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) Its a world wide network of networks that uses the Internet protocol suite
2) What is the world wide web? (hint: here) Interconnected system of public webpages accessible through 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? When 2 computers communicate with each other
   b) What are servers? Computers that store webpages, sites or apps.
   c) What are routers? Small computer that makes sure that a message arrives to the right destination computer
   d) What are packets? Small chunks of data that's sent from the server to the client.
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 perception with rain returning to the "cloud".
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)
   a) Check the GitHub repository for the Diagram.

Topic 2: IP Addresses and Domains

1) What is the difference between an IP address and a domain name? IP address is the specific location of the website in numbers that a computer understands and the domain is written so that way we can remember it and is broken down into the IP's numbers for the computer to go to the website.
2) What's devmountain.com's IP address? (Hint: use 'ping' in the terminal) 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? For Bot nets, and bot attacks. Especially if you have encrypted data.
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) First it checks the browser for the DNS of the Domain name, if the browsers cache doesn't have it, itll check the router, if the router doesn't know it. Itll talk to the Resolver (ISP) for the DNS and its root.

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 first because nothing can happen until the request is made.
HTML processing finishes	Request reaches app server	I put this second, because once the request is made itll be sent all the way to the apps server.
App code finishes execution	Browser receives HTML, begins processing	Then the app server will send the HTML in packets to my browser and start processing
Initial request (link clicked, URL visited)	HTML processing finishes	Right after processing it will finish
Page rendered in browser	Page rendered in browser	Then all the processed HTML/ CSS/JS will be rendered in the browser
Browser receives HTML, begins processing	App code finishes execution	After its rendered the JS files will run and the app code will finish.

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: 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: The HTML.
2) Predict what the content-type of the response will be: Text/HTML

- 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? The only send request is the HTML
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) Yes because the only send request was for the HTML text File.

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: The object of entries with the ID, date and Content
2) Predict what the content-type of the response will be: application/JSON

- 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 because any object is going to return JSON data which in turn comes as the application/ JSON content-type
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, and I see that the entries variable in send is an Object that's being retrieved with get.

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) Listening for /entry parameter
    b) Taking in the HTTP request data, and res is to send back to the desired HTTP response
    c) Creating a newEntry Object and pushing the new object into the entries array.
    d) And sending the status(200) with the entires array.
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) Id, date, content. Integer and 2 String's
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.
    - “Entries”: {
      “id”: “0”,
      “date”: “January 1”,
      “content”: “Hello world”
      }
4) What URL will you be making this request to?
    a) http://localhost:4500/entries
5) Predict what you'll see as the body of the response:
    a) The entire Json Object that I created in question 3
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 “Entries”: {
      “id”: “0”,
      “date”: “January 1”,
      “content”: “Hello world”
      } http://localhost:4500/entries

7) Were you correct about the body? If yes, how/why did you make your prediction? If not, what was it and why? Yes, because we gave it a JSON object so it'd return the data in the JSON object
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 because we gave it a JSON object so it'd return a application/json content type.

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)