Midterm 1

- **HTML**
    - Hyper-Text Markup Language
        - Links to other websites (hyperlinks)
        - Enhancements to our text (marking it up)
        - The way through which we communicate our document
    - Tags
        - The organizing marks of our document, that specifies the structure
            - <h1> All about Dogz! </h1>
                - **opening** to **closing tags**
        - **Attributes**
            - additional parts to our tags
            - Click me!
                - 'href' here is an attribute in the anchor tag
    - Accessibility
        - Use theming colors and structural elements that make your website useable by all
        - Proper contrast of text on background, alt attributes in <img> tags
    - **Styles**
        - The style of tags is determined by the browser that loads it
            - But we can control how the browser shows such information
        - Use style attribute
            - <h1 style="color: orange;"></h1>
            - Style of children inherited from parent
        - Basic styles
            - color -> color of text
            - background-color -> color of background
            - font-size -> size of font
            - font-style -> font style (Arial, Helvetica, Roman)
            - Look into it and play around!
- **Document Object Model**
    - Each element a child of another element, kind of an upside down tree
        - html is the parent of the body
        - body is the parent of the tags used within
        - The tags have their own children as used
- **Box Model** (all block elements in HTML)
    - The content is surrounded by the padding
    - The padding is surrounded by the outline
    - The outline is surrounded by the margin

- Development Cycle
    - Edit document
    - Save work
    - Communicate progress with client (feedback)
    - Repeat

Midterm 2

- Cookies
    - A piece of data stored on your browser that can be accessed by webpages
    - Request/Response cycle inherently stateless, cookies used for this purpose
- **HTTP** (*HyperText Transfer Protocol*)
    - Files sent across internet come from different servers, so how do they communicate?
        - HTTP is the communication protocol to transfer data between devices
    - **Methods**
        - Get -> Request a resource from server
        - Post -> Update resource on server
        - Put -> **Idempotent** update of server resource
            - Put is idempotent, so it doesn't update repeated attempts in short succession
            - Can't hit buy and end up with 50 orders by accident
        - Delete -> Get rid of resource on server
    - **HTTPS**
        - A secure (*encrypted*) connection to protect against HTTP bad actors
    - Example
        - https://www.amazon.com/How-To-Sing/dp/012?ref=sr&Keywords=happy+sing
            - https -> protocol to communicate
            - www.amazon.com -> domain name of website (where in internet located)
            - How-To-Sing -> where to go on server for resource
            - ?ref=sr&Keywords=happy+sing -> query string, parameters for request
- **JavaScript**
    - A popular dynamically typed interpreted language for web development
        - HTML - Structure, CSS - style, JavaScript - interaction
    - Not *actually* related to 'Java'
        - Named such as Java was popular at the time of creation
    - Asynchronous Execution
        - One of the most powerful aspects of javascript
            - Allows for concurrent execution of code
            - Large amount of HTTP requests and responses at varying speeds
    - **Client-Side**
        - JavaScript that is sent and ran through the individual users browser
        - Typically used to update website dynamically, or send requests to server
        - Part of the websites *frontend*

- **Server-Side**
  - JavaScript that runs on the server of the website
  - Typically handles requests and responses for website, as well as server resource use
  - Part of the websites *backend*

Midterm 3

- Ports / Local Server
    - 'localhost' or '127.0.0.1' for hosting off computer
    - Computer can have several types of servers runnign on it
        - 80 - HTTP
        - 80 - HTTPS
        - 80 - FTP
        - 80 - SSH
        - 3000 - General Development
- **Model-View-Controller Architecture**
    - Generally, webpages are broken into three parts:
        - **Model** -> Data of the webpage **(updates view)**
            - data represented in dedicated data structures
            - Databases, objects (JSON), tables (CSV, Excel)*Don't do this one, just data no changes
            - Tables typically expand rows to new data, not columns
        - **View** -> Visual Representation of data (What is seen)
            - Typically made of HTML, CSS, and JavaScript
            - Templating languages allow for iteration and conditionals in view
        - **Controller** -> Behaviour of the webpage (What is used)(**manipulates model**)
            - Manipulates the data model through the view
    - Thin Clients
        - Most of MVC application lives on the server
    - Thick Clients
        - Most of MVC application lives on the client side
- **NPM -> Node Package Manager**
    - NPM acts as the package manager (**like pip3 install**)
    - installs to current directory (so just in project)
    - Essentially a repository for software
    - How to use
        - npm init (within directory, sets up with you)
        - npm install [package_name] (Adds package to /node_modules where software actually installed)
        - using '-g' installs gloabally, as we did with pug
        - .gitignore node_modules to share with others, as just have to 'npm install' in directory to get all dependencies
- **Express use (tie it all together)**
    - express --view=pug [new_project_directory_name]
    - Change around app.js by adding new routers for each view (each new page)

    - Create layout to extend on other views for standard portions of pages shared across all
- **Example Request (How does it happen?)**
    - User enters a URL into browser (and the point to go i.e. localhost:3000/schedule or just .../ with home)
        - /schedule goes to router in app.js and routes folder
        - router takes URL and sends to different views
            - '/' to home view
            - '/schedule' to schedule page view
- HTML and CSS and JavaScript rendered serverside/clientside once sent