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 networks that passes around data.

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

Application of public web pages that uses 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? Ways to connect different devices together, it started with cables but it rapidly outgrew cables, and needed an advance solution. There are different levels of connectivity.
   b) What are servers? A server is the host to sending out information to the clients.
   c) What are routers? A router is a wireless way to connect to the internet so we aren't dealing with so many wires.
   d) What are packets? Packets are the way we break down the data we send into small replaceable places. Reason they started doing package it was faster and more people can access it at the same time.
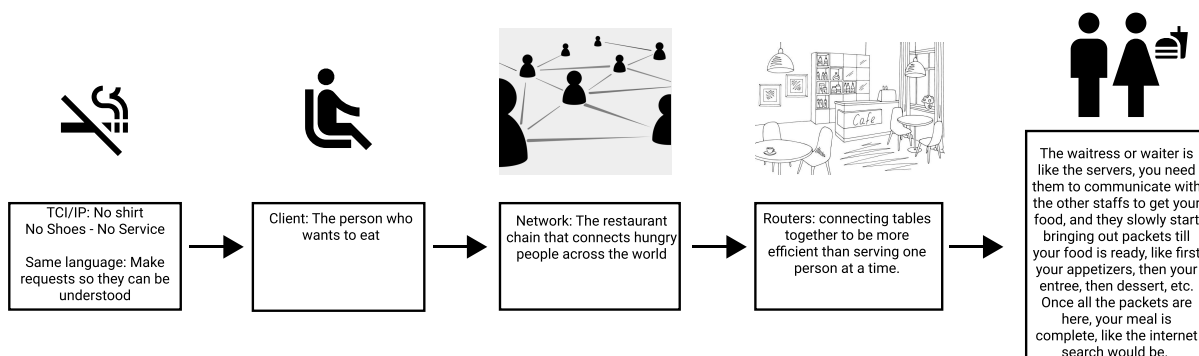4) Come up with a metaphor for the internet and the web, you can do a single one if you think of one that puts them together or two separate ones (feel free to use one you've heard today or read about if you can't think of a new one, but spend at least 10 minutes trying to think of something different before you resort to that)

A restaurant setting is like the web and internet. There are rules you have to follow first to get in the restaurant (ex no shirt no service, so if youre not following some of the rules you can't even enter inside), like if you dont have the correct set up (a router or a laptop, you can't even get on the internet).

The restaurant is like the network, it connects all the people who want to eat

The waitress or waiter is like the servers, you need them to communicate with the other staffs to get your food, and they slowly start bringing out packets till your food is ready, like first your appetizers, then your entree, then dessert, etc. Once all the packets are here, your meal is complete, like the internet search would be.

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)

TCI/IP: No shirt No Shoes - No Service

Same language: Make requests so they can be understood

→ Client: The person who wants to eat

→ Network: The restaurant chain that connects hungry people across the world

→ Routers: connecting tables together to be more efficient than serving one person at a time.

→ The waitress or waiter is like the servers, you need them to communicate with the other staffs to get your food, and they slowly start bringing out packets till your food is ready, like first your appetizers, then your entree, then dessert, etc. Once all the packets are here, your meal is complete, like the internet search would be.

Topic 2: IP Addresses and Domains

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

A domain name is whatever string you want it to be and it points to an IP address.

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?

Because security reasons

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)

There are DNS servers that will search the domain name and return the IP address.

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 picked you need to make a request to start
HTML processing finishes	Request reaches app server	I put this step next bc you need because the request needs to be received
App code finishes execution	App code finishes execution	This shows request was processed and understood before response
Initial request (link clicked, URL visited)	Browser receives HTML, begins processing	This step shows received and loading
Page rendered in browser	HTML processing finishes	Request is fulfilled
Browser receives HTML, begins processing	Page rendered in browser	Request is fully displayed

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

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

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.
4) What URL will you be making this request to?
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
7) Were you correct about the body? If yes, how/why did you make your prediction? If not, what was it and why?
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?

—talked with Megan through the get and post section

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)