Topic 1: The Internet and the World Wide Web

1) What is the internet? (hint: here)

The internet is a platform of data exchange between different hosts and servers around the world.

2) What is the world wide web? (hint: here)

One of the applications provided within the internet that connects public webpages. It connects resources through hyperlinks and consists of several components:

HTTP

URL

HTML

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?

To be able to allow computers to communicate or share information they can be connected both wirelessly or physically and the overview of this communication is referred to as the computer network.

b) What are servers?

Computers that store information that can then be requested and sent to a client.

c) What are routers?

A router is responsible for directing signals to the correct location to allow the sharing of information to occur.

d) What are packets?

The format information is sent from a server to a client. Information is sent in many small packets so it can be sent over many routes to make the transfer of information faster.

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)

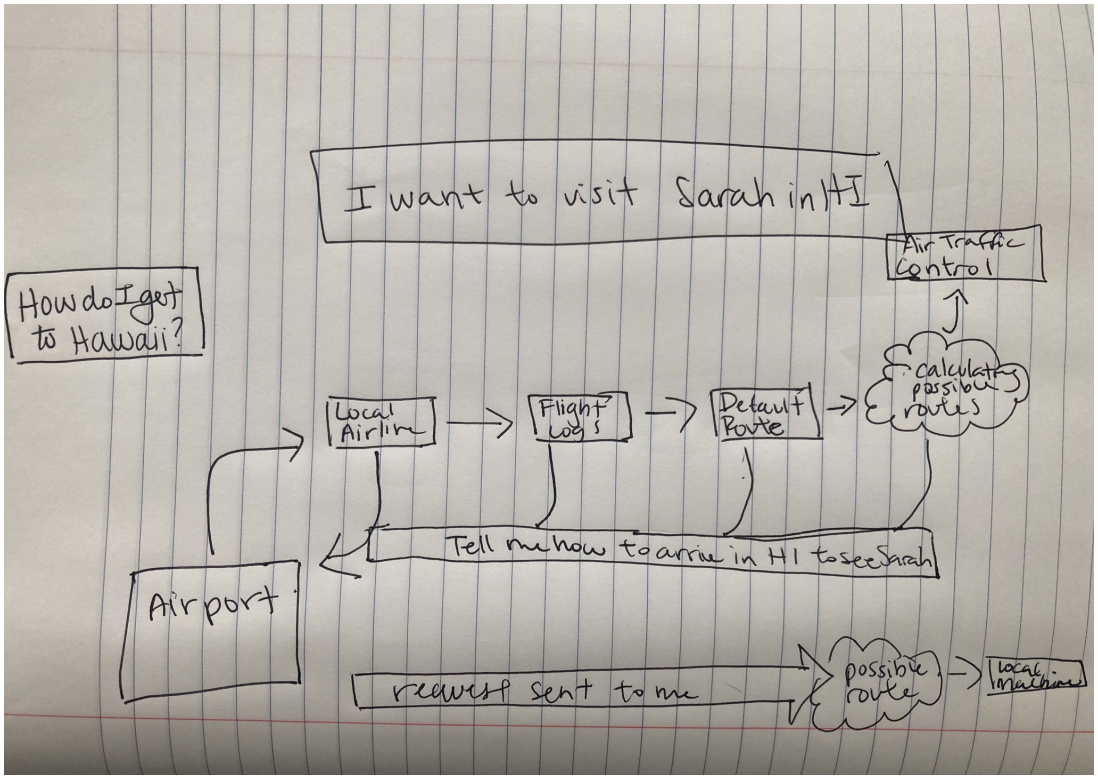
Airport: Internet

The airport houses and then routes different airlines worldwide

Airlines: World Wide Web

Airlines use specific resources and airplanes(applications) to transport passengers(data)

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 addresses are the actual numerical information assigned to every device and website. Domain names are comprised of IP addresses but are composed of words in place of numbers so they are easier to understand and remember for humans.

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

172.67.9.59

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?

Hiding the IP address is used to provide a more secure website and protect user information.

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)

If it can not be found locally on your cache a signal will be sent to...

the router-> ISP DNS -> interwebs-> Root DNS (saved to cache for future) ->back to server/display the browser

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)	Initial request (link clicked, URL visited)	We put this step first because a signal or request first needs to be sent out to a server to load a webpage on a client browser
HTML processing finishes(5)	Request reaches app server	We put this step second because the request has been received by the server
App code finishes execution(3)	App code finishes execution	This step comes next because the server is packing the request to send back to the client browser
Initial request (link clicked, URL visited)(1)	Browser receives HTML, begins processing	Next comes this step because the client browser receives the information from the server
Page rendered in browser(6)	HTML processing finishes	The information is downloaded and processed to become viewable to the client browser
Browser receives HTML, begins processing (4)	Page rendered in browser	This is last because all of the information steps have been processed and completed and the webpage is now accessible to the client 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:

We will see an id, a date, and content

2) Predict what the content-type of the response will be:

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?

No, we made our prediction based on the entries portion. The visible text in the web browser was after the get functions and labeled with a header and text format.

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, we were correct that it was a text format. We based this off of what we could see in the code file and saw they were written in strings which is something we are more familiar with.

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:

We will see an id, a date, and content

2) Predict what the content-type of the response will be:

It will be text

- 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, we were correct about our prediction being the values of the entries.

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, we were correct about the content type of the responses because in entires they were listed as strings which is something we are familiar with seeing.

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)

At a base leve, this function is pushing up new information to the browser that is and id, a date, and some content.

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)?

Id: number

Date: string

Content: string

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": "3", "date": "July 5", "content": "fireworks!"}'

4) What URL will you be making this request to?

http://localhost:4500/entries

5) Predict what you'll see as the body of the response:

A flash, a pop, a bang! We will see [ ] and our exact JSON request

6) Predict what the content-type of the response will be:

Text contained within an array

- 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 '{"id": "3", "date": "July 5", "content": "fireworks!"}' http://localhost:4500/entry

7) Were you correct about the body? If yes, how/why did you make your prediction? If not, what was it and why?

Yes, we were correct. We made our prediction by looking at the entries portion.

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, we were correct. They mirrored what was returned on the entries run.

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.