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)
   World wide network of smaller networks made up of interconnected computers.
2) What is the world wide web? (hint: here)
   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?
      Group of interconnected computers that communicate with one another.
   b) What are servers?
      Computers that host a web resource such as webpages, sites, or web apps.
   c) What are routers?
      Networking device that forwards data between different networks effectively linking them.
   d) What are packets?
      Small chunk of data sent through the internet.

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)

   Interstate/Highway System

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)

   https://www.figma.com/file/pHsfKQS1tG14ISxP2oVDzi/Untitled?node-id=0%3A1

Topic 2: IP Addresses and Domains

1) What is the difference between an IP address and a domain name?
   Domain names are human readable addresses that are mapped to an IP address
   IP addresses are unique numerical addresses that identify computers on the internet
2) What's devmountain.com's IP address? (Hint: use 'ping' in the terminal)
   172.67.9.59
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?
   Prevent malicious domain redirection
   Your IP address can change
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 domain name needs to be resolved into an actual IP address. In order to do that, the browser needs to check with multiple DNS servers going from a local cache all the way to the TLD DNS servers

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)	You start by making a request
HTML processing finishes	Request reaches app server	Server needs to receive a request in order to start working on it
App code finishes execution	Browser receives HTML, begins processing	HTML is the first thing that needs to be processed
Initial request (link clicked, URL visited)	HTML processing finishes	Browser starts processing all additional resources such as CSS and Js files
Page rendered in browser	App code finishes execution	After everything is compiled the browser will render the webpage.
Browser receives HTML, begins processing	Page rendered in browser	

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.

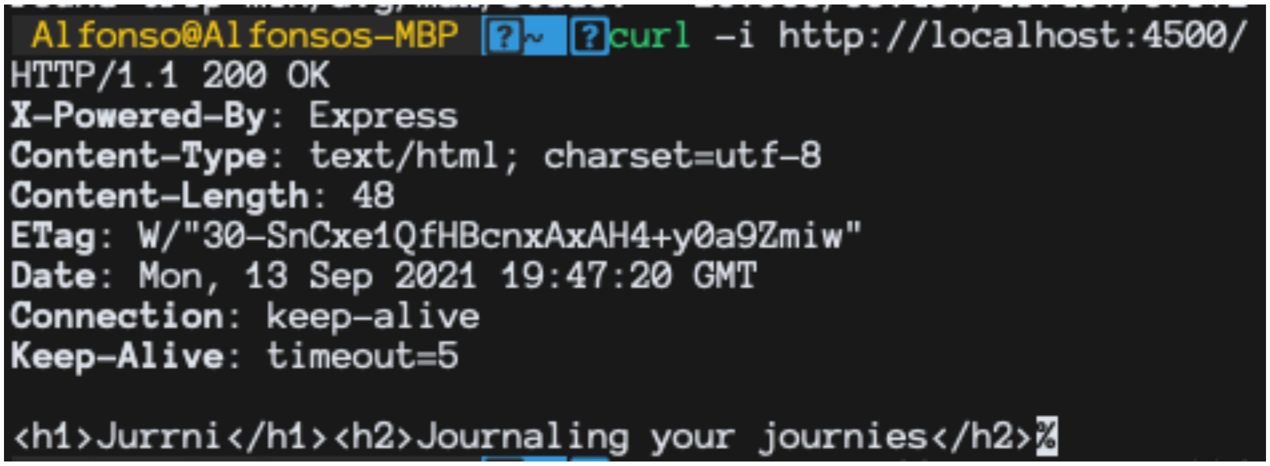
Part A: GET /

- 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:
   We gonna see a message containing "<h1>Jurrni</h1><h2>Journaling your journies</h2>%"

2) Predict what the content-type of the response will be:
   text/html because of the single html header that the server will respond with

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



```
 Alfonso@Alfonsos-MBP [?~ [?curl -i http://localhost:4500/
HTTP/1.1 200 OK
X-Powered-By: Express
Content-Type: text/html; charset=utf-8
Content-Length: 48
ETag: W/"30-SnCxe1QfHBcnxAxAH4+y0a9Zmiw"
Date: Mon, 13 Sep 2021 19:47:20 GMT
Connection: keep-alive
Keep-Alive: timeout=5

<h1>Jurrni</h1><h2>Journaling your journies</h2>%
```

3) Were you correct about the body? If yes, how/why did you make your prediction? If not, what was it and why?
   Yes, because in the server's code there is a send method that contains the H1 header before mentioned and is fired up when you access the home(/) endpoint.

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 html body is treated a text/html content-type

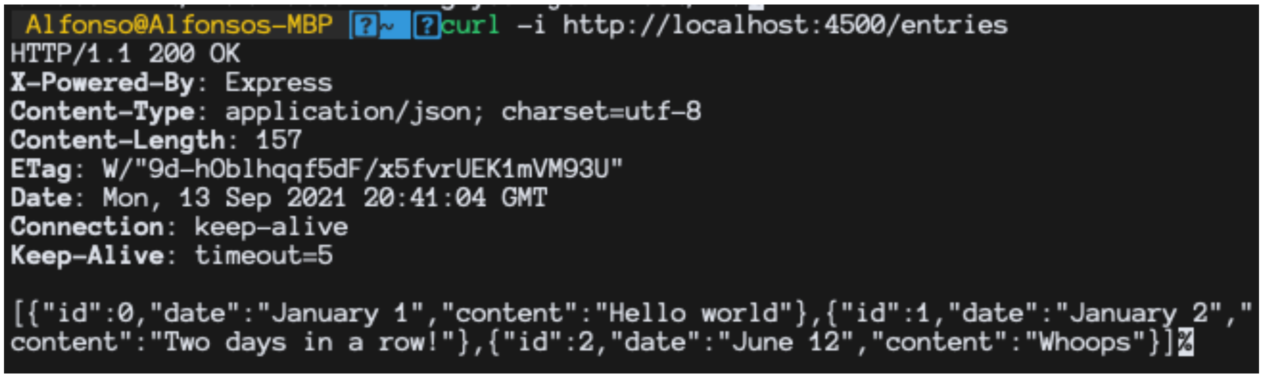
Part B: GET /entries

- 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:
   We expect to see a the array of the journal entries

2) Predict what the content-type of the response will be:
   Application/json because we are expecting an object (array)

- In your terminal, run a curl command to get request this server for /entries



```
Alfonso@Alfonsos-MBP ?~ ?curl -i http://localhost:4500/entries
HTTP/1.1 200 OK
X-Powered-By: Express
Content-Type: application/json; charset=utf-8
Content-Length: 157
ETag: W/"9d-hOblhqqf5dF/x5fvrUEK1mVM93U"
Date: Mon, 13 Sep 2021 20:41:04 GMT
Connection: keep-alive
Keep-Alive: timeout=5

[{"id":0,"date":"January 1","content":"Hello world"},{"id":1,"date":"January 2","
content":"Two days in a row!"},{"id":2,"date":"June 12","content":"Whoops"}]%
```

3) Were you correct about the body? If yes, how/why did you make your prediction? If not, what was it and why?
Yes, because in the server's code there is a send method that contains the journal entries array and is fired up when you access the /entries endpoint.

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 objects are treated a application/json content-type

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)
   I. **Creates a new journal entry object and assigns the values for its attributes using data sent on the request object.**
   II. **Adds the new entry to the journal entries array**
   III. **Increments the global id variable as each new entry object is created and added into the entries array.**
   IV. **The server responds with a 200 status code and the newly updated version of the entries array.**

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)?
date - string
content - string

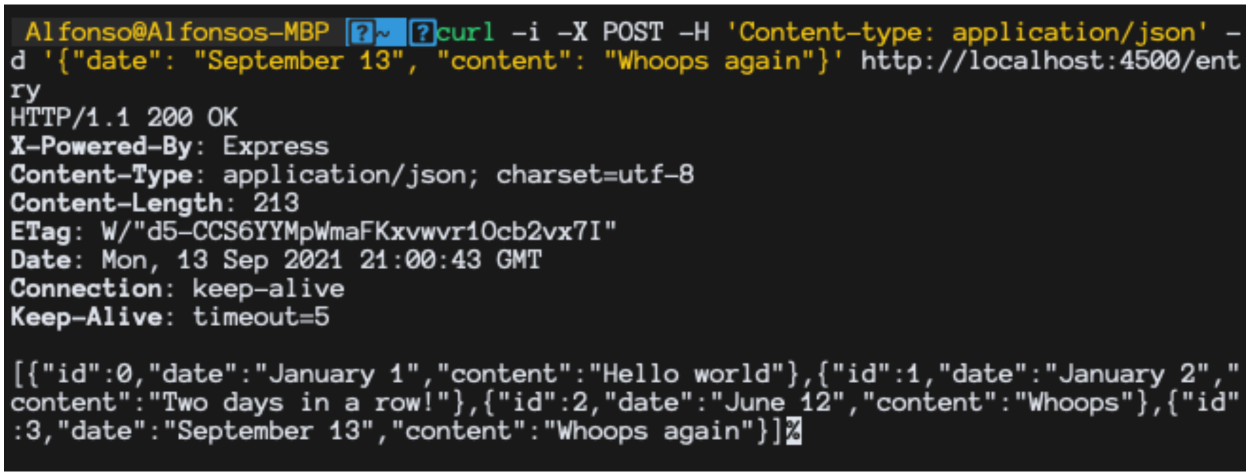
3) Plan the object that you'll send with your request. Remember that it needs to be written as a JSON object inside strings. JSON objects properties/keys and values need to be in **double quotes** and separated by commas.
'{"date": "September 13", "content": "Whoops again"}'

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 array of journal entries with the new entry we just created.

6) Predict what the content-type of the response will be:
   Application/json because the server will respond with the whole array of journal entries

- 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



```
 Alfonso@Alfonsos-MBP ?~ ?curl -i -X POST -H 'Content-type: application/json' -
d '{"date": "September 13", "content": "Whoops again"}' http://localhost:4500/ent
ry
HTTP/1.1 200 OK
X-Powered-By: Express
Content-Type: application/json; charset=utf-8
Content-Length: 213
ETag: W/"d5-CCS6YYMpWmaFKxvwvr1Ocb2vx7I"
Date: Mon, 13 Sep 2021 21:00:43 GMT
Connection: keep-alive
Keep-Alive: timeout=5

[{"id":0,"date":"January 1","content":"Hello world"},{"id":1,"date":"January 2","
content":"Two days in a row!"},{"id":2,"date":"June 12","content":"Whoops"},{"id"
:3,"date":"September 13","content":"Whoops again"}]%
```

7) Were you correct about the body? If yes, how/why did you make your prediction? If not, what was it and why?
   Yes, because in the server’s code there is a send method that contains the updated journal entries array and is fired up when you post to the /entry endpoint.

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, because objects are treated a application/json content-type