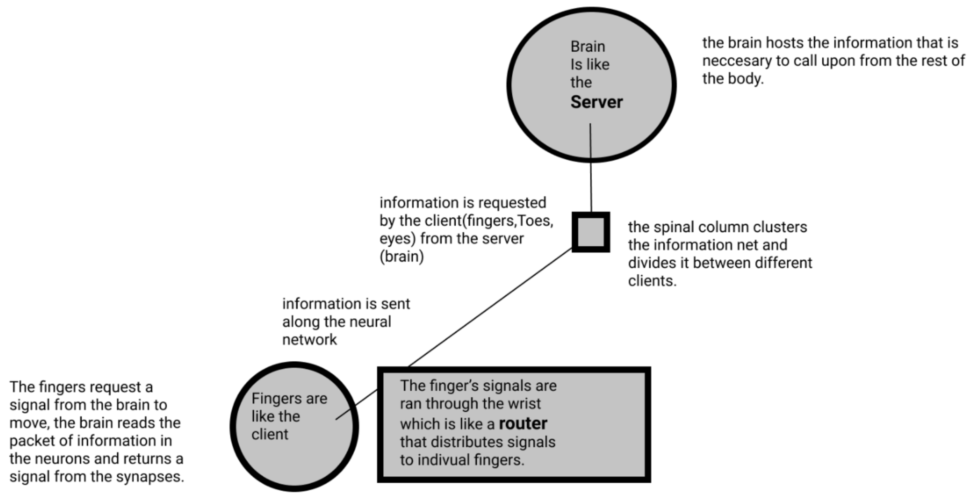
# How the Web Works

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)
2) What is the world wide web? (hint: here)
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? ***Networks are a series of computers linked together by cables to communicate with each other.***
    b) What are servers? ***Servers are computers that store data, webpages, sites or apps.***
    c) What are routers? ***Routers are simple computers with one task, to allow communication between computers on a network.***
    d) What are packets? ***Small chunks of information sent between the client and server.***
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)
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)


Brain Is like the **Server**
the brain hosts the information that is neccesary to call upon from the rest of the body.
information is requested by the client(fingers,Toes, eyes) from the server (brain)
the spinal column clusters the information net and divides it between different clients.
information is sent along the neural network
The fingers request a signal from the brain to move, the brain reads the packet of information in the neurons and returns a signal from the synapses.
Fingers are like the client
The finger's signals are ran through the wrist which is like a **router** that distributes signals to indivual fingers.

## Topic 2: IP Addresses and Domains

1) What is the difference between an IP address and a domain name? **IP is a protocol string of numbers, Domain name acts a link to the IP address.**
2) What's devmountain.com's IP address? (Hint: use 'ping' in the terminal) **172.66.40.149**
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? **To prevent DNS attacks against the IP address.**
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 DNS brings the two together, using a resolver which is typically your ISP, the resolver contacts the root server to find the IP address for the client. The root contacts the TLD where the registrar information is stored on the Authoritative name server. The TLD finds the correct IP address for the domain name submitted to it and provides that to the root. The root then returns with the correct IP address and tells the Operating system the correct IP address for the clients submission. The**

**browser then requests HTML, CSS and Javascript information in packets to display on the browser.**

## 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 | I put this step here because this is where the client initializes the request for information |
| HTML processing finishes | Request reaches APP server | I put this step next because the request has been sent but nothing has been processed yet. |
| App code finishes execution | Browser receives HTML, begins processing. | I put this step next because the APP has received the request and has sent back the HTML coding to process |
| Initial request (link clicked, URL visited) | App code finishes execution | I put this here because the app finishes sending the code back to the client before the HTML is finished |
| Page rendered in browser | HTML processing finishes | I put this here because the HTML is fully received after the APP has sent all the information and finished execution |
| Browser receives HTML, begins processing | Page rendered in browser | I put this step next because the page rendering is the final step in the request. |

## 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: **It will read out the same as the web browser,** Jurrni Journaling your journies

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

- 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, most information is sent as text, and the text that appeared in the web browser.**
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 HTML text was what was in the web browser.**

*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: **HTML code and the date and content in the function**
2) Predict what the content-type of the response will be: **HTML, text**

- 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 was an HTML string from the entries section of the server.js**
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, it was html text and was from the entries section of the server.js**

*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) **It is allowing the creation and entry of new id, date and content, it is requesting a response of Id, date, content. . Once the new information is provided it is pushing(adding) information to the host. It is then sending a response with the new information.**
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)? **Id will be a number((3) since 0, 1 and 2 are already taken), date will be a string and number, and entry is 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":3, "date":"September 27", "content": "Hello, World",**
4) What URL will you be making this request to? **http://localhost:4500/entries**

5) Predict what you'll see as the body of the response: **"id":3, "date":"September 27", "content":"Hello world"**
6) Predict what the content-type of the response will be:**HTML text**
- 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, I followed the information that was provided when we got the Get/entries.**
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, I predicted that the information would follow the formula in the id, date and content entries.**

## 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)