Your Next Week

Tuesday June 30 *6:30 PM* **— DUE Class 27 Lab** **— DUE Class 27 Code Challenge** **— DUE Class 28 Reading** — Class 28A	Wednesday July 1 *6:30 PM* — Class 28B *MIDNIGHT* **— DUE Class 28 Learning Journal**	Thursday July 2 *6:30 PM* — Co-working	Friday July 3
Saturday July 4 **— HOLIDAY: NO CLASS**	Sunday July 5 *MIDNIGHT* **— DUE CCW #2 Job Search, Interviews, Offers** **— DUE CCW #2 Mock Interviews** **— DUE Class 28 Feedback**	Monday July 6	Tuesday July 7 *6:30 PM* **— DUE Class 28 Lab** **— DUE Class 28 Code Challenge** **— DUE Class 29 Reading** — Class 29A

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 27 Review

Code Challenge 27 Review

Class 28

Context API

seattle-javascript-401n16

What is Context?

- What “`this`” refers to
  - The class
  - The function
  - The object
- In the case of React, we usually need “`this`” for referring to “`this.state`” in classes
- React has an actual Context object we can create, which lets us share state in a new way!

The Problem

- The current only way to share state is to pass `Parent -> Child` props
- This can be tedious if there is a long chain of descendants that need that state:

  `A` (provides state) `->`

  `B` (sends state) `->`

  `C` (sends state) `->`

  `D` (consumes state)
- Wouldn't it be nice if you could just have the state sent to all descendants automatically?

  `A` (provides state) `->`

  `B -> C -> D` (consumes state)

Why is this Useful?

- It allows you to create global settings / theme variables
- Any component can access things like local language, theme color, etc without you having to pass down so many props
- Allows for cleaner sharing of stateful data
    - Values passed DON'T have to be stateful though!
- You can have multiple contexts! One for theme colors, one for language, etc

UseContext

```
const value = useContext(myContext);
```

- Here, `value` is going to be whatever you set the Provider’s `value` prop to be (usually `this.state`)
- `myContext` is the `React.CreateContext()` initially created (usually in the same file as the Provider)

Lab 28 Overview