by a router.
    b) What are servers?
        i) Servers are computers that store webpages, sites, or apps.
    c) What are routers?
        i) A router is a tiny computer that makes sure each signal is received by the right computer.
    d) What are packets?
        i) Packets take large pieces of information and break it down into small chunks to send 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)
    a) The internet is like the transportation infrastructure (roads, rails, air) and the web is like the USPS that leverages the infrastructure to send things back and forth to different destinations.
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)


Movements from Production Areas to Refining Centers
The internet is like the frieght infrustructure and the web is like USPS frieght service.
Source:

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 specific location and instructions for the computer (123.00.00.4) a domain name is in human terms and links to the IP address (i.e. 'localhost')
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?
   a) The IP address may change especially if you're using a shared server. The domain name won't change most likely so it's better to have user access your site that way.
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) Because the domain name is linked to the 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 (link clicked, URL visited)	This step starts the process
HTML processing finishes	Request reaches app server	You have to reach the hosted server before any information can be exchanged.
App code finishes execution	App code finishes execution	This is done before the markup text is processed and displays the output to the user
Initial request (link clicked, URL visited)	Browser receives HTML, begins processing	Then the browser takes the received info and starts to interpret it
Page rendered in browser	HTML processing finishes	HTML finished being read and then put together for the display of the website
Browser receives HTML, begins processing	Page rendered in browser	Finally the info is displayed in its intended format

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: The HTML content?
2) Predict what the content-type of the response will be: HTML

- 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? No we were not correct about the body. We weren't sure exactly what the 'body' was.
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?

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: We will see all the information inside the entries object
2) Predict what the content-type of the response will be: Unsure

- 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?
    a) Still confused on the content type and how to predict that. All other predictions were good.
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?
    a) No, we weren't sure what this meant, but we can see now that it's application/json;

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's taking in a request from the user
    b) Creating a new Entry object
    c) Adding to Global ID so now two entries have the same ID
    d) Returning the new entry object
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) Date and content and both will be 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?
   http://localhost:4500
5) Predict what you'll see as the body of the response: we will see the entries array with our new fourth object included
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? 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)