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 Internet protocol suite

2) What is the world wide web? (hint: here)
   An interconnected system of public webpages 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?
      i) Two or more computers that are linked in order to share resources.
   b) What are servers?
      i) Computers that store webpages, sites, or apps.
   c) What are routers?
      i) A device that communicates between your internet and devices that connect to it.
   d) What are packets?
      i) Tiny chunks of data sent by websites.
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) The internet is like a warehouse: a place where things are stored that's ever changing with content coming and going.
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?
   a) An IP address is the numerical identity of a site, and a domain name is a string that points to that number.
2) What's devmountain.com's IP address? (Hint: use 'ping' in the terminal)
   a) 104.22.12.35
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?
   a) If you need to change your IP, you can just update your DNS to point to the new one. And it's a security liability, but that's secondary.
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) Your computer asks a nearby Domain Name System if it knows where the site you're trying to locate is, to make the process of getting there faster.

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	2	After the client sends the request, the server responds and sends the HTML files, and references to CSS and script files.
HTML processing finishes	5	HTML finishes processing and loads the CSS and script files.
App code finishes execution	6	The server finishes sending information.
Initial request (link clicked, URL visited)	1	I put this first because it is initial. Nothing else can happen before the request is actually sent.
Page rendered in browser	4	Having HTML, the page starts rendering.
Browser receives HTML, begins processing	3	The client receives HTML and starts to load in.

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:
    a) A header tag that says Jurrni with its subheader.
2) Predict what the content-type of the response will be:
    a) JSON

- 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?
    a) Yes. Because -i returns headers and the only two things on the webpage are two headers.

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?
    a) Kind of? JSON is written in strings, so technically it was text. But the more accurate answer is HTML, because that's the foundational language of most websites.

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:
    a) Header tags with the previous information as well as the entries array.
2) Predict what the content-type of the response will be:
    a) 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?
    a) Kind of. Did not have the previous headers, but did have the array. Which makes sense, because the headers are only sent when '/' is the requested path.

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?
    a) No, it was app/JSON which kind of makes sense because it's literally just sending objects, but I don't know how that makes it an application type.

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) Create new object based on request contents
    b) Push new object to existing array
    c) Increment global ID
    d) Return status to client
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)?
    a) Date
    b) Content
    c) Both are 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.
    a) {"date":"August 2","content":"Hello."}
4) What URL will you be making this request to?
    a) localhost:4500/entry
5) Predict what you'll see as the body of the response:
    a) The entries.
6) Predict what the content-type of the response will be:
    a) 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?
    a) Yes.
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?
    a) Yes, it did. Copy paste didn't work at all, and I spent over an hour this dumb curl command.

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)