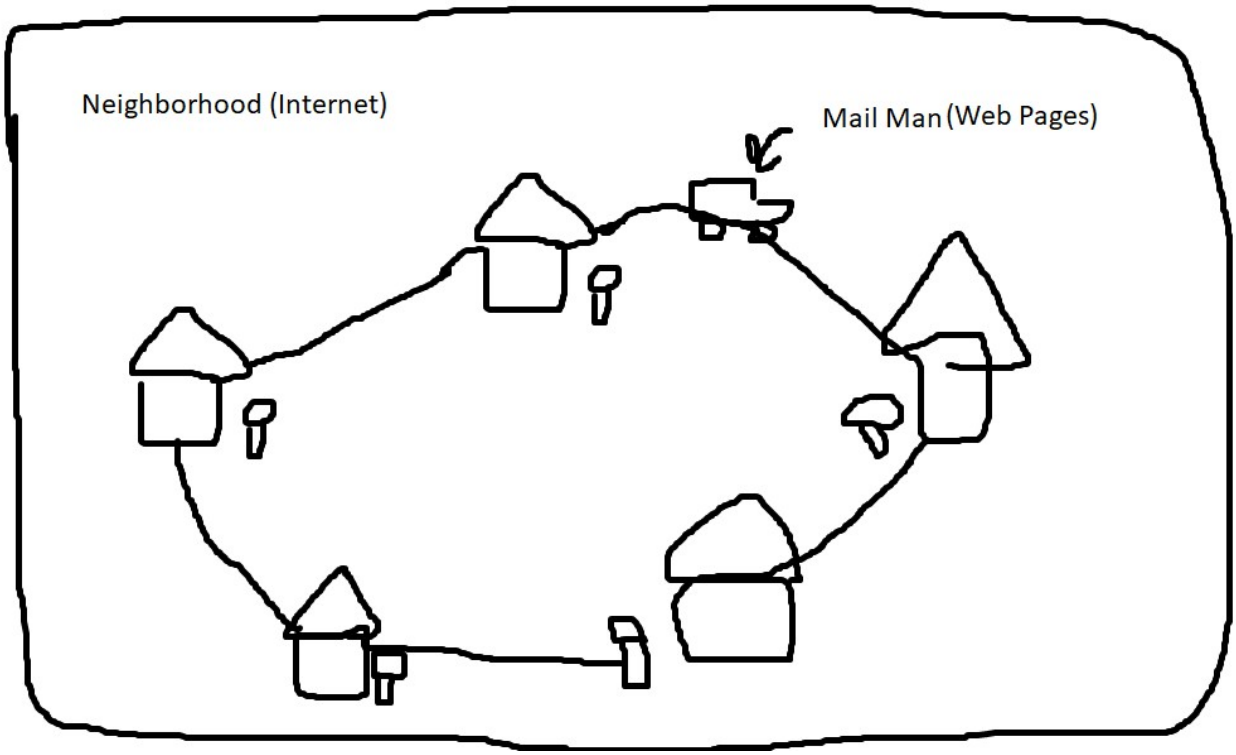
How the Web Works

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: The internet is a large network built from many small networks The basic definition of the internet is "a large network of computers communicating with each other"
2) What is the world wide web? (hint: here)
   A: The "web" is an internet application used to view web pages. There are many interconnected web pages within the web.
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: A network is a tree of communication links among computers, most commonly centralized via a router and further reached through modems and service provider cables. This gives the ability to have millions of computers be in a network and communicate with one another.
   b) What are servers?
      A: Servers are computers that hold files and web pages that will be requested by clients, for example someone browsing YouTube is a client and will communicate with YouTube's server to receive necessary files for the website to function.
   c) What are routers?
      A: A router is a small computer that will work as the center of a network so that a network of ten computers does not need 9 cables each to form a complete connection, instead will each have one cable connected to the router.
   d) What are packets?
      A: Packets is how data is sent from computer to computer efficiently. Instead of sending one large chunk of data to a computer it is broken down into many smaller chunks which is referred to as packets. This makes it possible for millions of people to receive chucks of data from a popular webpage all at once.
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: You can think of the web as a mail man delivering letters (web pages) from house to house. The houses are computers making requests from servers so its easy to think of them as houses asking to receive mail from server houses. And none of this is possible without the existence of a neighborhood where all of the houses are (the internet). With no neighborhood there are no houses, likewise with no internet there is no web.
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?
   A: domain name is an easy to remember version of an ip address. Websites can be accessed via their ip address but remembering a list of ip addresses is difficult so instead we can refer to domain names.
2) What's devmountain.com's IP address? (Hint: use 'ping' in the terminal)
   A: 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?
   A: Having your server's IP be publicly available makes it more vunerable to certain types of attacks such as a DDOS attack which is a common method to bring down servers.
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)
   A: Our computers know because of DNS (domain name system) which pairs domain names with their respective ip address. We need this because remembering the ip address of all the websites we want to visit is not a feasible method to browse the web.

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)	This is first because it starts the entire request process. If no URL is clicked there will never be a request sent out
HTML processing finishes	Request reaches app server	This is the next step after the request has been sent, the app code will now undergo its process
App code finishes execution	App code finishes execution	This goes here because the app server has finished the request execution
Initial request (link clicked, URL visited)	Browser receives HTML, begins processing	Browser will now begin to process the HTML contents that were received from the request
Page rendered in browser	HTML processing finishes	This comes after since all of the html information is done processing
Browser receives HTML, begins processing	Page rendered in browser	This is last because the page is now fully rendered in the browser meaning the request was a success

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
   A: Will print out the html content of the page to the terminal

- 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?
   A: It printed out more than just the html but also the date, time, Etag, and content-type
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?

A: Because of the used ‘get’ method I got the html tags that say `<h1>Jurrni</h1><h2>Journaling your journies</h2>`

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: Expecting to get the entries object located in the server.js file
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
- **A:** Expecting a value called “entries” with three object key/pair values located inside

3) Were you correct about the body? If yes, how/why did you make your prediction? If not, what was it and why?
   A: Yes because from looking in the server.js file we can see what the /entries command will return
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?
   A: Yes because the server.js file shows the three objects that come with “id”, “date”, and “content” keys with their respective values

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)
   A: post is a method that allows for data to be sent to a server.
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)
   A: we will be sending a json object that will include the key/pair values of id, date, and content
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.
   A: curl -i X POST -H 'Content-type: application/json' -d '{"id" : "9", "date" : "July 8", "content" : "hi"}' http://localhost:4500/entry
4) What URL will you be making this request to?
   A: 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:
   A: content type will be a json object passed to entries

- 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?
   A: Yes because the entry object gives a clear outline of the json object that will be created and passed in
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?
   A: Yes because it was only going to accept the content type specified which is a json object

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)