Your Next Week

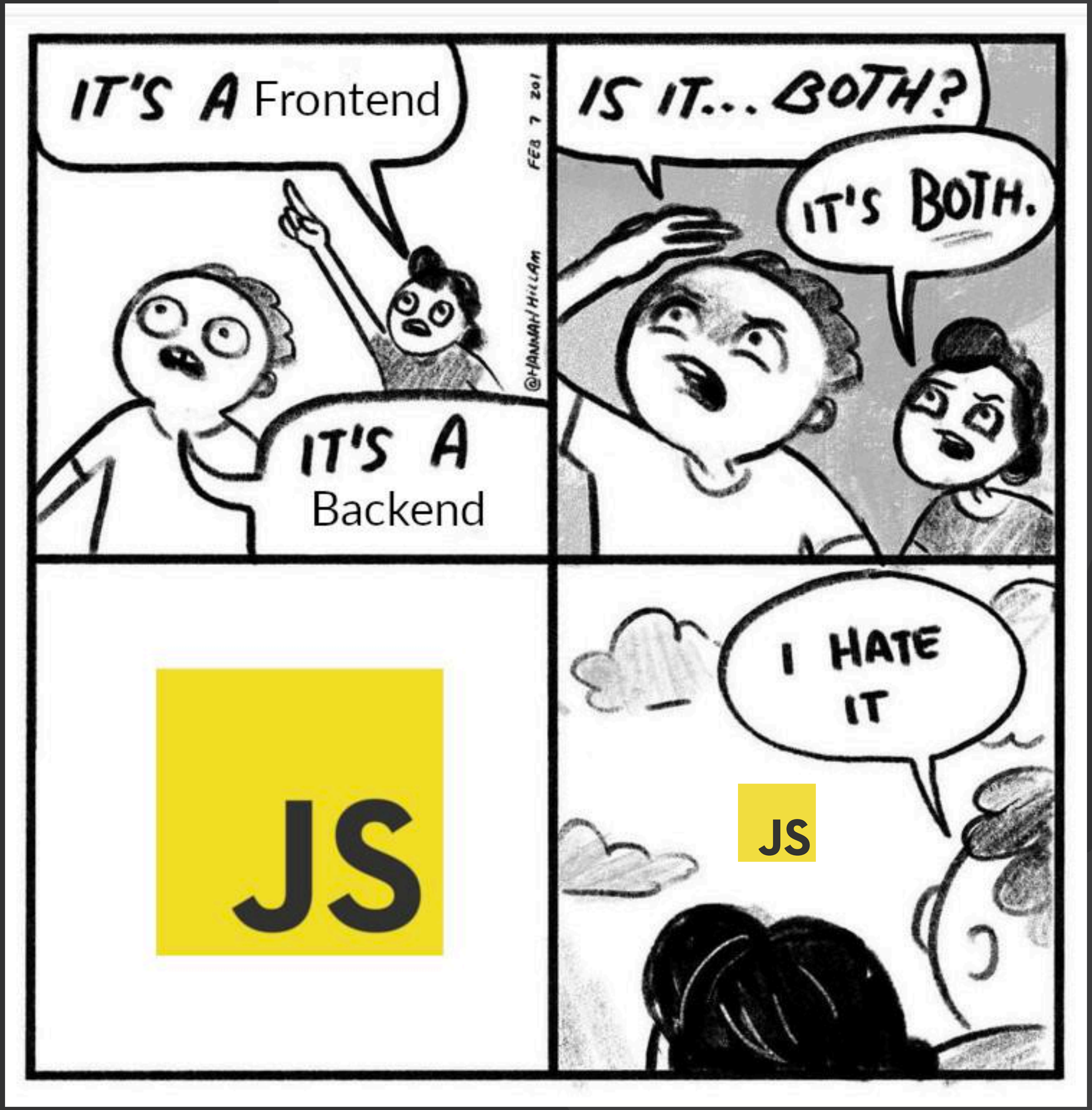
Saturday June 6 *9:00 AM* **— DUE Class 21 Reading** — Class 21 — Interview Prep *MIDNIGHT* **— DUE Class 21 Learning Journal**	Sunday June 7 *MIDNIGHT* **— DUE Midterms - Class 21 Feedback**	Monday June 8 *6:30 PM* — Optional Co-working	Tuesday June 9 *6:30 PM* **— DUE Class 21 Lab** **— DUE Class 21 Code Challenge** **— DUE Class 22 Reading** — Class 22A
Wednesday June 10 *6:30 PM* — Class 22B *MIDNIGHT* **— DUE Class 22 Learning Journal**	Thursday June 11 *6:30 PM* — Co-working	Friday June 12	Saturday June 13 *9:00 AM* **— DUE Class 22 Lab** **— DUE Class 22 Code Challenge** **— DUE Class 23 Reading** — Class 23 — Interview Prep *MIDNIGHT* **— DUE Class 23 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

You survived Midterms!!

Ready for Front End?



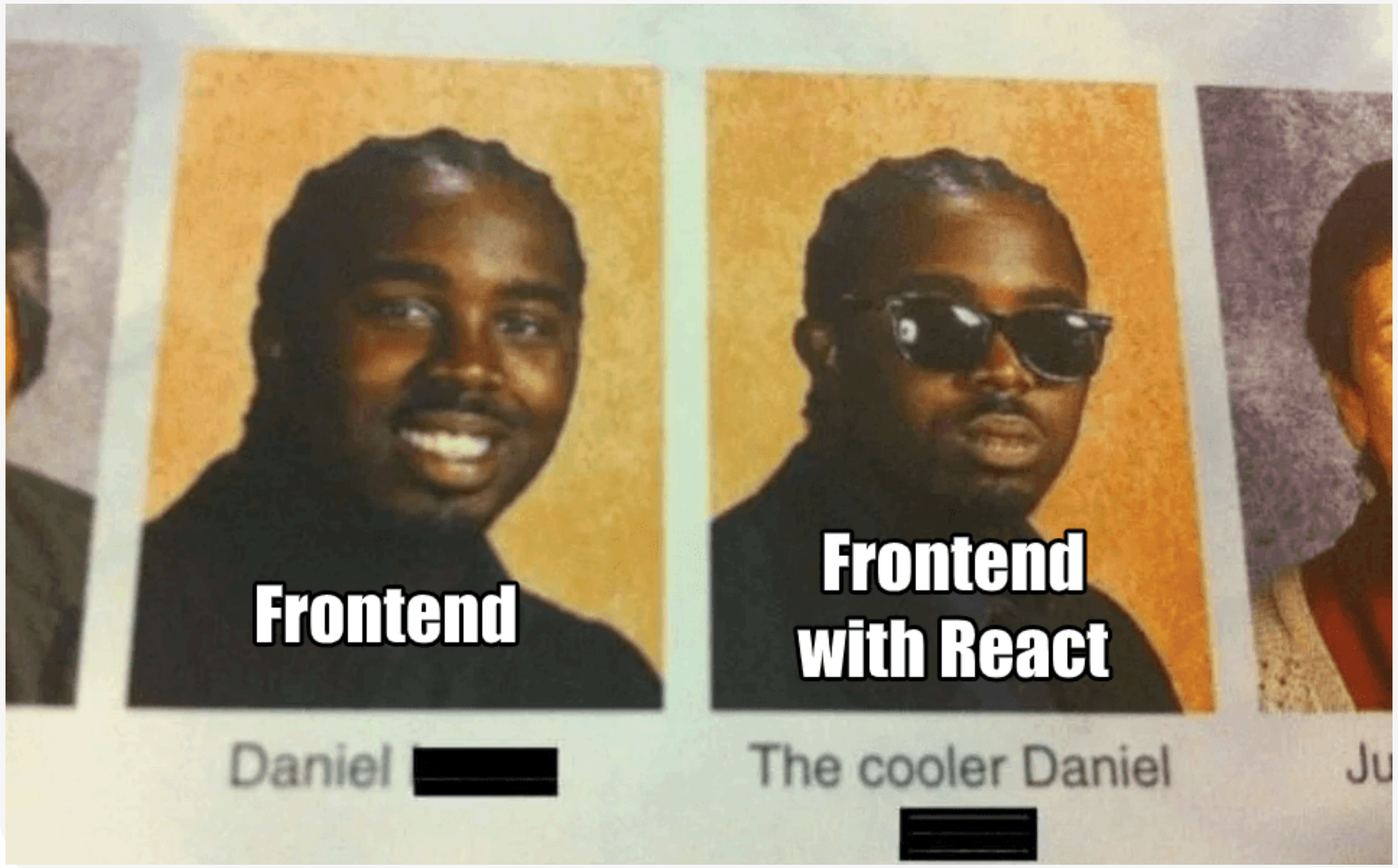
Class 21

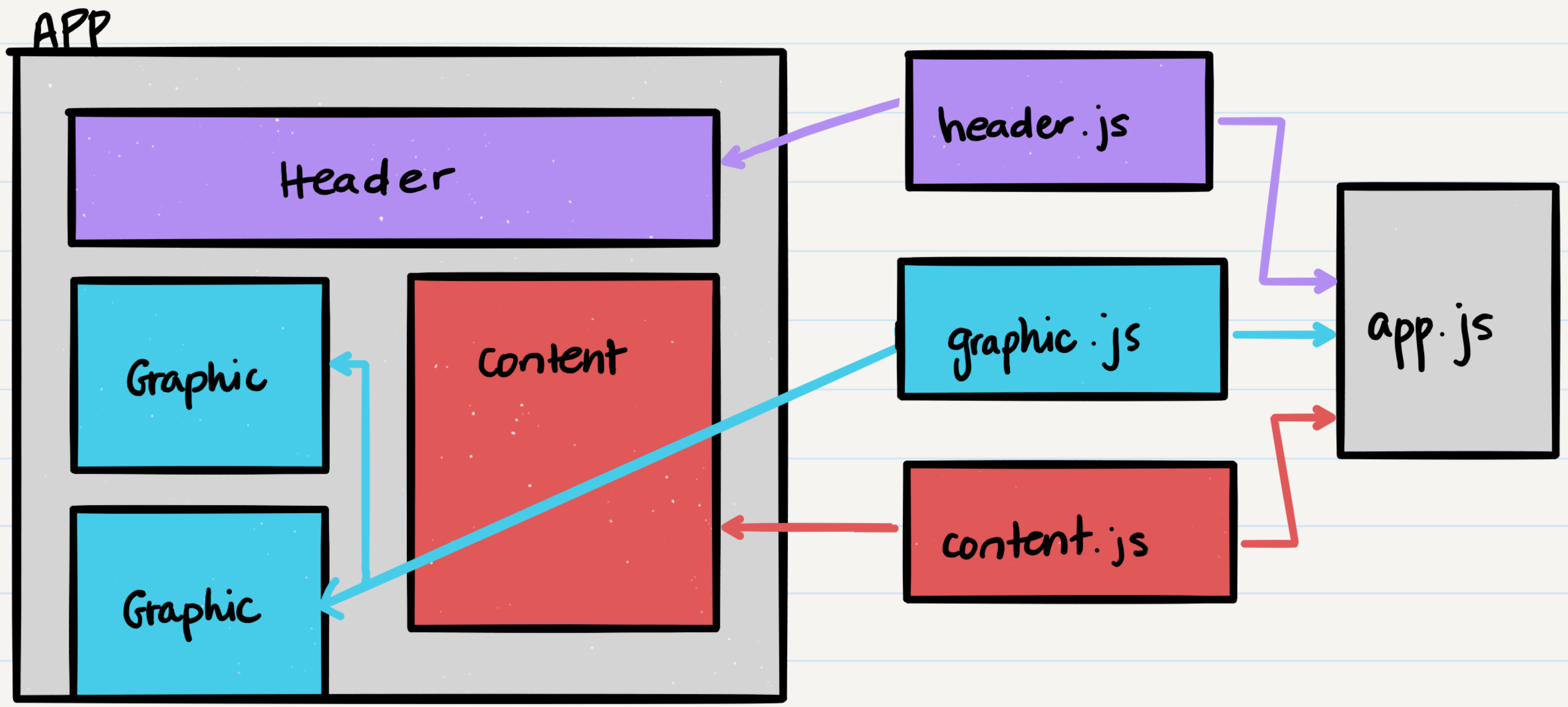
Component Based UI

seattle-javascript-401n16

Front End with React

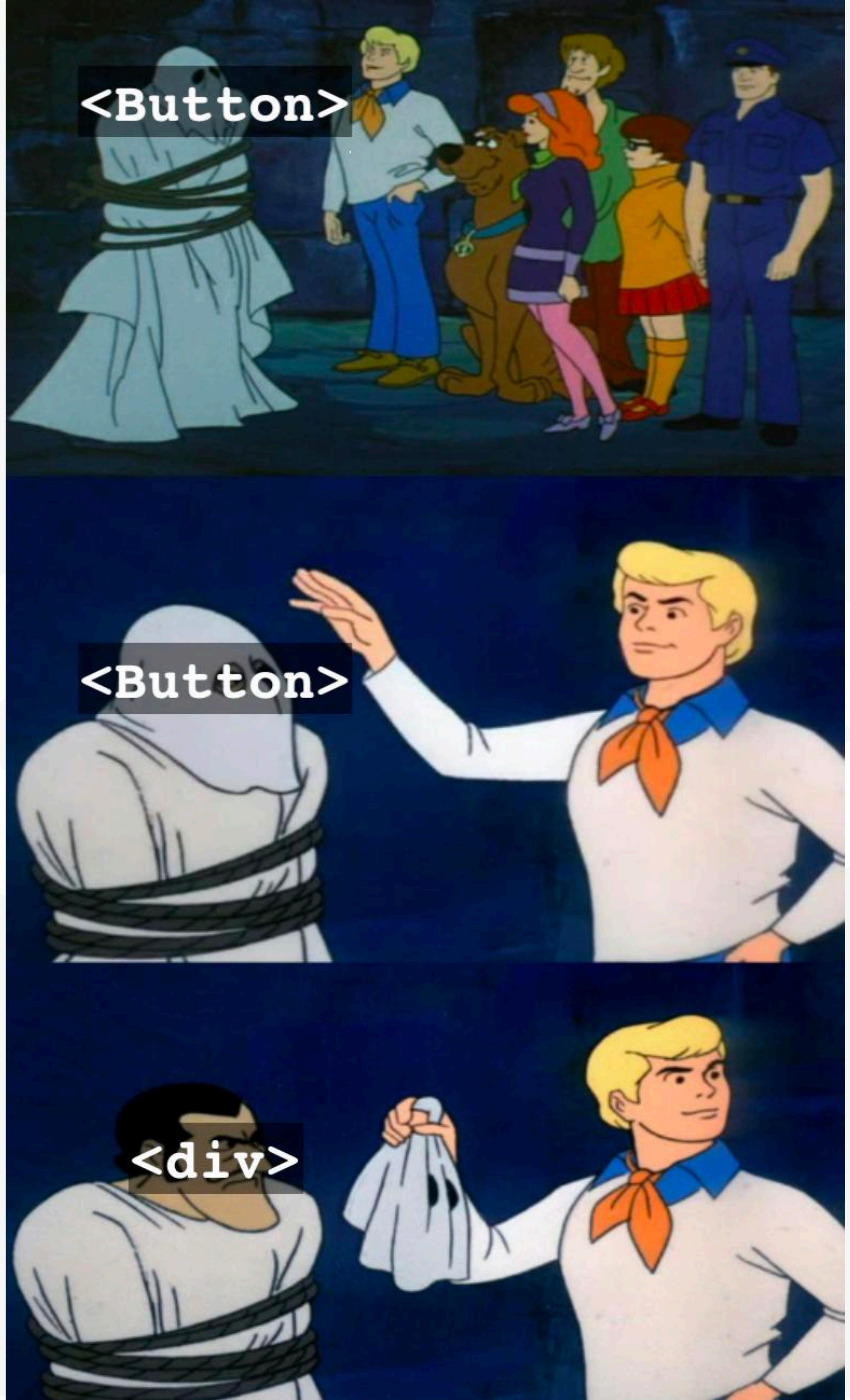
- A JavaScript **library** by Facebook
- Combines JavaScript and HTML into one file (**JSX**)
- Let's us do cool JavaScript type stuff with our HTML
    - Modularize
    - Use variables
    - Dynamic updates




APP
Header
Graphic
Graphic
Content
header.js
graphic.js
content.js
app.js

Components

- Pieces of UI that are self-contained/reusable
- They produce some dynamic HTML
- They can maintain their own **state**
  - A collection of local variables
  - When the state variables change, the HTML is re-generated / **re-rendered**
- A component is basically a dynamic HTML package! HTML + JavaScript logic



State vs Props

- State = **local variables** that cause the HTML to re-render on change
- Props = **parameter variables** that are given to this component from an outside source
    - Props don't effect render unless bound to state
- You can also have static (not stateful) local variables in a component

CSS Isn't Good

- Very restrictive
- Multiple sections of duplicated code
- Hard to change things
- Hard to separate pieces out / modularize
- No way to add code logic
- So…let's use a language that **compiles** into CSS!

CSS Preprocessors (SASS, Less)

- Languages with more code-like features
- **Variables**! Change colors, spacing, fonts, etc across your whole UI effortlessly
- **Mixins**! Generate snippets of style / CSS which you can include anywhere
- **Functions**! Add, subtract, calculate style values
- **Partials**! Modularize your styles
- **Nesting**! Better organization of styles based on nested selectors

Lab 21 Overview