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?

Clients are the typical web user's internet-connected devices (for example, your computer connected to your Wi-Fi, or your phone connected to your mobile network) and web-accessing software available on those devices (usually a web browser like Firefox or Chrome).

A network 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.

    b. What are servers?

Servers are computers that store webpages, sites, or apps. When a client device wants to access a webpage, a copy of the webpage is downloaded from the server onto the client machine to be displayed in the user's web browser.

c. What are routers?

A router is a switching device for networks, which is able to route network packets, based on their addresses, to other networks or devices. Among other things, they are used for Internet access, for coupling networks or for connecting branch offices to a central office via VPN (Virtual Private Network).

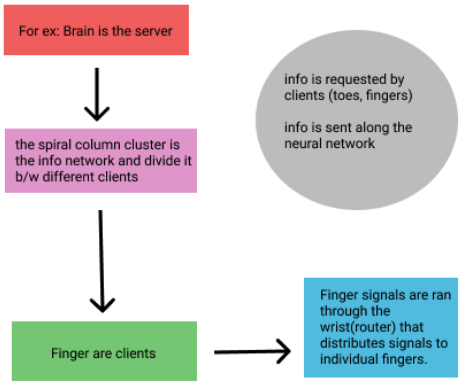
d. What are packets?

"packets" to describe the format in which the data is sent from server to 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)

Metaphor of the Internet(web) : human brain(server) (information is requested by the client[finger, toes'] from server to brain⇒ finger (clients)[the finger signal ran through the wrist (router) that distributes signals to individual fingers

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)



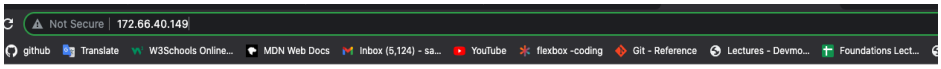
Topic 2: IP Addresses and Domains

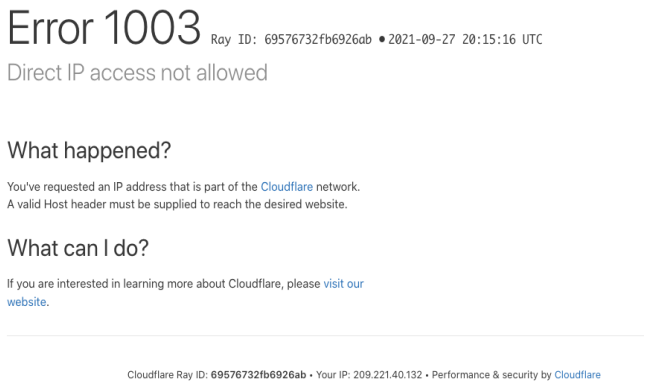
1. What is the difference between an IP address and a domain name?

The IP address is an actual set of numerical instructions. ... The domain name functions as a link to the IP address. Links do not contain actual information, but they do point to the place where the IP address information resides.

2. What's devmountain.com's IP address? (Hint: use 'ping' in the terminal)

devmountain.com IP address 172.66.40.149





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?

   To prevent DNS attacks against the 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 process starts with **browser checks its cache (memory)** to see if it knows which IP address the domain name resolves to. If it knows, it will resolve it and display the web page. If your computer doesn't already know the answer, it needs to perform a DNS query to find out

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.

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 this is where the client initializes the request for informations.
HTML processing finishes	Request reach APP server	I put this step next because the request has been sent but nothing has been processed yet
App code finishes execution	Browser receives HTML begins processing	I put this step next because the APP has received the request and has sent back the

		HTML coding to process
Initial request (link clicked, URL visited)	App code finishes execution	I put this step next because the app finishes sending the code back to the client before the HTML is finished
Page rendered in browser	HTML processing finished	I put this step next because the Html is fully received after the app has sent all the info and finishes execution
Browser receives HTML, begins processing	Page renderer in browser	I put this step next because the page rendering is the final step in the request

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 curl -i http://localhost:4500

1. Predict what you'll see as the body of the response:
2. Predict what the content-type of the response will be: HTML text

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

<h1>Jurrni</h1><h2>Journaling your journies</h2>

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 , bc the HTML text was in the web browser

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 dates, id : 0,1,2, the contents
2. Predict what the content-type of the response will be: 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

[{"id":0,"date":"January 1","content":"Hello world"},{"id":1,"date":"January 2","content":"Two days in a row!"},{"id":2,"date":"June 12","content":"Whoops"}]⏎

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

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)
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. json' -d '{"date": "September 27", "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:
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
    - curl -i -X POST -H 'Content-type: application/json' -d '{"date": "September 27", "content": "Hello, World"}' http://localhost:4500/entry (don't need to id)

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)