How the Web Works

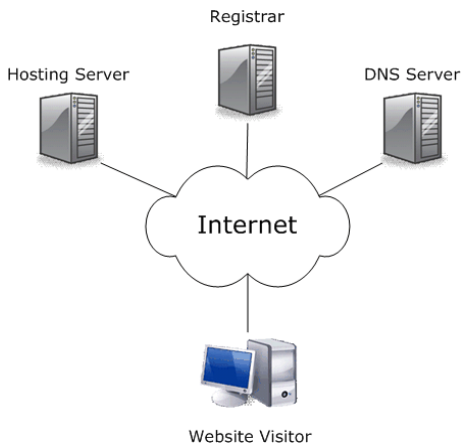
In this lab, you'll be working with a partner to explore a little more about the internet, the web, requests, responses and more. You'll be reading and writing about concepts as well as practicing some of the commands that we saw during the lecture earlier.

Topic 1: The Internet and the World Wide Web

1) What is the internet? The Internet is a vast network that connects computers all over the world. Through the Internet, people can share information and communicate from anywhere with an Internet connection (hint: here)
2) What is the world wide web? Interconnected system of public web pages 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. (hint: here)
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? consists of two or more computers that are linked in order to share resources (such as printers and CDs), exchange files, or allow electronic communications. The computers on a network may be linked through cables, telephone lines, radio waves, satellites, or infrared light beams. LAN - local area network, WAN-Wide Area Network.
    b) What are servers? Computers that store web pages, sites, or apps.
    c) What are routers? LAN - local area network, WAN-Wide Area Network, ports used to connect computers. The port that connects the router to the outside world is usually labeled WAN
    d) What are packets? A piece of information being sent, break up the file into piece (packets) and sends it to its destination
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 internet is like the US postal service just electronic and faster
    The web is like a Flea Market for everyone

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?
   IP address: Is the physical website ex: 127.0.0.1 Domain name: nickname for IP address, can have multiple domains go to the same address ex: www.google.com

2) What's devmountain.com's IP address? (Hint: use 'ping' in the terminal)
   IP Address: 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? Makes it easier for someone to attack the website if you don't protect it, not as convenient to remember and use an IP address.

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 brings the two together and gets you to your destination.

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	I put this step first because 'initial means first and the initial request will start the process.
HTML processing finishes	Request reaches app server	I put this step after the initial request because after you send the request it will soon reach the app server.
App code finishes execution	App code finishes execution	I put this step after the request reaches the app server because you must wait for the code to execute before moving on
Initial request (link clicked, URL visited)	Browser receives HTML, begins processing	I put this step next because the HTML needs to be processed before it's rendered correctly.
Page rendered in browser	HTML processing finishes	I put this step after processing, because it has now finished processing and will soon render

		the browser
Browser receives HTML, begins processing	Page rendered in browser	I put this step last because this is when the web page has loaded into the browser.

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: I will see two sentences in the body in the html format.
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? Yes, I used notes and MDN research to make a decision.
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, I used my notes and MDN to make a decision.

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 function will send back entries
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, thinking about it aloud and reading the code as it was to find the solution.
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, the use of notes and reading the code.

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)

- Post entry end point
- Creates a new entry for entries array from the request body.

- Pushes new entry into entries array
- Iterate globalId once each time it is run.
- Then sends entries back.

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)?
   Properties needed: date, content. The data type for both will be: application/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.

```
{
        "id": "0",
        "date": "January 1",
        "content": "Hello world"
    },
```

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: received entries
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, Using notes and reading the code to determine the answer
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, Using notes and reading the code to determine the answer.

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)