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 network that connects computers all over the world
2) What is the world wide web? (hint: here) An interconnected system of public web pages 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? 2 or more computers connected together thru wifi, bluetooth, ethernet cable
    b) What are servers? Servers are computers that store web pages, sites, or apps.
    c) What are routers? An intermediary that connect a number of different computers
    d) What are packets? The format in which data is sent from server to client.
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)

Topic 2: IP Addresses and Domains

1) What is the difference between an IP address and a domain name? The IP address is an actual set of numerical instructions / the actual address of the server. Domain name is a user friendly version of this info

2) What's devmountain.com's IP address? (Hint: use 'ping' in the terminal) 2606:4700:10::6816:d23
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? Allows hackers access to your devices / information, activity tracking, framing
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) Websites and their corresponding IP addresses are stored in a DNS server, browsers call on the DNS to find out the IP addresses that corresponds to that domain name

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)	Tells the server what we want to see
HTML processing finishes	Request reaches app server	Server receives this request before processesing request
App code finishes execution	App code finishes execution	Sends request to the consumer
Initial request (link clicked, URL visited)	Browser receives HTML, begins processing	Users browser needs to parse through and properly display information since it was received as packets
Page rendered in browser	HTML processing finishes	Browser finishes going through data and sorts
Browser receives HTML, begins processing	Page rendered in browser	Website is displayed properly

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: 'Jurrni' 'journaling your journies'
2) Predict what the content-type of the response will be: words inside h1, words inside h2
3) In your terminal, run a curl command to get request this server for /entries

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

4) Were you correct about the body? If yes, how/why did you make your prediction? If not, what was it and why? Yes – based on content in the VS code If not, what was it and why? Yes based on what was displayed in VS code
5) 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 based on the h1 and h2 tag

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: Current entry values in id0, id1, id2
2) Predict what the content-type of the response will be: Array values

3) Were you correct about the body? If yes, how/why did you make your prediction? Yes – because the function app.get/entries called on the entries array
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 function called on the 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) 1)Takes in user input info (id, date, content) 2) pushed user input info to the entries array 3)adds the new info as a new id in ascending order 4) sets the HTTP status to the same as the rest of information it shoudl belong with (200) and adds it to the entries part of the website
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 in number form, date and content both in string form
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. "car": "ford"
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: the entire array with our added content
6) Predict what the content-type of the response will be: an array with keys / properties with strings as their values

- 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

  It will look like the array with each ID and hte corrseponding string with id values displayed

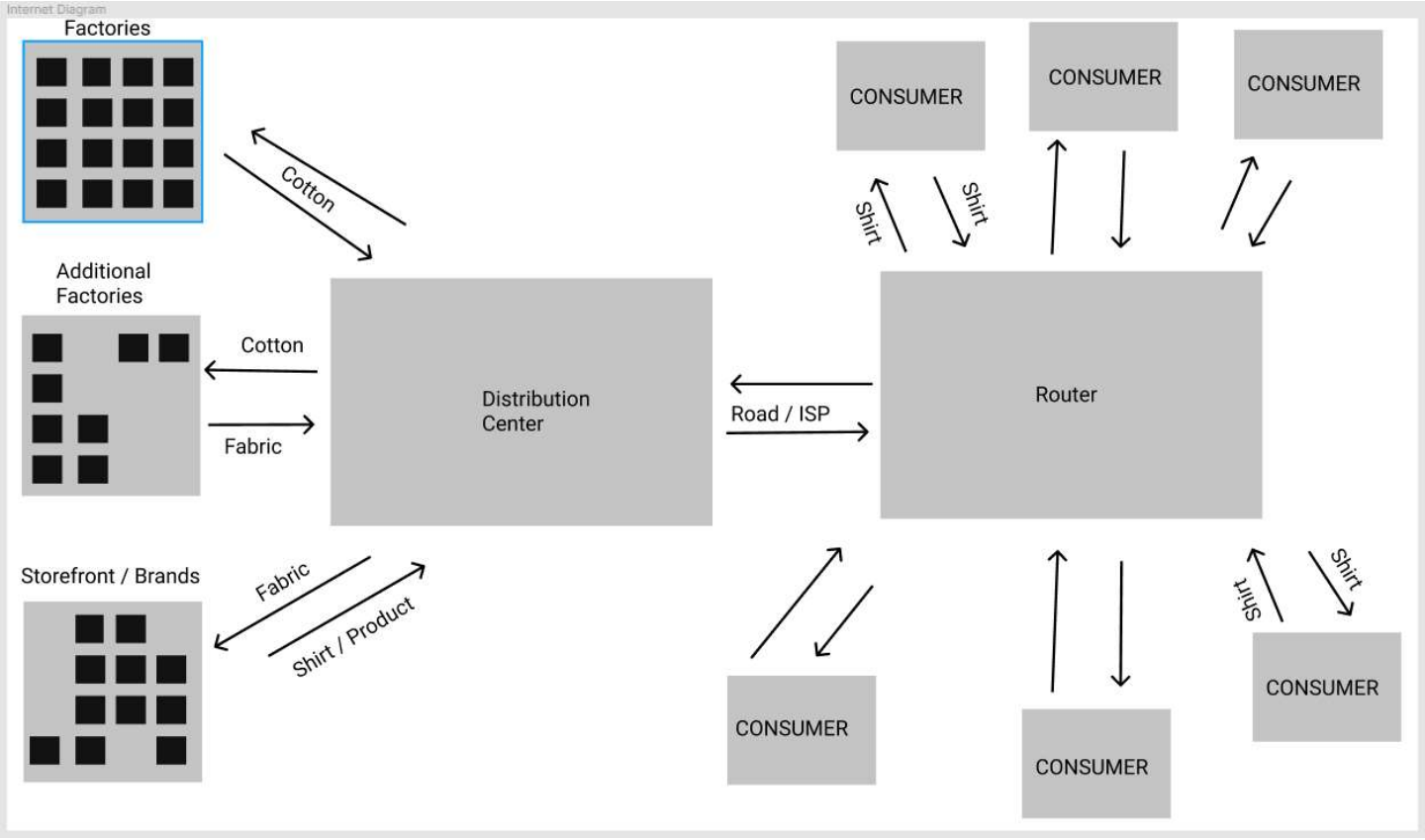
7) Were you correct about the body? If yes, how/why did you make your prediction? If not, what was it and why? Yes – it displayed what was in the array
8) Were you correct about the content-type of the response? If yes, how/why did you make your prediction? If not, what was it and why? Yes we were correct. We looked at what content was in the array

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)


Internet Diagram
Factories
Cotton
Additional Factories
Cotton
Fabric
Distribution Center
Road / ISP
Router
CONSUMER
CONSUMER
CONSUMER
Shirt
Shirt
Storefront / Brands
Fabric
Shirt / Product
CONSUMER
CONSUMER
Shirt
Shirt
CONSUMER