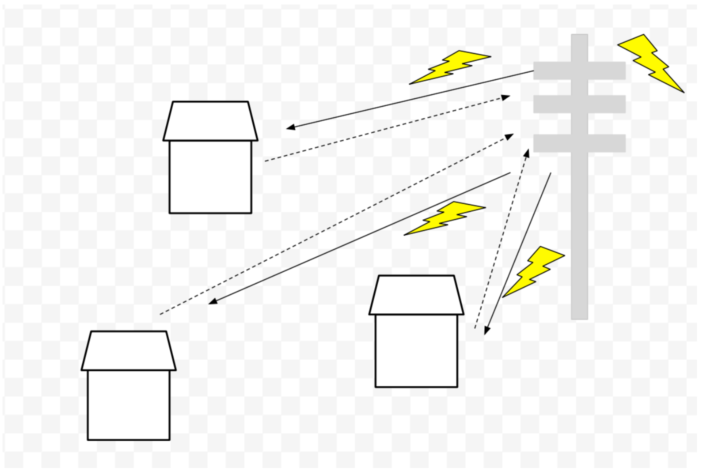
How the Web Works

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 the technical infrastructure for the worldwide network of networks that uses the Internet protocol
2) What is the world wide web? (hint: here) It is a series of public webpages that are interconnected, and are accessible 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? Two or more computers that are connected together with cables, wireless, or bluetooth. Multiple networks can also be connected through routers.
    b) What are servers? A server is a centralized computer that processes and responds to requests over the internet and amongst networks (send emails, etc).
    c) What are routers? Routers are tools/devices that connect multiple computers together and ensures that the data being sent across computers is sent (or 'routed') to the correct place.
    d) What are packets? The server sends the client the data information in the form of packets. Packets are data that is formatted into smaller chunks (like postcards sent from one city to another).
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) The internet and web is like an electrical grid and electricity (circuit). The internet is like wires that connect an electrical grid or circuit. The web is the same as that electrical current (flow of that energy) in motion. You initiate the current for the (type out a domain name for an IP address). We we turn 'on' a device, the current of energy flows through that grid to return our result (turn on a light). Similarly, when we sent a request through the web, that request flows through the internet (the grid), to return our results.

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) an IP address is a set of arranged numbers that are used to uniquely identify a computer. Humans, unlike computer's have a harder time memorizing IP addresses so we use domain names. A domain name is an 'alias' name used i place of Ip addresses to easily identify websites.
2) What's devmountain.com's IP address? (Hint: use 'ping' in the terminal)
    a) 172.66.43.107
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) It's important to protect the confidentiality and security of DevMountains private network and data.
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) There are a series of ways:
        i) All IP Addresses are registered online for easy access to resolve requests and for and record keeping.
        ii) If we've visited the website before, there are caches of stored memory
        iii) If it's not there, it goes to the resolver who takes it to the router, then the ISP, then the root (has stored domains .com, .net, .org)

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 put this first because the beginning of this process starts with an initial request.
HTML processing finishes	Request reaches app server	I put this step second because the request then travels to the server to get a response.
App code finishes execution	App code finishes execution	I put this next because the request supports with identifying what type(s) of execution needs to happen. It sets up the plan.
Initial request (link clicked, URL visited)	Browser receives HTML, begins processing	I put this next because now that the plan is laid out, the browser can receive the correct HTML layout, and can begin processing the website according to the execution laid out.
Page rendered in browser	Page rendered in browser	I put this next because the HTML will be finished processing.

Browser receives HTML, begins processing	HTML processing finishes	The final step is the fully rendered page int he browser.

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:
    a) I think I'll see: the html information about the website

```
(`<h1>Jurrni</h1><h2>Journaling your journies</h2>`)
```

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

```
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'
```

- 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) I was correct about the body. I remembered from the interactive lab that we would have information showed in < > notations.
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) I was not correct about the content type. I over thought what the term "content type" meant. I realize this may information that we will eventually add to the website.

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:
   a) The information will show up as objects
- 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?
   a) I was correct about the content in the body, however I did not realize it would show up as an array. I though it would show up as a series of objects.
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) I was correct about the type of content. However, similarly as above, I though it would show up as a series of objects, yet it showed up as a long array.

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) Declaring a post function named '**/entry**' that takes in the parameters.
   b) It contains a object called **newEntry** that has id, date, and content
   c) It then adds the newEntry into the current set of entries with the push function **entries**.push(**newEntry**)
   d) contains It then returns a status of 200 (OK)
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) Properties: id, date, and object
   b) Data type for id (**globalId**, as a number)
   c) Data type for date would be (**req.body.date** as a string)
   d) Data type for object would be (**req.body.content** as a 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:4"date":"June 13", "content":"New dawn, new day"}'
4) What URL will you be making this request to?
   a) http://localhost:4500/entries
5) Predict what you'll see as the body of the response:
   a) [{"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","id:4","date":"June 13"}]

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?

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)