NODEJS / LIVE CODE NOTES

- Express makes spinning up an HTTP server easier.
- Pattern of requiring middleware and then using app.use or app.engine to set them to it is common practice.
- REQUIRE IT, REGISTER IT, USE IT.
- Nodemon does not reload files until told to rs, its not watching things in the templates directory, just the root project directory.
    - Add script to package.json so we never have to worry about the one that gets it to watch everything again.
- MVC = Model, View, Controller
- *Find* finds **first element it returns true for and returns the object,** *filter* returns **all elements that match as a new array**.
    - Array.find((newvariable) => { return newvariable.id == 3 }) <— or whatever condition you want to check inside the {}
    - Array.filter((newvariable) => {return newvariable.id ==3 }) <— or whatever condition you want to check inside the {}
- / is for absolute path, . Is for relative

Live Coding Notes - User Directory App

- Make directory for project
- Go into it and open atom -> make package.json -> “private”: true
- Make .gitignore file -> node_modules/ inside it
- Git init -> git add . -> git commit -m “first commit -> hub create -> git push
    - Do this at the beginning just to have a thing to start from.
- Npm install express, npm install mustache-express
    - These are installed into the local project directory while we are inside it in the terminal.
    - EXPRESS IS HOW WE SET UP OUR WEB SERVER.
    - Mustache is separate, mustache-express ties the two together.
- New file -> server.js
    - Const express = require(‘express’)
        - Express is name of library, requiring express package into app
    - Const app = express()
        - Build the express app.
    - Const mustacheExpress = require(‘mustache-express’)
        - Require mustacheExpress so that it can be used throughout the project.
    - Const data = require(‘./data’) (added after we began looking at the data)
        - Bring in data from local file
    - App.engine(‘mst’, mustacheExpress())

        - Teach app that there is a middleware engine that comes from mustacheExpress.
        - Teach our app to use the mustache engine for rendering templates.
    - App.set('views', './views')
        - Teach our app what directory to find our views(templates)
    - App.set('view engine', mst)
        - Teach app to use mustache for our template
    - App.get('/', (request, response) => { response.send("Hello world") })
        - //Testing to show fellow world onto page the default localhost page "/" (commented out later)
        - response.render('home', userDirectory) ((created after send is commented out to tell express where it gets its content) (userDirectory was added after we imported the data, stores the data import in the variable)
            - Taught it that mst is mustache, things are in the home folder, etc.
    - App.listen(3000, () =>{ console.log('Hooray our app is listening) })
- Committed & Pushed after hello world happens
- Decided to mock it up in the HTML & CSS first before we build with dynamic data.
- New folder -> views
    - New File -> home.mst
        - Placed hello world into it for test without HTML stuff. (Sanity check that was deleted later)
        - Added HTML -> title User Directory
        - Moved hello world into p tag inside body.
        - Made h3 tags for robot name and p tags for other elements to mock up the index page before data is passed to it.
        - Began styling in CSS
            - body{ font-family: sans-serif; font-size 14px}
            - {Display flex on users(which is the ul, each robot is an li) in css, flex flow row wrap, list-style: none}
            - .users li {flex basis calc(25%-2em), border: thin solid color, border-radius: 5px, padding & margin 1em}
            - MOAR STYLING IM TOO LAZY TO WRITE
        - Gavin imported the HTML from Jason via copy paste into the mustache file.
        - New folder -> public
            - New file -> screen.css(due to Jason's naming convention)
            - Copied Jason's CSS & threw it into screen.css
            - Taught the app to use the CSS
                - app.use(express.static('public')) (PUT IN THE SERVER.JS FILE)
- Saved Data as data.js outside of the folders but still in the project.

    - Look and understand shape of data, we see an *id*, a *name*, a *picture*, a *job* and a *company* on our index mockup(all of these are also in the data).
    - See the null existing for the available for hire bots.
    - Data is an array of users, we have an object for every element. *Important*
- We are now inside the index mustache page!
    - Created a mustache loop in the ul with LI inside the loop. {{#users}} stuff
{{/users}}
        - # = start using the variable, / = stop using the variable
        - Inside of the loop it gets the keys
        - Put {{name}} instead of stuff
        - Took Jason's example mock LI and threw it up into the loop. Started changing syntax to match each variable. ALL OF THIS GOES INSIDE THE LI IN THE LOOP.
    - Created a case inside the loop for if they do not have a job/employer.
        - Around the P tags in the loop that shows the job and company if they exist we created a new loop of {{#job}}P tags with job stuff{{/job}} {{^job}}P tags for jojobas {{/job}}
            - # = exists, ^ = null/does not exist
    - Added skills to demonstrate looping over an array inside the objects
        - {{#skills}} {{.}} {{/skills}} = loop syntax to pull each element inside the array in the object. The dot is the individual element each time the loop happens.
    - Gavin takes the a href tag and links it to each person's individual page.
        - "/info/{{id}}" <— goes into the <a href = ""> part.
        - Went to server.js
            - app.get('info/:id', (request, response) =>
              { response.json(foundUser) })
                - Shows "info about user ___" on the page that we go to once we click.
                - Const requestID = parseInt(request.params.id)
                - Const foundUser = userDirectory.users.find(user => user.id === (request.id)
                - Response.render('info', foundUser)
                    - Render info based on the founder object
- Meanwhile Jason is creating the HTML & CSS for the details page. Who's got two thumbs and is too lazy to be writing more CSS? This guy.
    - Copied them over and created a new file in views called info.mst to paste Jason's html into.
    - Changed the href for screen.css by adding a / in front of screen on both pages.
- Now we are on the info page
    - Changed up Jason's HTML with loops similar to how it was done in the index page.

Q & A After Done

- Q: Can we try stuff out like manipulating objects in console? A: Yes, in fact we should. Enter that mode by just typing node. .exit or CMD+D to exit.