Your Next Week

Tuesday April 28 *6:30 PM* — **DUE Class 13 Lab** — **DUE Class 13 Code Challenge** — **DUE Class 14 Reading** — Class 14A	Wednesday April 29 *6:30 PM* — Class 14B *MIDNIGHT* — **DUE Class 14 Learning Journal**	Thursday April 30 *6:30 PM* — Co-working	Friday May 1 *May Day Protests*
Saturday May 2 *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**	Sunday May 3 *MIDNIGHT* — **DUE CCW #1 Completed Personal Pitch** — **DUE CCW #2 Completed Resume** — **DUE Class 14-15 Feedback**	Monday May 4	Tuesday May 5 *6:30 PM* — **DUE Class 15 Lab** — **DUE Class 16 Reading** — Class 16A

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 13 Review

Code Challenge 13 Review

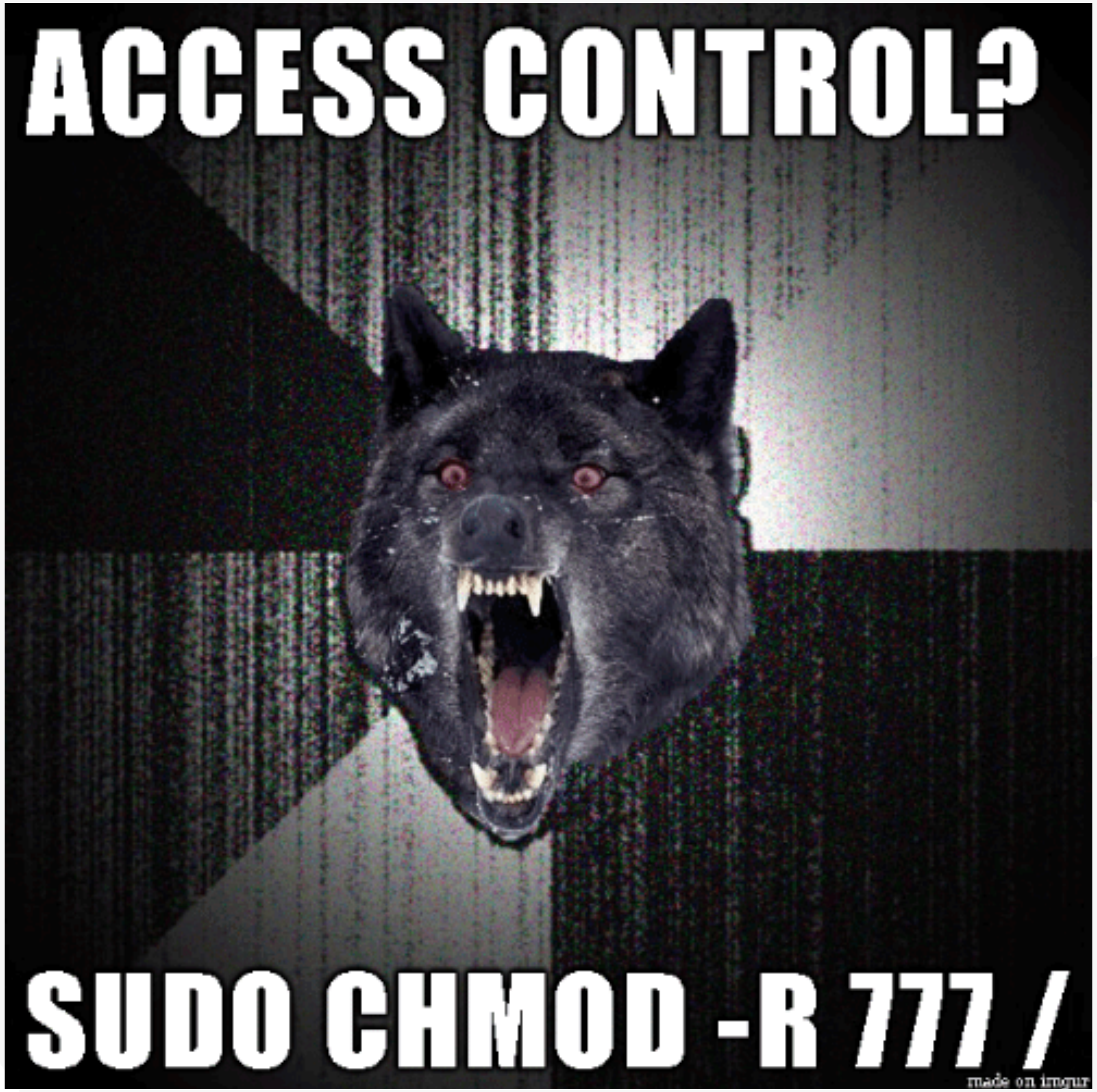
Class 14

Access Control

seattle-javascript-401n16

Access Control

- We have users that are signed in, now what?
- What should these users have access to?
- **Access Control** specifies that every system should have restrictions on data users can see/modify
- An example is file read/write privileges



Types of Access Control

- **Mandatory** - Central authority defines levels of clearance and who fits in what level
- **Discretionary** - Data owners decide who can access on a case-by-case basis
- **Role Based** - Each user has a role, and that role defines access
- **Rule Based** - Each data item has some rules about how and when it can be accessed
- **Attribute Based** - Data and users have attributes, and access is dynamically decided based on those attributes

Role Based Access Control

- This is one of the most common implementations of access control
- Simple to set up, easy to change
- Every user has a **role** field
    - ex: “admin”, “editor”, “user”
- We define the **capabilities** of each role
    - Could be hard-coded
    - Could be defined in a **roles model**

Capabilities

- These are really just what CRUD operations a role can do on certain types of data
- Usually defined as an array of strings
- Capabilities can be as detailed or surface level as the app designer wants. Common capabilities are:
    - Create, Read, Update, Delete

Lab 14 Overview