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 very complicated system of networks
2. What is the world wide web? (hint: here) a system of connection between other computers and servers
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? **A network is when two computers are connected together to communicate.**
    b. What are servers? **Servers are computers that store webpages, sites, or apps**
    c. What are routers? **A router is a type of computer that directs information from one computer to another computer. Interlinking the computers in a network.**
    d. What are packets? **They are small packets of data that are sent across the web in packets because they are less likely and easier to fix when they become corrupted.**
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) **An example for the internet could be the flight control at airports. The flight control is a means to communication between all of the different airplanes. Connecting all of the airplanes together through communication which they can then use to send information amongst each other. Information like landing times, flight problems, and number of passengers and be communicated to help complete a communal task. Whereas the flight board within the airport would be a service that is given through the "backbone" of the airport. Passengers can look at the flight board and know when their planes are taking off and what gates they need to go to in order to catch their flights.**
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? **The IP address is your unique location on the web whereas your domain name is the name of the said site that you are trying to visit**
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? **Because the name should be something that is easy to remember for ease of use.**
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 domain name server is a computer that is responsible for attaching all of the IP addresses to their corresponding domain names.**

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 goes first because a action is required to start code
HTML processing finishes	Request reaches app server	The action goes somewhere next
App code finishes execution	Browser recieves HTML, begins processing	The action is received.
Initial request (link clicked, URL visited)	App code finishes execution	The action is coded out and finished
Page rendered in browser	HTML processing finishes	The HTML finishes next
Browser receives HTML, begins processing	Page rendered in browser	The final result would be the rendering of the 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: JournalServer
2. Predict what the content-type of the response will be:

- Open a terminal window and run `curl -i http:localhost:4500`

1. Were you correct about the body? If yes, how/why did you make your prediction? If not, what was it and why? No, I missed the journaling journeys part.
2. 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 not correct. I saw the name and assumed that that would be printed on the page but I didn't think that "journaling journeys" would be on the page as well.

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: January 1, hello world ,january 2, two days in a row, June 12, whoops
2. Predict what the content-type of the response will be: An array

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

1. 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 went to the different info that was in the array and posted it

2. 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 saw that It was sending the “entries array”

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) its going to push a (Newentry) to the array above
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

1. Were you correct about the body? If yes, how/why did you make your prediction? If not, what was it and why?
2. 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)