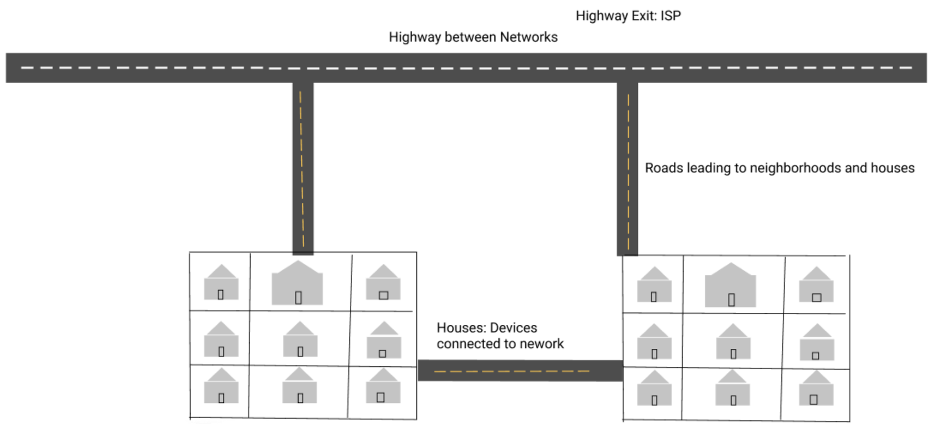
How the Web Works

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) It's a network of connected computers that the web works on.
2) What is the world wide web? (hint: here)
   a) What the user sees and interacts with while on 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) A network is a connection between 2 or more computers, by way of Ethernet, Bluetooth, or Wifi.
   b) What are servers?
      i) Servers are computers that can send information intelligible to web browsers.
      ii) Storage places for webpages, sites, and apps.
   c) What are routers?
      i) A router makes sure that a message sent from a given computer arrives at the right destination computer.
   d) What are packets?
      i) Small chunks of data that are sent from the server to the client/user. Packets can take multiple different routes so that it loads faster for you and for others trying to access that same website at the same time. If it was sent in one big chunk, only one person could download that website at a time.
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) If your device is like a car, then the connections between networks are like highways, and the packets are like smaller roads that get you between houses.
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)


Highway Exit: ISP
Highway between Networks
Roads leading to neighborhoods and houses
Houses: Devices connected to nework

a)

Topic 2: IP Addresses and Domains

1) What is the difference between an IP address and a domain name?
    a) The IP address is the actual address of a webpage, and a domain name is like a nickname that makes it easier to remember.
2) What's devmountain.com's IP address? (Hint: use 'ping' in the terminal)
    a) 104.247.82.130:
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) Because there could be more than one domain using that ip address.
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 DNS converts the IP address to the domain name.

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	I put this step first because the initial request starts the process of loading a web page
HTML processing finishes	Request reaches app server	I put this step after the initial request because you have to make a request in order for it to

		reach the app server.
App code finishes execution	App code finishes execution	I put this step next because the code needs to execute in order to execute the HTML
Initial request (link clicked, URL visited)	Browser receives HTML, begins processing	I put this one after app code finishes execution because the server needs to read the code before it can begin to send what the website will look like back to the user.
Page rendered in browser	HTML processing finishes	I put this step next-to-last because the webpage would be fully loaded once the HTML finishes processing
Browser receives HTML, begins processing	Page rendered in browser	I out this step last because displaying the webpage is the end-goal

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 journies
2) Predict what the content-type of the response will be: 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? Yes, because after the app.get with the '/' it has a string that says what we predicted
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, it was using html formatted text so that's what we put.

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: 0 January 1 Hello World 1 January 2 Two days in a row! 2 June 12 Whoops
2) Predict what the content-type of the response will be: 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?
    a) Almost- we didn't predict that it would also read out the properties along with the string. (include "date:" next to "January 2nd")
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 about the content type. It turned out to be application/json. Honestly, not sure why it was 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) 1:
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)?
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?
5) Predict what you'll see as the body of the response:
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?
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?

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)