What Did You Learn?

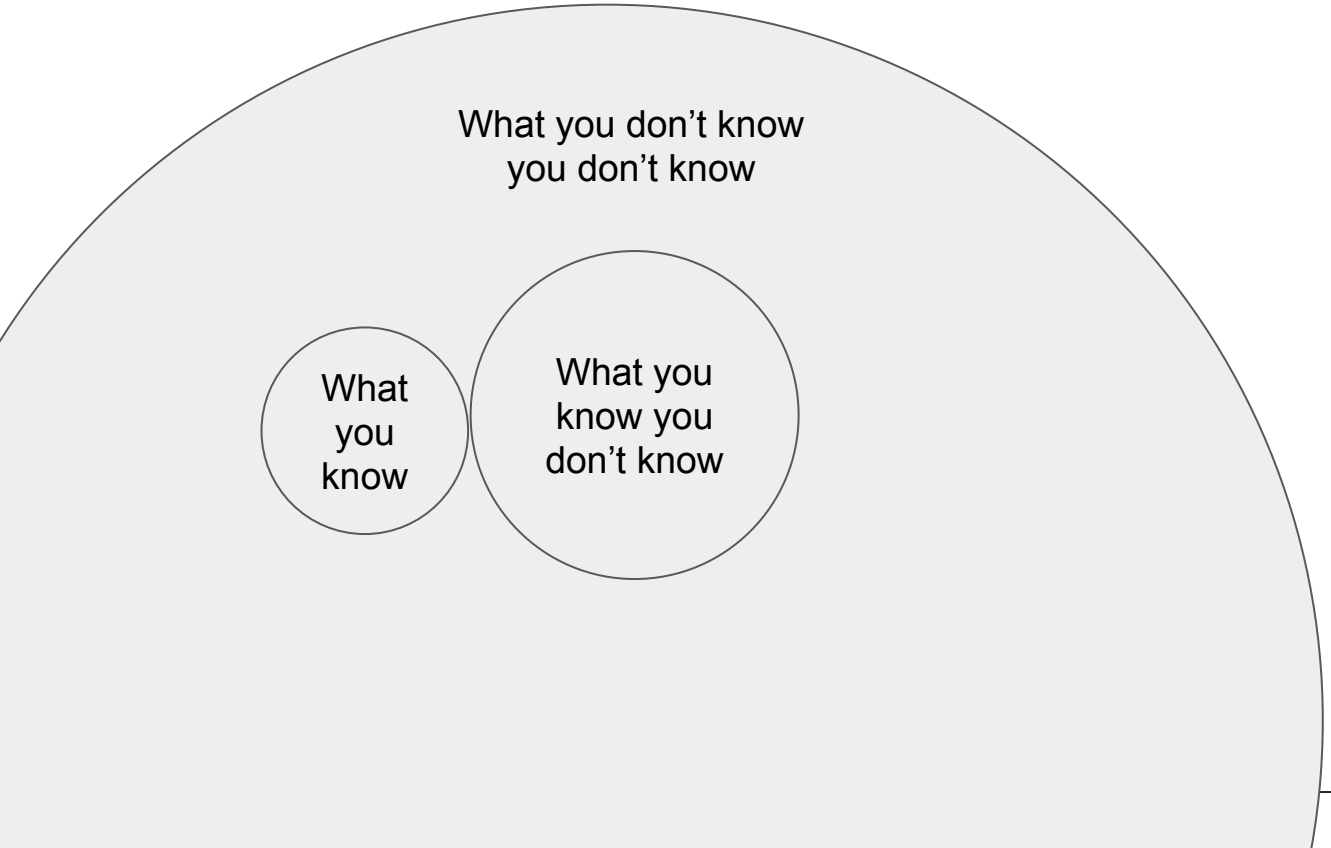
401 Advanced FullStack JavaScript

Before We Start...

Good to Look Back and See What You've Learned

- You should be impressed!
- Let's talk through these, verbiage is important too!
- Okay if aspirational!

What Do You Really Know?



Full Stack

What do you do?

Develop full stack applications:

- ExpressJS web server with a RESTful API and secured with JWT
- Connected to MongoDb using Mongoose as an ODM (Object Document Mapper)
- Front end built with React and React Router using a modern component architecture
- Leveraging Redux and immutable data for state management

MEAN -> MERN

Mongo Express React Node

NodeJS

Build, test and deploy front and back end JavaScript applications

- npm and package.json
  - Install, manage and track dependencies
  - Consolidate and standardize scripts used in development
- Mocha
  - Write and organize tests
  - Make valid assertions
  - Asynchronous testing
- Modules
  - CJS (Common JS) module.exports and require
  - ES Modules
    - export and import
    - Named exports
- Operating System
  - File System
  - Network

Coding Practices

“How” as well as “What”

- TDD
  - Red, Green, Refactor
  - Drive Design
  - Regression “safety net” for refactoring
- Clean Code
- Whitespace, phrasing
- Naming things
- Modularity - SRP "One Concern"
- Remove duplication in code or concept

ESNext JavaScript

Follow current industry use and practices

- Arrow Functions
- Promises
- let and const
- Template strings
- Destructuring
- Rest and Spread
- Default Parameters

Asynchronous JavaScript

Navigating the Node Event Loop

- Styles
  - Callbacks and node standards
  - Promises
  - async/await
- Workflows
  - Sequential
  - Parallel
- Handle error propagation correctly

Data Structures and Algorithms

Choose the right data structure for the job

- Objects as “classes”
- Objects as “dictionaries”
- Arrays
  - Functional array methods
  - As stacks and queues
- Trees
  - Recursive data structures
  - Binary Trees
  - Traversal
  - Sort/Findings
- Operational Complexity
  - Time and Space
  - O(1), O(logn), O(n), O(n*logn), O(n^2)

Binary Data

1101 0110

- Bits, bytes, words, dwords
- Use of hex to represent bytes
- Buffer
  - > NodeJS interface into binary data
  - > LE, BE UInt U

Streams and Event Emitters

- Events
  - Subscribe via function
  - Respond to events
- Streams
  - Event Emitters
  - Reading and writing data in chunks

HTTP

Web Communication

- Create vanilla Node http servers
- Request
  - Server (host and port)
  - Url (path)
  - Query
  - METHOD
  - Headers
- Response
  - Content-Type
- RESTful
  - Resources

MongoDB

Save the data, Read the data

- Install and setup
  - > Dev environment
  - > Travis CI
  - > Add to Heroku
- CLI and Robo 3T
- Manage Documents
- Aggregation Pipeline
  - > Design pipelines
  - > Use tooling like Robo 3T for rapid feedback
  - > Integrate into NodeJS

ExpressJS

Most commonly used web framework for NodeJS

- Organize routes and manage http
- Use 3rd party middleware
  - body-parser, logging (morgan), static files
- Write middleware
  - authentication
  - Authorization
  - Error handling
- Common error handling
- Structure project for:
  - end-to-end testing
  - dev and production use

Mongoose ODM

Eliminate Garbage In/Garbage Out

- Schemas and Models
  - > Model data and relationships
  - > Validate and enforce data constraints
  - > Organize model logic using static and instance methods
- Use models in routes to issue commands to Mongo

Authentication and User Management

Securing the Server

- Authentication
  - Are you who you say you are?
- Use bcrypt to manage user/password authentication without storing the password
- Create routes for users to signup and signin
- Use JWT to issue and validate tokens for authenticated users
- Use middleware to “protect” routes
- Authorization
  - Are you allowed to do this?
- Maintain user roles
- Use middleware to enforce roles for access to routes

Deploy to Heroku

Push to Production

- Team deployments to a cloud-based service provider
- Manage environment variables
- Add additional services like mLab (Mongo)

React Presentation

HTML in JavaScript

- Use JSX to write HTML
- Know JSX output is resolved via reconciliation process to minimize DOM updates
- Interpolate data with presentation logic
  - Children
  - Attributes
- Use JavaScript to compose UI
  - Map arrays
  - Compose via functions and classes
- Pass props from parent components to child components
- Adhere to one-way data flow
  - Data down
  - Events up
- Use this.state and setState to manage state in components
- Forms
  - Uncontrolled - source of truth is html controls
  - Controlled - React state is source of truth

React Router

Manage the App “State”

- Integrate with React via Router component
- Add Links and NavLinks components that modify the url
- Add Switch and Route components that react to change in the url
- Use query parameters to save additional state in url
- Pass and match, history and location in components

Container vs Presentation Components

- React Component Types
  - Function - only takes props and has no state
  - Class - extends Component
    - can have state
    - has lifecycle methods
- Strive for:
  - Presentation Components to be pure functions that only take props
  - Container Components to manage data and state

Service Modules

AJAX Requests to the Server

- Encapsulate http communication to reduce boilerplate and configuration headaches
- Create domain abstractions for use by components or action creators

Redux

Separate State Management from Presentations

- Redux
  - Store
    - Create with reducers
    - Holds state accessible via .getState
    - Receives actions via .dispatch
    - Publishes changes via .subscribe
  - Actions
    - Atomic operations dispatched to store
    - Have a type and optionally a payload
  - Action creators
    - Functions that create actions
  - Reducers
    - Discrete pure functions for managing changes to state
    - Changes follow immutable data principals
    - Can be combined to manage a state tree
    - Typically initialize state

React-Redux

Integration helpers for using Redux with React

- Integrate with React via Provider component that takes a store prop
- connect creates Container components around Presentation components
  - mapStateToProps - limit state to only what component needs
  - mapDispatchToProps - inject action creators that component needs
  - ownProps - props passed by parent component

Async Redux

Thunks FTW!

- Add middleware to Redux like redux-thunk
- Action Creators that return functions
  - > Called back with .dispatch and .getState
  - > Make multiple dispatches, over time
  - > Access current state of store

Manage CSS

Classic and CSS in JS

- Use SCSS
  - CSS Nesting
  - Variables
  - Shared files
- Use node-sass to transpile scss into css
- CSS in JS (Styled Components)
  - Create styled react components

User Authentication

- Manage and detect JWT tokens
- Route user to signup/in page for private routes
- Gather credentials and submit to server
- Integrate token with service apis for token submission


alchemy code lab
developing the future