How the Web Works

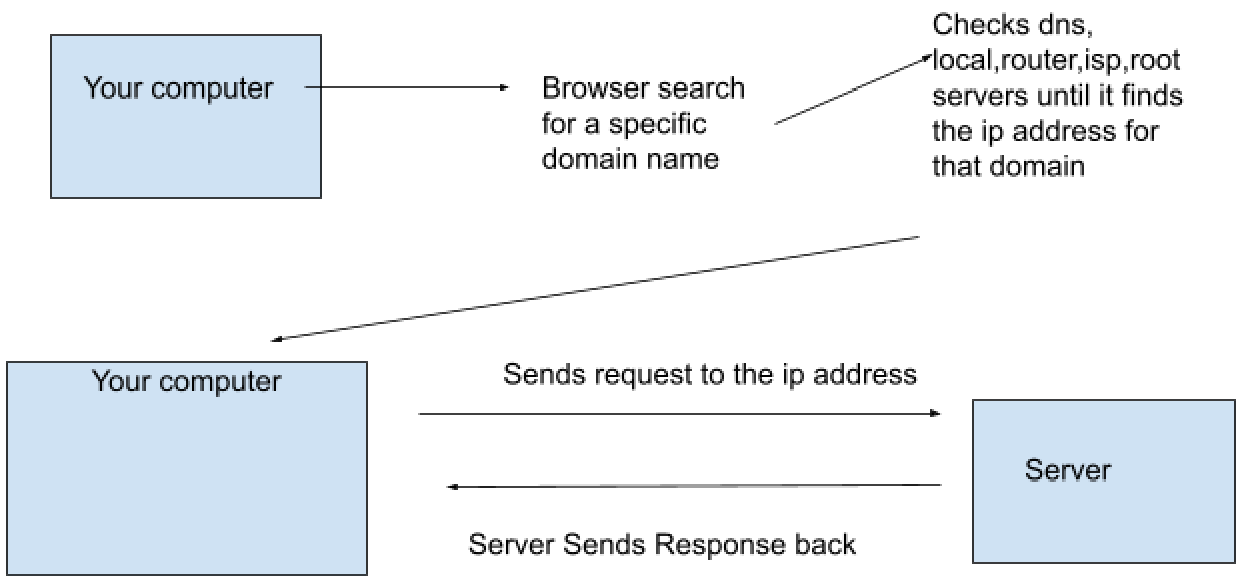
In this lab, you'll be working with a partner to explore a little more about the internet, the web, requests, responses and more. You'll be reading and writing about concepts as well as practicing some of the commands that we saw during the lecture earlier.

Topic 1: The Internet and the World Wide Web

- What is the internet? (hint: here) - a giant network of computers that are able to communicate with each other
- What is the world wide web? (hint: here) interconnected system of web pages accessible through the internet
- 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
  - What are networks? Are groups of computers that are all connected to eachother
  - What are servers? Are computers that store webpages, apps, websites
  - What are routers? Are computers that can connect to many computers to create a network from those computers without connecting each one to eachother, each can connect to the router and the router will then make sure information can be passed to each effectively
  - What are packets? Are groups of data stored in small chunks, this way if something goes wrong you can pinpoint where the issue occurred and keep better track of how much data has been transferred
- 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 internet is like a library, books are like webpages and the shelves that hold them are like individual computers that are part of the network the librarian can act as the dns, so when you the client want to access a specific book(webpage) you can go to the librarian and them the name of the book you would like to find, they will then tell you the number of the shelf it is located on(dns change name into ip address) you can then go there find the book, access it and make requests(check it out) which they will respond yes, unless the book is no longer there.

- 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)


Your computer
Browser search for a specific domain name
Checks dns, local,router,isp,root servers until it finds the ip address for that domain
Your computer
Sends request to the ip address
Server
Server Sends Response back

Topic 2: IP Addresses and Domains

- What is the difference between an IP address and a domain name? Domain names are easy to remember, but do not hold any instructions as to where the information is actually held, Ip address is a numerical value that shows the location/computer where the information is being held.
- What's devmountain.com's IP address? (Hint: use 'ping' in the terminal) 172.67.9.59
- 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? It could be a security issue which could allow users to have access to information you don't want them to see
- 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)it uses dns which allows browsers to check for what ip address is linked to that domain name, checking against its own dns cache first, then moving outward checking the router or even our isp

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	An initial request has to be made first
HTML processing finishes	Request reaches app server	Has to arrive at the server before a response is formulated
App code finishes execution	App code finishes execution	Response decision decided here
Initial request (link clicked, URL visited)	Browser receives html begins processing	Response received it needs to process before it is finished processing
Page rendered in browser	Html processing finishes	^^^
Browser receives HTML, begins processing	Page rendered in browser	Once all is completed the page can render

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
- Predict what you'll see as the body of the response:
- Predict what the content-type of the response will be:
- Open a terminal window and run `curl -i http:localhost:4500`
- Were you correct about the body? If yes, how/why did you make your prediction? If not, what was it and why?yes because when I ran it in my browser I could tell that the body would be some simple html with different sized text
- 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 just assumed that it was using 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.
- Predict what you'll see as the body of the response:
- Predict what the content-type of the response will be:
- In your terminal, run a curl command to get request this server for /entries
- Were you correct about the body? If yes, how/why did you make your prediction? If not, what was it and why? I thought it would show the entries array because it referenced it in the send method
- 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? I didn't know that it would be application/json

Part C: POST /entry

- Last, read over the function that runs a post request.
- At a base level, what is this function doing? (There are four parts to this) taking in a function as a parameter, creating a new entry and then pushing this new entry into the entries array it then sends the new entries array.
- 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)? We will need an id with type number and a date and a content both with type String.
- 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”:”globalId”,”date”:”june 4”,”content”:”hola”}
- What URL will you be making this request to? http:localhost:4500/entry
- Predict what you'll see as the body of the response: I think it will add a new entry to entries and then when i refresh it will appear on the screen
- Predict what the content-type of the response will be: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
- Were you correct about the body? If yes, how/why did you make your prediction? If not, what was it and why? Yes I was, it added the object I made onto the end of the array.
- 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 I was inputting an array of objects i was using json objects.

Submission

- Save this document as a PDF
- Go to Github and create a new repository. (Click the little + in the upper right hand corner.)

- Name your repository “web-works” (or something like that).
- Click “uploading an existing file” under the “Quick setup heading”.
- Choose your web works PDF document to upload.
- Add “commit message” under the heading “Commit changes”. A good commit message would be something like “Adding web works problems.”
- 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)