

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 answer the following questions
 - a) What are networks? A bunch of computer linked together via router.
 - b) What are servers? The Servers are computers that store the files, the host of a file or files.
 - c) What are routers? A router is consider a special tiny computer that makes sure a messages sent from a computer that is connected/delivered to another computer.
 - d) What are packets? The small piece of data through sources, html, css, and javascript.
- 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)
- The internet in conjunction with the World Wide Web is like a functional car. The Internet being the body and physical aspects of the car. The World Wide Web being the engine/motor of a car, even steering of a car.
- 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?
- 2) An IP address is a unique address to find networks. The domain name is the nickname of the IP address.
- 3) What's devmountain.com's IP address? (Hint: use 'ping' in the terminal)
- 4) 172.67.9.59
- 5) 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?
- 6) Error 1003 Direct IP access not allowed. Users shouldn't be able to use the IP address directly to the site just for security reason. Encrypted data, and personal information.
- 7) 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)
- 8) Computer communicate with numbers.

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?
<i>Example: Here is an example step</i>	<i>Here is an example step</i>	- I put this step first because ____ - I put this step before/after ____ because ____
Request reaches app server	Initial request	This should be the first thing "Initial"
HTML processing finishes	Request Reaches app server	I believe this should be the 2nd step
App code finishes execution	App code finishes execution	I put this 3rd because of the the App code should be finish before.
Initial request (link clicked, URL visited)	Browser receives HTML	HTML should start to build.
Page rendered in browser	HTML processing	HTML should start to began to display
Browser receives HTML, begins processing	Page rendered in browser	Page should be fully loaded. "Rendered" vs rendering

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: Serving Up port 4500!
 - 2) Predict what the content-type of the response will be: Jurri Journaling your journeys
- 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? No, for some reason I though my console log would display first and it didn't. Journaling your journeys appeared because it was the the send status function.
- 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?

Part B: GET /entries

- Now look at the next function, the one that runs on get requests to /entries.

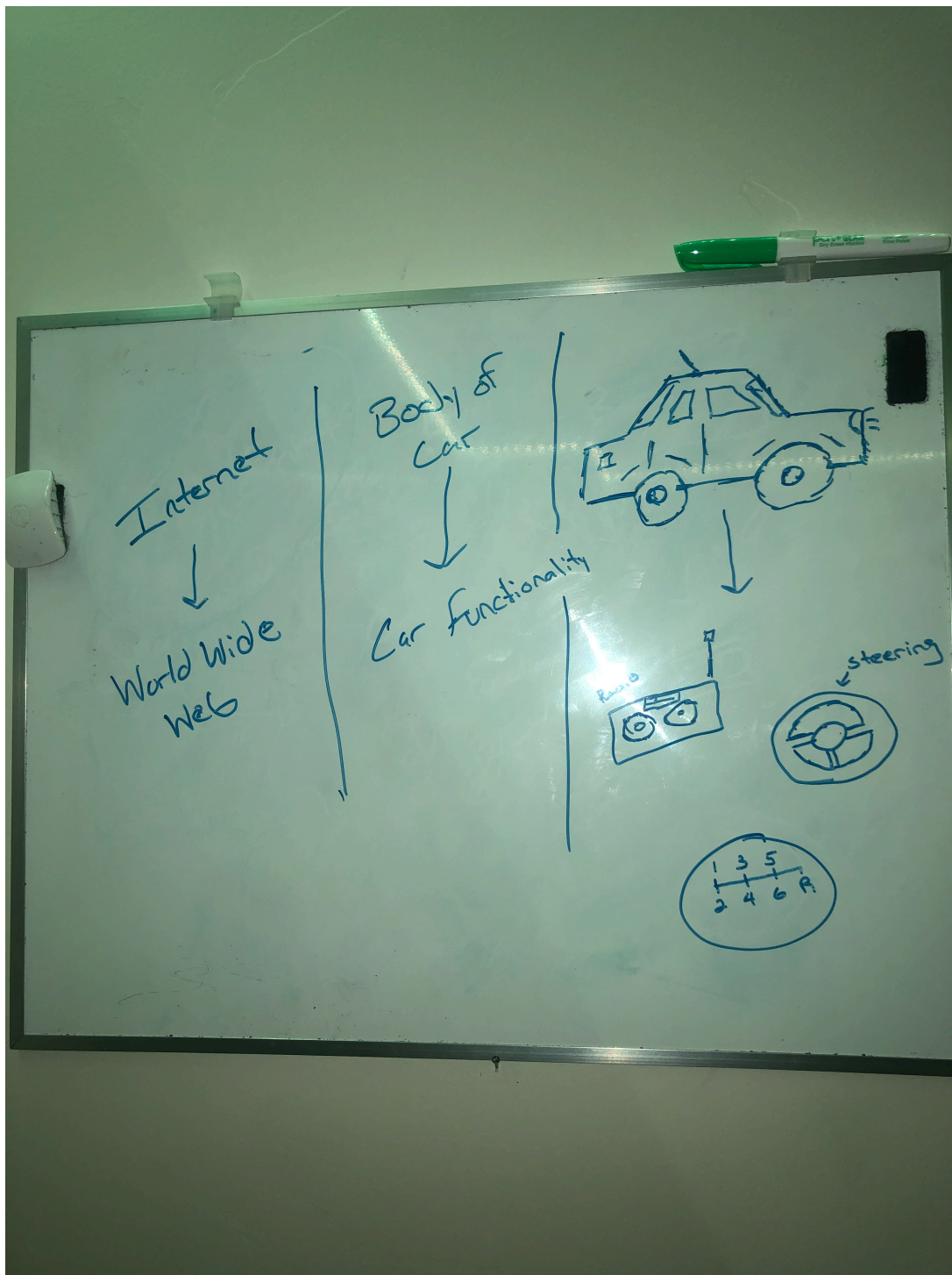
- You'll use the curl command again. This time, you'll need to figure out how to modify it to get the response that you need.
- 1) Predict what you'll see as the body of the response:
 - a) Nothing will change.
- 2) Predict what the content-type of the response will be:
 - a) Jurri Journaling your journies
- 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) No, I figured nothing would change. The (let entries objects) popped up with when I submitted that into the web browser.
- 4) Were you correct about the content-type of the response? If yes, how/why did you make your prediction?
 - a) If not, what was it and why? I was not right on my predictions, because I didn't read the syntax correctly.

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) Choosing dates and moving around the ID(s).
- 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) The properties I'll possibly be moving are the date and content properties.
- 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.
- 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:
- 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)