# 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): ***Worldwide network of networks with it own internet protocol suite***
2) What is the world wide web? (hint: here): ***Interconnected system of public webpages, an application built over 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 system of interconnected computers, either physically or wirelessly***
    b) What are servers?: ***Computers that store webpages, sites, or apps and responds to clients***
    c) What are routers?: ***A router is special tiny computer that 'routes' messages to the appropriate computer on the network***
    d) What are packets?: ***The format in which data is sent from client to server, breaking it down into smaller pieces allows data to travel along various paths to the server asynchronously and facilitates multiple users to receive responses 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) ***The internet is like your skeleton, it provides the foundation as well as the limitations of the things built upon it. The web is like your muscles the muscles in conjunction with the skeleton are able to produce a result that is desirable to you at that time.***
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): https://www.figma.com/file/RMITDfHk9Vq57Urn94O7BC/HOW-THE-INTERNET-WORKS?node-id=0%3A1

## Topic 2: IP Addresses and Domains

1) What is the difference between an IP address and a domain name?: ***IP address is a numeric address that you can type in to connect to another computer (ie. Server) where as a domain name is the word url that we are most accustomed to typing in (ie. Google.com)***
2) What's devmountain.com's IP address? (Hint: use 'ping' in the terminal): ***104.22.13.35***
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?: ***There are security risks that could potentially be an issue, another issue is the need for a static IP address, and if your DNS host needs to reassign your IP then there would be no way to redirect or forward a client to your actual site.***
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): ***It will first check its local cache, then the router cache, then the ISP cache, then if it absolutely cannot find it there it will go beyond to other ISP's and then to the root directory***

## 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 ____*<br>*- I put this step before/after ____ because ____* |
| Request reaches app server | Initial request | You need to make a request to trigger the action first |
| HTML processing finishes | Request reaches app server | The request has to go somewhere |
| App code finishes execution | App code finishes execution | The code has to finish executing first so that all of the HTML requests can be sent if needs be |
| Initial request (link clicked, URL visited) | Browser recieves HTML, begins processing | The server responds to the client and begins sending back the processed HTML |
| Page rendered in browser | HTML processing finishes | After HTML requests for outside resources are sent they will all finish asynchronously |
| Browser receives HTML, begins processing | Page rendered in browser | After everything is received by the client the browser will construct the webpage based on the HTML received |

## 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: ***Jurrni Journaling your Journies***
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 because the first element in the code that had a 200 acceptance flag as well as HTML content was what  written.***
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 because the first element was returning a string***

*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 body will be the entries variable, and be the array with the 3 objects within it***
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 it returned the variable entries like I had guessed***
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? ***Looking at the variable I saw that there was an array full of objects, and the only format I knew that would transmit an object was JSON***

*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): ***1. Defines a variable newEntry 2. It fills the variable newEntry with an object containing the same keys as the entries variable as well as the data you gave it 3. Pushes the newEntry variable into the entries array 4. Increments the globalId variable so that the next object that gets created has an ascending id number and they're not all the same***
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)?: ***You need to give it the date and content data for associated keys and they will be 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. ***'{"date": "today", "content":"HEY BUDDY"}'***
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: ***entries variable + new JSON***
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 we were correct about our prediction. We used the same logic to predict the previous part***
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 because we got to set the data type in our command, so hopefully we know what to expect back haha***

## 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)