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 (also named TCP/IP from its two most important protocols).
2) What is the world wide web? (hint: here) The *World Wide Web*—commonly referred to as **WWW**, **W3**, or **the Web**—is an interconnected system of public webpages accessible through the Internet. The Web is not the same as the Internet: the Web is one of many applications built on top of 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? Collection of connected computers/devices
    b) What are servers? Computer that stores files and sends the response from a client computer
    c) What are routers? Minicomputer that connects other computers
    d) What are packets? Small bits of information parsed and sent by server
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) The web is sort of like a library. They use the Dewy Decimal System, like an IP address, to connect familiar names with information addresses. The internet is like the network of libraries where you can request a book in Utah from Nevada as long as it's in the network, and the protocol, or paperwork is sent and a book, or web page, is returned.
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) https://www.figma.com/file/J44z9hRrVQYeSGITBpSsWa/Untitled?node-id=0%3A1

Topic 2: IP Addresses and Domains

1) What is the difference between an IP address and a domain name? The IP address is the actual number associated with a location on the internet. The domain name is the registered placeholder name for an IP Address using words that make it easier for humans to read and remember.
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? Makes it easier to hack
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) It checks first for history, then for stored searches on router or network, then for associations with ISP, and finally it checks domain providers like GoDaddy to find the IP address of a 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	2	Server must receive request for anything else to happen
HTML processing finishes	4	I believe all HTML is processed before anything else is
App code finishes execution	6	Javascript applied last; after HTML and CSS render page.
Initial request (link clicked, URL visited)	1	I put this first because the request is what starts the process
Page rendered in browser	5	Page is rendered after DOM tree but before Javascript is parsed and applied
Browser receives HTML, begins processing	3	Web documentation said HTML is parsed first, then any CSS or Javascript or other links.

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: h1, h2

- 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 made my prediction based on the test that was supposed to appear with the basic index request (port 80)
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? Yesish. Says text/html specifically.

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
2) Predict what the content-type of the response will be: Javascript
- 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? No. It displayed all of the entries instead of just one.
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? No. It says 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) Creating a new entry object, adding that object to the entries array, incrementing the global ID variable, sending the new entry back to the page.
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)? Date and content. JSON.
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. ‘{“date”: “July 19”, “content”: “learning JSON”}’
4) What URL will you be making this request to? Localhost:4500/entry
5) Predict what you’ll see as the body of the response: All of the entries object again
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
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

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)