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)

Answer: It is a network of computers and servers communicating to each other.

2) What is the world wide web? (hint: here)

Answer: A system of connected web pages that can accessed 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?

   Answer: A network can be as simple as two computers communicating to each other and as complicated as the world wide web network of computers.

   b) What are servers?

   Answer: computer that store webpages, site, or apps.

   c) What are routers?

   Answer: A router is a device meant to make sure that one computer is able to communicate to another computer and receive or send out information.

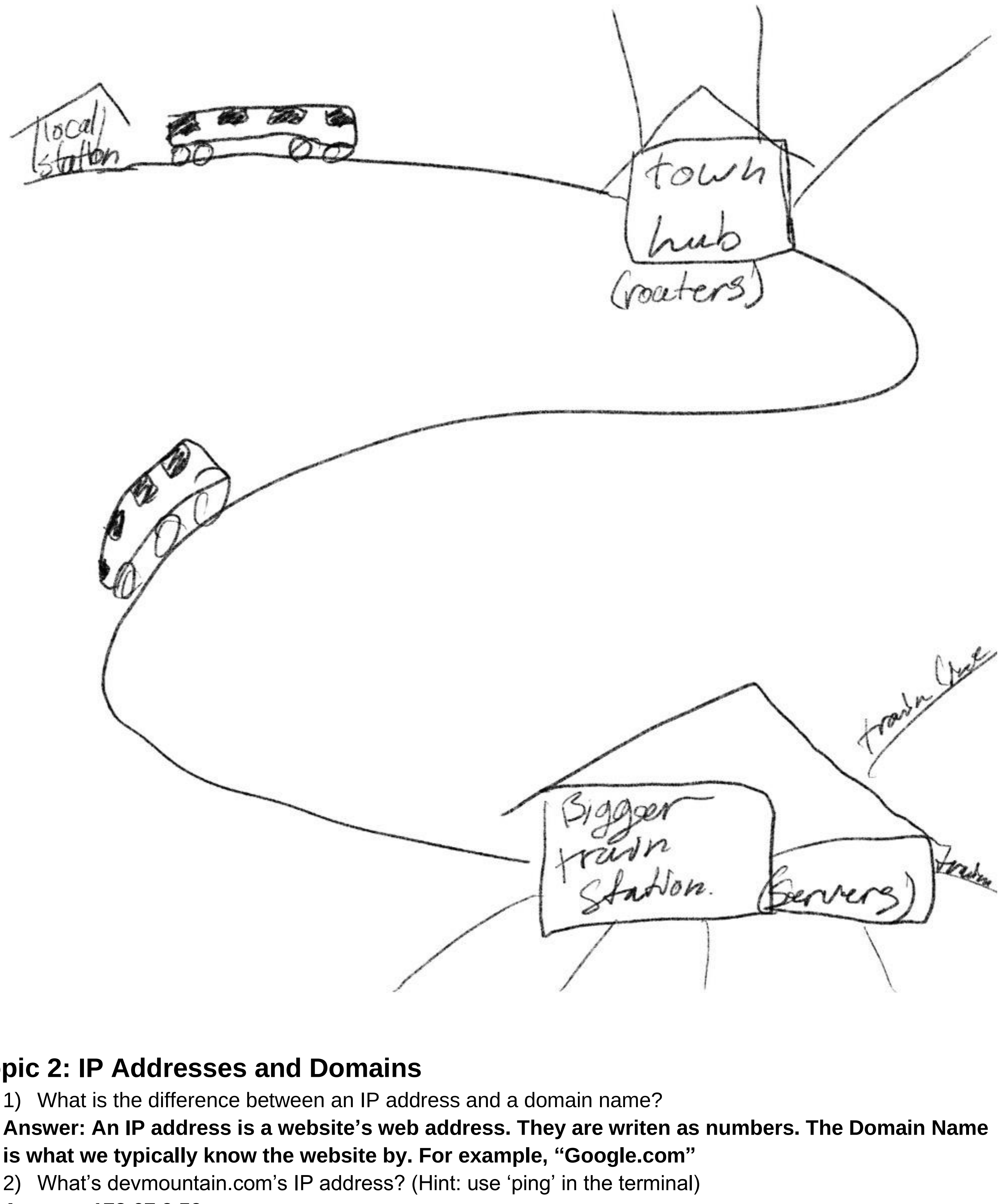
   d) What are packets?

   Answer: Chunks of data for a web page that multiple users can download at the same 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)

Answer: The web is like the network of trains you see in Japan. Almost every town has a station that can be linked to a bigger station that is used a hub for different trainlines to meet and where people can transfer to get to their destination. The people are like the data. The trains are like packets. The smaller local stations are like the various local networks. And the bigger stations(hubs) are like the bigger servers from around the world.

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?

Answer: An IP address is a website's web address. They are writen as numbers. The Domain Name is what we typically know the website by. For example, "Google.com"

2) What's devmountain.com's IP address? (Hint: use 'ping' in the terminal)

Answer: 172.67.9.59

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?

Answer: Because it can contain sensitive information.

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)

Answer: Because of the Domain Name Servers that match those IP addresses to their 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 (link clicked, URL visited)	I put this step first because before any kind of info is to be received, a request must be sent first.
HTML processing finishes	Request reaches app server	This is the next step because the request needs to reach the server for the server to send information.
App code finishes execution	App code finishes execution	I put this here because then the server needs to be able to process that data.
Initial request (link clicked, URL visited)	Browser receives HTML, begins processing	Once the data is sent to the local browser, it begins to download the information in packets.
Page rendered in browser	HTML processing finishes	I put this here because these are the code files that build a website. So once that is finished we go on to the next and final step.
Browser receives HTML, begins processing	Page rendered in browser	I put this last because after all of the information is sent, the local computer should be able to see the web page.

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: **I can't even begin to predict a response.**
2) Predict what the content-type of the response will be:

- 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? **I honestly had no clue what to expect.**
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? **I honestly had no clue what to expect.**

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.

5) Predict what you'll see as the body of the response: **I can't even begin to predict a response.**
1) Predict what the content-type of the response will be: **Entries of some sort.**

- In your terminal, run a curl command to get request this server for /entries

2) Were you correct about the body? If yes, how/why did you make your prediction? If not, what was it and why? **No, it was an array of objects.**
3) 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? **Again, same as the last time, I don't honestly know what to expect.**

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)

This function is adding a new entry to the array with the object and its keys. GlobalID is adding a number to the id of the 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)?
   We will need the Id, the date, and the content. Number, string for both 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.

{"id":3, "date":"May 30","content":"sup!"}

4) What URL will you be making this request to?

5) Predict what you'll see as the body of the response:

I think I will see a two, the date I wrote down, and content:"sup!"

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. **I think I will see a two, the date I wrote down, and content:"sup!"**

- 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?

No. Because I could not figure it out.

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. Because I could not figure it out.

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)