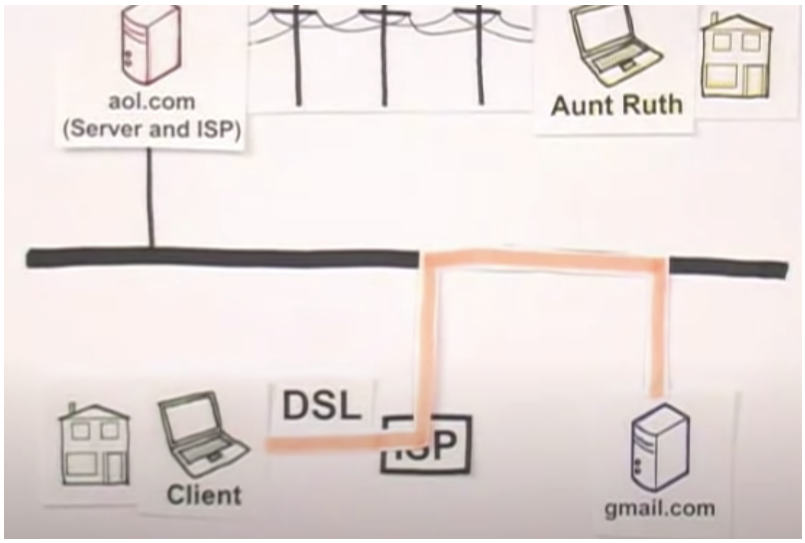
How the Web Works

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 a worldwide network of networks that allows billions of computers to be connected together.
2) What is the world wide web? (hint: here) An interconnected system of public webpages accessible through the Internet. The Web is not the same as the Internet: the Web is one of many applications built on top of 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 network is two or more connected computers which can share resources like a printer, an internet connection, application, etc.
    b) What are servers? computers that store webpages, sites, or apps.
    c) What are routers? Special tiny computer that makes sure that a message sent from a given computer arrives at the right destination computer.
    d) What are packets? When data is sent across the web, it is sent in thousands of small chunks. They are sometimes dropped or corrupted, and it's easier to replace small chunks when this happens. Additionally, the packets can be routed along different paths, making the exchange faster and allowing many different users to download the same website 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 Web is like a road. One end of the road is the client (you on your computer), and the other end of the road is the server (or site you want to access).
    a) The browser goes to the DNS server, and finds the real address of the server that the website lives on (you find the address of the shop).
    b) The browser sends an HTTP request message to the server, asking it to send a copy of the website to the client (you go to the shop and order your goods). This message, and all other data sent between the client and the server, is sent across your internet connection using TCP/IP.
    c) If the server approves the client's request, the server sends the client a "200 OK" message, which means "Of course you can look at that website! Here it is", and then starts sending the website's files to the browser as a series of small chunks called data packets (the shop gives you your goods, and you bring them back to your house).
    d) The browser assembles the small chunks into a complete web page and displays it to you (the goods arrive at your door — new shiny stuff, awesome!).

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

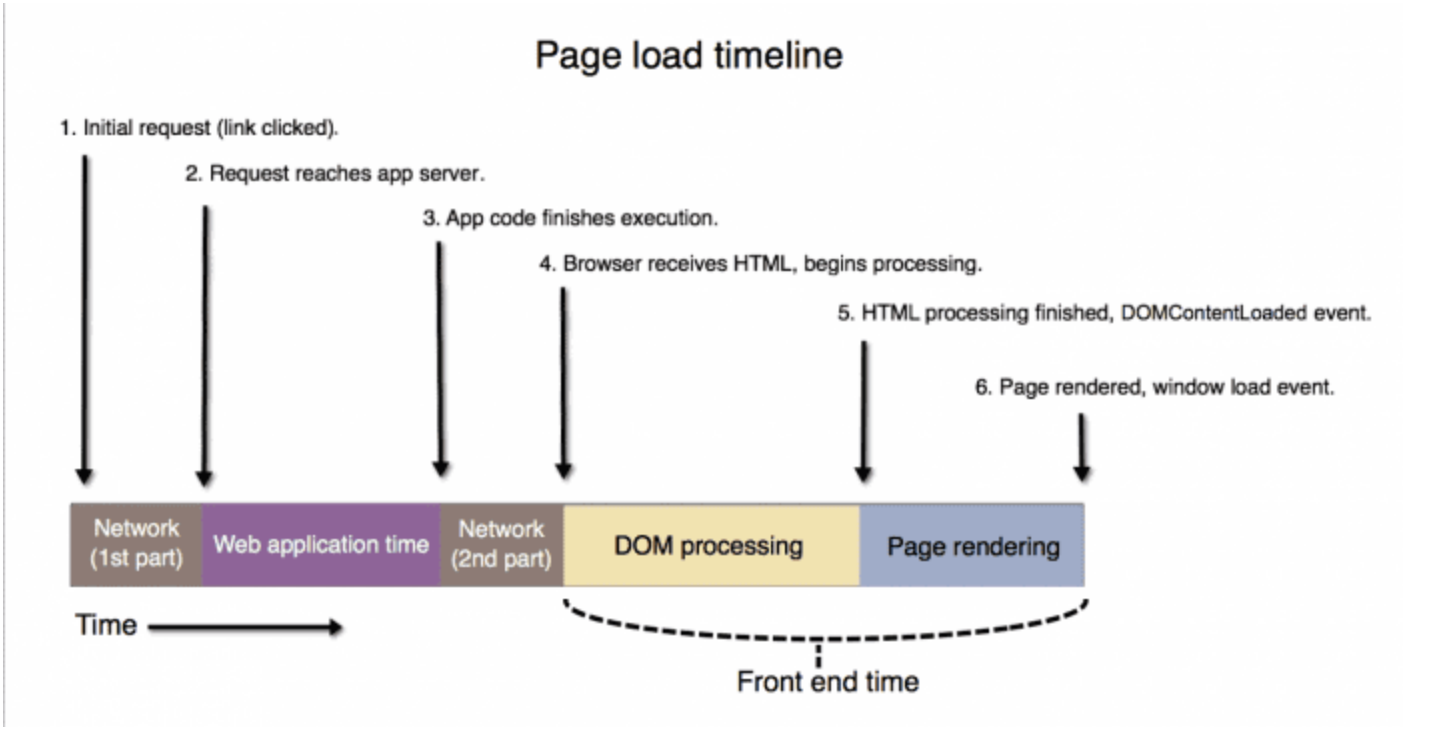
1) What is the difference between an IP address and a domain name? IP address is an address assigned to any computer (including servers) to identify it on a given network. A DNS address is a Domain Name Service which is used to convert alphabetic references into a server's IP address generally for hosting services.
    a) Example: You have a bunch of friends that you call on your cell phone all the time in your contacts. The contact is a way of going from a friend (Bill in my case) to a phone number. You want to remember me, you won't remember my phone number.
    b) The contact is a domain name, the phone number is 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? If for some reason, the IP address should have to change, you have no way of redirecting users. Whereas with a domain name, it's simply a matter of updating DNS records to point to the new IP address. Another con would be that an IP address is harder to remember than a url.
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) When we type in a website's url, it sends a signal to the dns and sends back the website's 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)	The user's action sends a request across the network to the web application server.
HTML processing finishes	Request reaches app server	The request reaches the application for processing.
App code finishes execution	App code finishes execution	The app finishes processing and sends an HTML response back across the network to the user's browser
Initial request (link clicked, URL visited)	Browser receives HTML, begins processing	The user's browser receives the HTML response and starts to process the Document Object Model
Page rendered in browser	HTML processing finishes	The DOM finishes loading; this point is known as DOM ready. Using the DOM, the user's browser starts to render the page.
Browser receives HTML, begins processing	Page rendered in browser	The page finishes rendering in the user's browser and the window load event fires.



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: `<h1>Jurrni</h1><h2>Journaling your journies</h2>`
2) Predict what the content-type of the response will be: a string

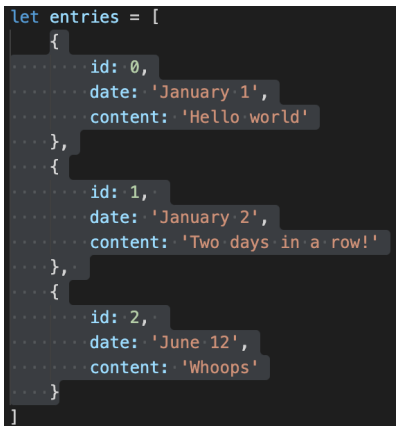
- Open a terminal window and run `curl -i http:localhost:4500`

3) Were you correct about the body? If yes, how/why did you make your prediction? If not, what was it and why? Looked at the function in the server.js file and saw the html string that would be displayed
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:



```
let entries = [
    {
        id: 0,
        date: 'January 1',
        content: 'Hello world'
    },
    {
        id: 1,
        date: 'January 2',
        content: 'Two days in a row!'
    },
    {
        id: 2,
        date: 'June 12',
        content: 'Whoops'
    }
]
```

2) Predict what the content-type of the response will be: the object '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? Looked at the function in the server.js file and saw the object 'entries' that would be displayed
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

Part C: POST /entry

- Last, read over the function that runs a post request.

1) At a base level, what is this function doing? Creates a new entry, pushes the new entry into the object 'entries', assigns a global id to the entry, and lastly, sends the 'entries' to the console to be displayed
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)? Send an id, a date, and 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": "September 13", "content": "Hello there"}'
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: my object added to the previous 3 entries
6) Predict what the content-type of the response will be: an object with the 3 values (id, date, content)

- 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 '{"id":3, "date":"September 13", "content":"Hello there"}' 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
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

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)