# 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)
   A worldwide network of networks that uses tcp/ip to communicate
2) What is the world wide web? (hint: here)
   Public webpages, accessed through 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? A group or set of interconnected computers/machines
   b) What are servers? Computers that store web-pages/sites/apps
   c) What are routers? A specialized computer that connects multiple devices to each other and also to the internet.
   d) What are packets? Small pieces of data from a websites files in the server that are sent to the client, accessing the website
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 a road system, that serves many different types of traffic at different levels (highways vs farm to market roads) where www is more akin to business fronts where you can access specific items.

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)
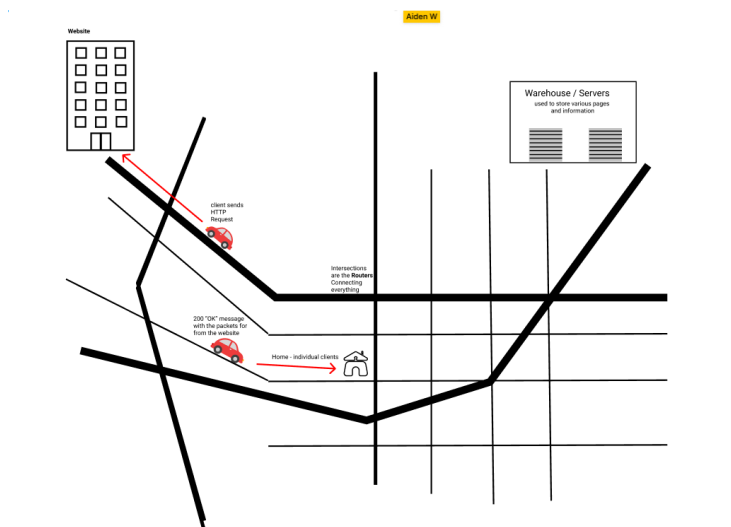
Topic 2: IP Addresses and Domains

1) What is the difference between an IP address and a domain name?
   Domain is the 'nickname' of the IP address.. Ip address is a set of numerical identifiers.
2) What's devmountain.com's IP address? (Hint: use 'ping' in the terminal)
   2606:4700:10::6816:c23
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?
   Protect against DDoS
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 IP address is being translated into a much more readable domain name/DNS

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)	Starts process, user action
HTML processing finishes	Request reaches app server	Recognizes requests, starts to respond
App code finishes execution	App code finishes execution	The answer to the request - what needs to happen to fulfill the client request
Initial request (link clicked, URL visited)	Browser receives HTML, begins processing	Start of loading on client side of html and any embedded objects/functions
Page rendered in browser	HTML processing finishes	The code is executed - under the hood processes - user doesn't' see this, they see the rendered version of this
Browser receives HTML, begins processing	Page rendered in browser	Fully rendered page is the end product - what the client sees

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? Yes - gave back basic curl information, once I got rid of all the user errors(fat fingers) - the '/' indicated that this was going to be the general response from a home page
4) Were you correct about the content-type of the response? If yes, how/why did you make your prediction? Seems like reasonable feedback although I'm not quite sure what it all means except for the <h1> message </h1>

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: a list of the entries from the /entries function
2) Predict what the content-type of the response will be: template style of entries?
- 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 - it's listed in the javascript code
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? Not sure what is meant by content-type. Entries seemed more like a template/array of classes.

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)
   Defining a new Entry
   adding next in line to entries array
   Adding a counter to the global ID - moving the placeholder up by 1
   Returning entries to the client?

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)?
   Would need to include date and content, both as strings

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) {"date":"June 1", "content":"I don't know"}

5) What URL will you be making this request to? localhost:4500/entry
6) Predict what you'll see as the body of the response: [{"id":"4","date":"June 1", "content":"I don't know"}]
7) Predict what the content-type of the response will be: array with strings and pairs
- 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
8) Were you correct about the body? If yes, how/why did you make your prediction? If not, what was it and why? Mostly yes, I forgot that it sends all entries, so you get back the entire array of objects (all dates and comments)
9) 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? I still don't understand content-type response ~ yes and no

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)