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 worldwide network of networks that uses the protocol suite**
2) What is the world wide web? (hint: here)
   - **An interconnected system of connected web pages 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?
   - **Networks are a set of computers that can intercommunicate**

b) What are servers?
   - **Servers are computers that store webpages, site, or apps**

c) What are routers?
   - **A router is like is like a signaler at a railway station. Routers make sure that a message sent from a given computer arrives at the right destination computer**

d) What are packets?
   - **When data is sent across the web, it is sent in thousands of small chunks Then chunks are called 'packets'**

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)
   - https://www.figma.com/file/JRm626vkEoTKLq0j08IHtk/Untitled?node-id=0%3A1
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?
   - **An IP address is a unique address that finds a computer on the network. IP addresses look like four numbers (0-255) connected by dots.**
     A domain name is a unique, easy-to-remember address used to access websites, such as 'google.com', and 'facebook.com'
2) What's devmountain.com's IP address? (Hint: use 'ping' in the terminal)
   - **104.22.13.35**

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?

- **It might be important to not let users access your site directly at IP address so that your data is not tracked or hacked into.**

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)

- **The domain name system connects the domain name entered by the user to the IP address that is identified using only number. The DNS checks the local cache, the local router, the ISP DNS and then the internet**

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?**
Request reaches app server	Initial request (lin clicked, URL visited	This step is first because an initial request is necessary to load a web page into a browser
HTML processing finishes	Request reaches app server	Next, the request reaches the application for processing
App code finishes execution	App code finishes execution	The app finishes processing before the browser receives the HTML
Initial request (link clicked, URL visited)	Browser receives HTML, begins processing	The browser receives the HTML after the app finishes execution

Page rendered in browser	HTML processing finishes	Browser finishes processing the HTML
Browser receives HTML, begins processing	Page rendered in browser	Page renders to finish the process

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/HTML (line 27)**

- 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, I read it in the code in the HTML string**
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, in the demo that's what happened.**

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:
   - **It will show the entries array that was created in line 6**
2) Predict what the content-type of the response will be:
   - **text/HTML**

- 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?

- **Yes, I was correct about the body, I read the function and guessed it would put out the variable entries**

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 was wrong. I thought it would output text again, but now I know that it will output 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)

- **It is creating a new entry for the entries variable**

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)?

- **It takes in 2 parameters: date, content. It will be two 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.

- **Curl -i -X POST - H 'Content-type: application/json" -d '{"date":"today","content":"Hello there"}'**

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:

- **The 'entries' array as well as the**

6) 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

7) Were you correct about the body? If yes, how/why did you make your prediction? If not, what was it and why?

- **Yes I was correct in my prediction. It showed the contents of the 'entries' array as well as the new one I created**

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, I was correct, I figured when I called the function with the '/entries' and it output application/json then this one should too**

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)