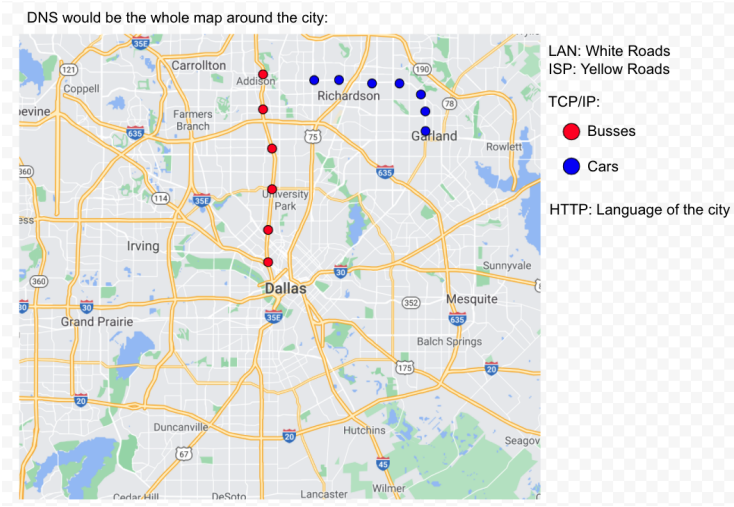
How the Web Works

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

- What is the internet? (hint: here) A large network of smaller networks.
- What is the world wide web? (hint: here) An interconnected system of public webpages accessible through the internet. The web is not the same as the internet: the web is one of many applications built on top of the internet.
- 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
    - What are networks? Connection of Computers/Servers
    - What are servers? Computer on the direct line of the internet.
    - What are routers? Routes the information so that the information goes where it needs to go and come back.
    - What are packets? Thousands of small chunks of data that are sent to compile whatever you are trying to view.
- 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 City
    - The Internet:
        - Local area network is the roads
        - ISP: is the interstate and freeways
    - The Web:
        - The Internet Connection: The roads connecting everything in the city.
        - TCP/IP: Transportation Systems. Cars/Bikes/Busses.
        - DNS: Maps for locations throughout the city.
        - HTTP: The spoken language of the people in the city.
        - Component Files
            - Code Files: Blueprints of how shelves are arranged.
            - Assets: Goods/Products from stores.
- 

Topic 2: IP Addresses and Domains

- What is the difference between an IP address and a domain name? IP address is a set of numerical instructions. The domain name is a link to the IP address.
- What's devmountain.com's IP address? (Hint: use 'ping' in the terminal) 104.22.13.35
- 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? The website might not be the only domain at that IP. Another is that using the IP can access the backdoor and using that can give unwanted access to information.
- 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) It is able to communicate with the Domain Name System to obtain the IP based on the URL/Domain name that we entered.

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.

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 after the initial request because it receives my request for the information I want to receive.
HTML processing finishes	5	It finished processing it for the display
App code finishes execution	3	The app server has processed the request and has sent it back to the browser.
Initial request (link clicked, URL visited)	1	I put this first because it initializes the process.
Page rendered in browser	6	The display has been rendered.
Browser receives HTML, begins processing	4	Browser has received the information and is processing it for display.

If app code is referencing the other languages within the web application, then it would be moved to the final step. As the HTML and CSS are rendered prior to the background code of the website.

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
- Predict what you'll see as the body of the response: Header 1: Jurrni. Header2: Journaling your Journies
- Predict what the content-type of the response will be: Text/HTML string
- Open a terminal window and run `curl -i http:localhost:4500`
- Were you correct about the body? If yes, how/why did you make your prediction? If not, what was it and why? Yes, if you look at line 27 on the server.js file, it shows the response will be: `` `<h1>Jurrni</h1><h2>Journaling your journies</h2>` ``
- 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, line 27 the send string is in text/html syntax

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.
- Predict what you'll see as the body of the response: An array of objects.
- Predict what the content-type of the response will be: .js
- In your terminal, run a curl command to get request this server for /entries
- Were you correct about the body? If yes, how/why did you make your prediction? If not, what was it and why? Yes, because line 30 and 31 show what is being requested. Lines 6-22 are showing what was defined within the 'entries' variable. Which is an array of objects.
- 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 JSON line 4 formats non strings to a Json content-type.

Part C: POST /entry

- Last, read over the function that runs a post request.
- At a base level, what is this function doing? (There are four parts to this)Making a new Object, Pushing it into the array, it is increasing the global id by +1, and will update the /entries page to reflect the new object.
- 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 month and a Day as well as 'content'. They will be a .json string.
- 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. :{"June 21" , "I hope this works"}:
- What URL will you be making this request to? http://localhost:4500/entry
- Predict what you'll see as the body of the response: An array of objects.
- 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
- Were you correct about the body? If yes, how/why did you make your prediction? If not, what was it and why? Yes, because we were pushing an object into the new array.
- 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 Line 4 formats the object to Json
-

Submission

- Save this document as a PDF
- Go to Github and create a new repository. (Click the little + in the upper right hand corner.)
- Name your repository “web-works” (or something like that).
- Click “uploading an existing file” under the “Quick setup heading”.
- Choose your web works PDF document to upload.
- Add “commit message” under the heading “Commit changes”. A good commit message would be something like “Adding web works problems.”
- 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)