**Topic 1: The Internet and the World Wide Web**

* What is the internet? The Internet is a worldwide network of networks that uses the Internet protocol suite (also named TCP/IP from its two most important protocols).
* What is the world wide web? an interconnected system of public webpages accessible through the internet
* Partner One: read [this page](https://developer.mozilla.org/en-US/docs/Learn/Common_questions/How_does_the_Internet_work) on how the internet works, Partner Two: read [this page](https://developer.mozilla.org/en-US/docs/Learn/Getting_started_with_the_web/How_the_Web_works) on how the world wide web works. When you’re done reading, come back together and and answer the following questions
* What are networks? a group or sysemt of interconnected computers that allow the transmission of data
* What are servers? Servers are computers that store webpages,sites, and apps.
* What are routers? A device that connects two or more networks.
* What are packets? How data is stored and sent from server to client. In many small packets, that way if anything is corrupted or lost it is easier to replace. This also allows more than one person to download the website at a time.
* Come up with a metaphor for the internet and the web: the internet is like the street that takes you to the store. The communication protocols (TCP/IP) is the car, aka the transposrt mechanism that allows you to go to the store. The servers/web is the many stores you could visit with the many goods that are at these stores. These good are then transfered to the client in packets or grocery bags.
* Draw out a diagram of the infrastructure of the internet and how a request and response travel using your metaphor <https://www.autodraw.com/share/18FGWSG7JDD1>

**Topic 2: IP Addresses and Domains**

* What is the difference between an IP address and a domain name? The IP address is the numerical address of the website, app, or webpage. The domain name acts as a readable link to the IP address.
* What’s devmountain.com’s IP address? 104.22.13.35
* 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? IP addresses change, however the domain name can stay the same. Also many doamins can share a single IP address. If you allow direct access to your server it can also allow undesirable changes to the server.
* How do our browsers know the IP address of a website when we type in its domain name? The browsers store these IPs in their cache and can check/reference them based on the domain name given.

**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) | User submits request, as we do. |
| HTML processing finishes | Request reaches app server | Request reaches server, as it does. |
| App code finishes execution | Browser receives HTML, begins processing | HTML is always processed first |
| Initial request (link clicked, URL visited) | HTML processing finishes | Once HTML finishes, CSS, then JS begin |
| Page rendered in browser | Page rendered in browser | Page is Rendered while script are running |
| Browser receives HTML, begins processing | App code finishes execution | Code is finsihed executed when the process is over. |

**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](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?
* 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.
* 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?
* 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 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)
* 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)?
* 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.
* What URL will you be making this request to?
* Predict what you’ll see as the body of the response:
* 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
* Were you correct about the body? If yes, how/why did you make your prediction? If not, what was it and why?
* 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**

* 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](https://jvns.ca/blog/2019/08/27/curl-exercises/) 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)