Your Next Week

Saturday May 2	Sunday May 3	Monday May 4	Tuesday May 5
6:30 PM **— DUE Class 14 Mock Interviews** **— DUE Class 14 Lab** **— DUE Class 15 Reading** — Class 15 — Interview Prep 02 *MIDNIGHT* **— DUE Class 15 Learning Journal**	*MIDNIGHT* **— DUE CCW #1 Completed Personal Pitch** **— DUE CCW #2 Completed Resume** **— DUE Class 14-15 Feedback**		*6:30 PM* **— DUE Class 15 Lab** **— DUE Class 16 Reading** — Class 16A
Wednesday May 6	**Thursday May 7**	**Friday May 8**	**Saturday May 9**
6:30 PM — Class 16B *MIDNIGHT* **— DUE Class 16 Learning Journal**	*6:30 PM* — Co-working		*6:30 PM* **— DUE Class 16 Code Challenge** **— DUE Class 16 Lab** **— DUE Class 17 Reading** — Class 17 — Interview Prep 03 *MIDNIGHT* **— DUE Class 17 Learning Journal**

What We've Covered

Module 01

Javascript Fundamentals and Data Models

C01 — Node Ecosystem, TDD, CI/CD
C02 — Classes, Inheritance, Functional Programming
C03 — Data Modeling & NoSQL Databases
C04 — Advanced Mongo/Mongoose
C05 — DSA: Linked Lists

Module 02

API Servers

C06 — HTTP and REST
C07 — Express
C08 — Express Routing & Connected API
C09 — API Server
C11 — DSA: Stacks and Queues

Module 03

Auth/Auth

C10 — Authentication
C12 — OAuth
C13 — Bearer Authorization
C14 — Access Control (ACL)
C15 — DSA: Trees

Module 04

Realtime

C16 — Event Driven Applications
C17 — TCP Server
C18 — Socket.io
C19 — Message Queues
C20 — Midterms Prep

Midterms

Module 05

React Basics

C21 — Component Based UI
C22 — React Testing and Deployment
C23 — Props and State
C24 — Routing and Component Composition
C25 — DSA: Sorting and HashTables

Module 06

Advanced React

C26 — Hooks API
C27 — Custom Hooks
C28 — Context API
C29 — Application State with Redux
C30 — DSA: Graphs

Module 07

Redux State Management

C31 — Combined Reducers
C32 — Asynchronous Actions
C33 — Additional Topics
C34 — React Native
C35 — DSA: Review

Module 08

UI Frameworks

C36 — Gatsby and Next
C37 — JavaScript Frameworks
C38 — Finals Prep

Finals

Lab 14 Review

Code Challenge 14 Review

Class 15

DSA: Trees

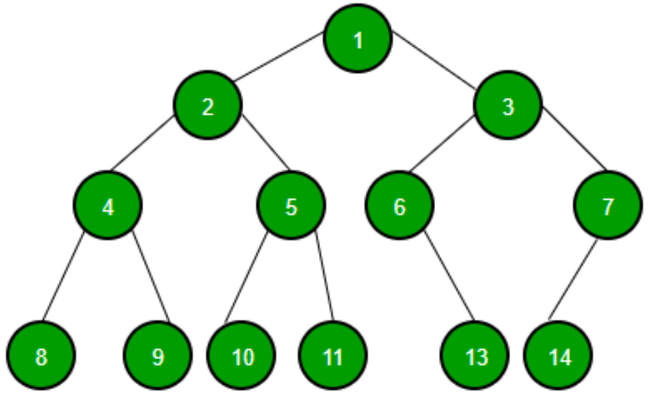
seattle-javascript-401n16

Traversal

- Trees are powerful because there's multiple ways to **traverse** them
- Depth-first traversal is the most popular, and there are three types!
    - Pre-order, in-order, post-order
- Breadth-first traversal has its own benefits as well

Trees

- The **tree** data structure consists of parent and child nodes
- A parent can have any number of children, and each child can have its own children
- The top of the tree is the **root**
- The childless ends of the tree are the **leaves**
- We like **binary trees** the most!



Traversal

- Trees have multiple ways to **traverse** them
- **Depth-first traversal** is going top-down, and there are three types
  - PreOrder, InOrder, PostOrder
  - Uses recursion
- **Breadth-first traversal** is going left to right, using a queue

Balanced Trees

- A **binary search tree** is a sorted binary tree
    - We can find things quicker because we can cut out half of the tree each layer
- Time complexities for binary search tree searching is better when a tree is **balanced**
    - `O(n)` for unbalanced, `O(logn)` for balanced
- A balanced tree has the same number of left and right `descendants` (two nodes per parent)

Lab 15 Overview