DAY #2

TODO:

- **Review**
- **Lesson 2:** What on Earth is a web app?
- **Lesson 3:** How Web Servers Chat
- Some time to work on your projects

REVIEW

- What system handles long-term data storage and retreival?
- What is a path?
- What makes a path absolute or relative?
- What are the key differences between a Windows and Linux FS?
- What does `IP` stand for (in IP address)?
- What are the two most common networking protocols?
- When should you use TCP over UDP?
- What must we do to code before it can be run on a computer?
- What is the difference between high-level and low-level (in computers)?

THE NATIVE APP

- Must be compiled to run on the computer.
- Has access to all the interfaces that the OS makes available.

INTERPRETED LANGUAGES

- Cannot access anything outside of their interpreters.
- Any language can be interpreted, but all languages have an official (or most common) standard.
- Some are read and executed top-to-bottom, some are only read top-to-bottom.

LESSON 2: WHAT ON EARTH IS A WEB APP?

THE WEB SERVER

- Websites are stored as a group of resources on a web server.
- The server is essentially responsible for content delivery.
- Web browsers request a web page (written in HTML) from the server.
- During loading of the page, the browser will request all needed assets from the server.

THE WEB BROWSER

- Web pages are HTML files.
- Flow of displaying a web app:
  - Render HTML top to bottom.
  - If there are any asset imports, fetch assets and process them.
  - If CSS is loaded, render using CSS engine and re-render whenever the HTML updates.
  - If JS is loaded, execute the code using the JS engine.

NOTES ON THE BROWSER

- All web browsers are different.
- Always native applications.
- Web browsers are very low-level but websites are very high-level.

SANDBOXING

- Websites live only within the confines of the web browser.
- The web browser restricts websites into a sandbox with no OS access.
- Websites can only access limited resources as allowed by the browser.
- For this reason, web apps are often thought to be limited and useless.

THE STRENGTHS OF A WEB APP

- All OS-related access (networking, filesystems, etc.) can be handled on the backend.
- The front-end is very capable of adjusting to every platform that supports a web browser.
- Nothing is ever compiled and so you can get rid of the concepts of:
  - Software updates
  - Prerequisites
  - System requirements
  - Administrator access

LESSON 3: HOW WEB SERVERS CHAT

THE NETWORKING SPECIFICS

- All web traffic happens over TCP.
- Port 80 is for regular web traffic.
- Port 443 is for secure web traffic.
- These are defaults but other ports are allowed.
- For example, entering `google.com:8080` in your browser will use port 8080.
- For the web browser, different ports = different domains.

THE HYPERTEXT TRANSFER PROTOCOL

- Protocol for communication between the web server and web browser.
- Consists of two types of data packets: requests and responses.
- All HTTP data is defined to maintain this format:
  - A single line specifying the protocol version and status code or request method.
  - A certain number of HTTP defined key-value pairs known as `headers`.
  - All data related to the request/response.

TWO MORE SPECIAL CHARACTERS

- At the base level, all text is saved as a sequence of bytes.
- Bytes are numbers that represent each individual character.
- The translation from byte to character is defined by a character set.
- The line feed is a type of whitespace character on all systems. (LF)
- The carriage return is a special character for use in console applications.
  - LF = newline on *nix
  - CRLF = newline on Windows

- CRLF = newline on Windows

ESCAPE SEQUENCES

- `\ ` = single whitespace character.
- `\n` = LF
- `\r` = CR

THAT FIRST LINE

- Different for requests and responses.
- Requests must specify the request method, resource name, and HTTP version.
- For instance:
  - `GET /some-file HTTP/1.1`
  - `POST /some-file HTTP/1.1`

HTTP HEADERS

- Consist of two important parts: a key and a value.

REQUEST HEADERS

- `Host:` defines the host you are visiting.
- `User-Agent:` a really long string defining the OS and browser you are visiting from.
- `Accept:` defines the mimetype of the content the browser is expecting.
- and MANY more...

RESPONSE HEADERS

- `Server:` like user-agent but shorter and describes the server.
- `Content-Type:` the mimetype of the incoming content.
- and MANY more...