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 worldwide network of networks that uses the Internet protocol suite
2) What is the world wide web? (hint: here)
    a) Is an interconnected system of public webpages 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?
        i) A connection between two or more computers.
    b) What are servers?
        i) Remote computers that can be accessed from other computers on the network
    c) What are routers?
        i) A computer specifically used to communicate with other computers on the network as well as the modem and other router.
    d) What are packets?
        i) Packets are small bits of data sent from one computer 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) A metaphor for the internet that helps me understanding is like a city bus system delivering people (packets) all over the the city, you have smaller bus systems in different communities (router) that bring people to a hub (modem) where they can reach other smaller bus systems in other communities through a highway system (ISP).
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)
    a) See bottom.

Topic 2: IP Addresses and Domains

1) What is the difference between an IP address and a domain name?
    a) Domain name is points to a specific IP address, it is easier to read and remember.
2) What's devmountain.com's IP address? (Hint: use 'ping' in the terminal)
    a) 104.22.12.35
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?
    a) There may be multiple sites hosted at the same IP address by CloudFlare as well as protection for the website itself.
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 is a tiered list of tools that operate in a top down manner to locate the IP address from cached memory, if all of this is unable then a root search is run to cache any missing addresses and then it is passed back up the chain.

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 ____*
Initial request (link clicked, URL visited)	Initial request (link clicked, URL visited)	This would be the first interaction with anything related to accessing the site
Browser receives HTML, begins processing	Browser receives HTML, begins processing	Browser begins to parse the HTML file that it recieves first from the
Request reaches app server	Request reaches app server	Script or other application is initiated from HTML
App code finishes execution	App code finishes execution	Application process finished before HTML continues parsing
HTML processing finishes	HTML processing finishes	HTML has to complete process before page can load
Page rendered in browser	Page rendered in browser	Page loads and request is complete

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) Based on the GET "/" code I am assuming there will be an H1 tag with Jurrni and an H2 showing Journaling with your Journies
2) Predict what the content-type of the response will be:
    a) I would guess the content type will be JSON.
- 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) Yes, I was able to see the html tags that would go in on a GET request to "/"
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) No I assumed because it was a express server that the information would still be passed via javascript object notation but I'm guessing because there is no POST the file reads straight in to the html.

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) A list of objects defined as the entries variable in the server.js file
2) Predict what the content-type of the response will be:
    a) JSON? maybe.

- 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) Yes I was correct, I based this on the js objects we were sending with the entries variable
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 this time because they were js objects it sent them as JSON I believe.

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) Reads in the data from the request, increments the globalid, pushes the new entrie, and sends the req to the page
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) Send a body object with date and content attributes.
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) '{"date":"August 2", "content":"Frick"}'
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:
    a) We will see all of the entries including the new one we just added
6) Predict what the content-type of the response will be:
    a) 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

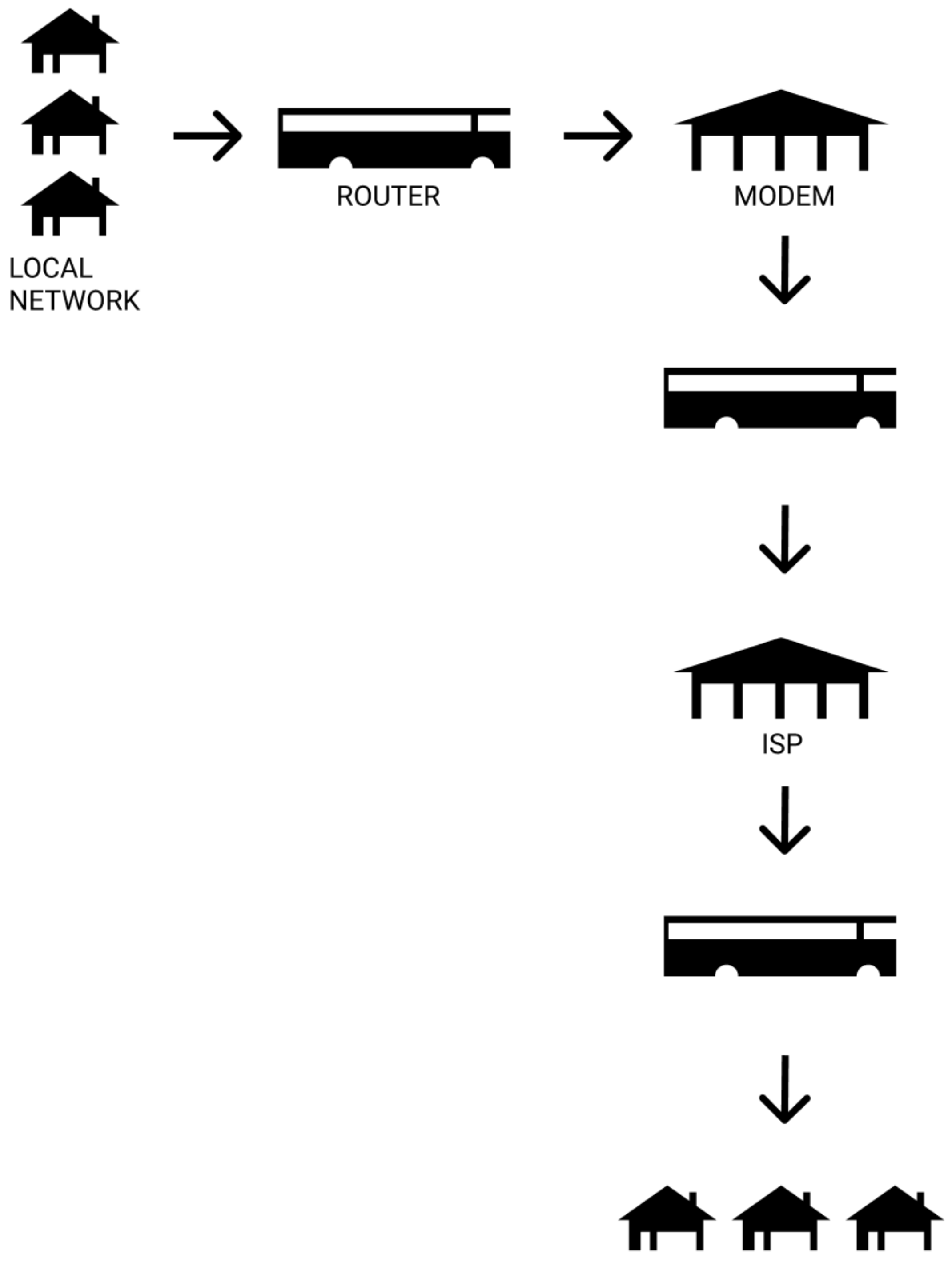
7) Were you correct about the body? If yes, how/why did you make your prediction? If not, what was it and why?
    a) Yes I was correct on the body because I learned from the other examples
8) Were you correct about the content-type of the response? If yes, how/why did you make your prediction? If not, what was it and why?
    a) Yes our array of objects was returned in JSON

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)


LOCAL NETWORK
ROUTER
MODEM
ISP