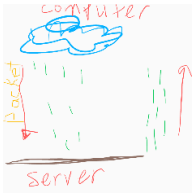
How the Web Works

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) It is a world wide network of networks that uses the internet protocol suite.
2) What is the world wide web? (hint: here)
    a) Interconnected system of public webpages accessible through the internet. The web is one of many applications built on top of 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) Two computers that communicate wireless or wired to share information
    b) What are servers?
        i) Servers are computers that store web pages sites or apps
    c) What are routers?
        i) This *router* has only one job: like a signaler at a railway station, it makes sure that a message sent from a given computer arrives at the right destination computer.
    d) What are packets?
        i) Are small chunks of data that are sent from the users to server
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) The internet is like precipitation with rain sending water(information out) then water returns to the starting computer(cloud) The ground is the server.
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?
    a) Ip address is a numbered version that a computer can read while the domain name is a human version.
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) This is important to block bot attacks and protect user information.

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) First it checks the browser's DNS cache, if not then router's DNS, if not then resolver(ISP) DNS , lastly checks the root DNS.

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)	I put this step first because nothing can happen before the request is made.
HTML processing finishes	Request reaches app server	I put this step second because after the request is made it goes to the app server
App code finishes execution	Browser receives HTML, begins processing	I put this step third because after reaching the server it sends packets back of the data
Initial request (link clicked, URL visited)	HTML processing finishes	I put this step fourth because after it begins processing it has to finish to load website
Page rendered in browser	Page rendered in browser	The CSS and HTML will be rendered in.
Browser receives HTML, begins processing	App code finishes execution	After its rendered the JS files will run and the app code will finish.

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) Jurrni
2) Predict what the content-type of the response will be:
    a) Journaling your journies

- 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, when using the curl to inspect the I saw h1 was Jurrni so I predicted that was the body
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, when using curl I saw Journaling your journeys so I predicted it would be that.

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) You will see
2) Predict what the content-type of the response will be:
    a) January 1 Hello World, January 2 Welcome back, June 12 Whoops

- 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 reviewed the information and then when I ran it showed up.
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, I was correct about the content. It displayed dates that I was expecting.

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) Listening for /entry parameters
    b) Taking in the HTTP request data, and the res is to send back to desired HTTP response
    c) Creating a New entry object and pushing the new object into the entries array.
    d) And sending the status(200) with 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)?
    a) Id, date ,content , integer and 2 strings
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) "Entries": {
    "id" : "0",
    "date": "January 1",
    "content": "Hello World"
    }
4) What URL will you be making this request to?

    a) http://localhost:4500/entires
5) Predict what you'll see as the body of the response:
    a) application/JSON
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?
    a) Yes, because we gave it a JSON object so it'd return the data in the JSON object
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, because we gave it a JSON object so it'd return a application/json content type

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)