Your Next Week

Tuesday June 23	Wednesday June 24	Thursday June 25	Friday June 26
6:30 PM **— DUE Class 25 Lab** **— DUE Class 26 Reading** — Class 26A	*6:30 PM* — Class 26B *MIDNIGHT* **— DUE Class 26 Learning Journal**	*6:30 PM* — Co-working	
Saturday June 27	**Sunday June 28**	**Monday June 29**	**Tuesday June 30**
9:00 AM **— DUE Class 26 Lab** **— DUE Class 26 Code Challenge** **— DUE Class 27 Reading** — Class 27 — Interview Prep *MIDNIGHT* **— DUE Class 27 Learning Journal**	*MIDNIGHT* **— DUE Class 26-27 Feedback**	*6:30 PM* — CCW #2B	*6:30 PM* **— DUE Class 27 Lab** **— DUE Class 27 Code Challenge** **— DUE Class 28 Reading** — Class 28A

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 25 Review

Class 26

Hooks API

seattle-javascript-401n16

Why Do We Use Classes?

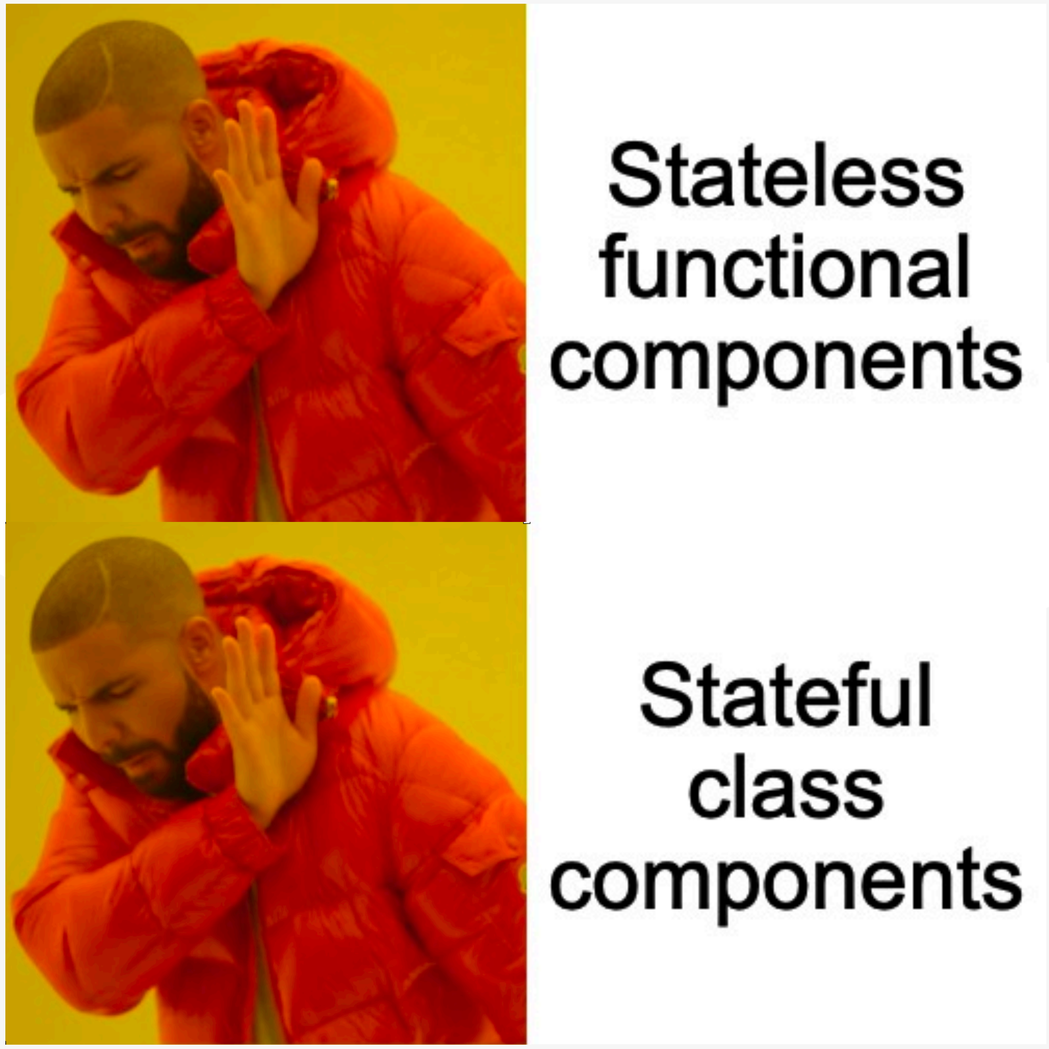
- Lets you manage a **state** to store data or record user input
- Gives you **lifecycle methods**, such as `componentDidMount`
- **Self contained module** that you can plug-and-play in many places

Stateless functional components

Stateful class components

Classes Are Bad Tho…

- Classes can’t easily **share their state** with one another
- **Manually manage context** (make sure the “`this`” reference works)
- Classes can easily **get very large and confusing**



Replacing Classes using Hooks

- Gives you the **best of both worlds**; the power of a class in the simplicity of a function
- **Easily pass states** between functions
- **Cut down on code size** and automate / standardize manual work

UseState

```
const [getter, setter] =
  useState(initValue);
```

- Creates a **getter** and **setter**, and updates the DOM on a value change (just like a state variable!)

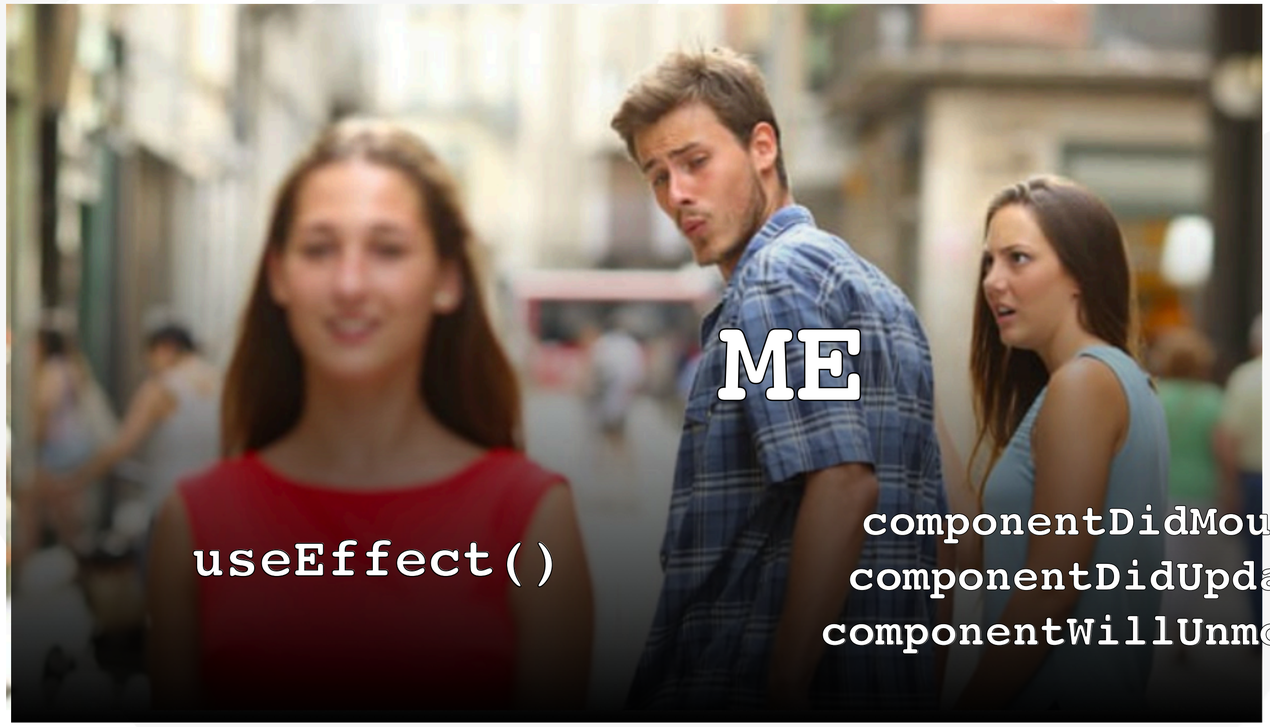
```
1  import React, { useState } from "react";
2
3  export default function MyFunction(props) {
4      const [firstName, setFirstName] = useState("John");
5      const [lastName, setLastName] = useState("Doe");
6
7      function changeFirstName(e) {
8          setFirstName(e.target.value);
9      }
10
11     function changeLastName(e) {
12         setLastName(e.target.value);
13     }
```

UseEffect

```
useEffect( () => {
 //function contents
 return () => { //do on
 unMount }
}, [ optional trigger
var(s) ] );
```

- Executes code during component lifecycle methods.
- Runs on every render by default, but you can customize to run when a trigger variable changes

```
7    useEffect(() => {
8        document.title = firstName + " " + lastName;
9    }, [firstName, lastName]);
10
11   useEffect(() => {
12       window.addEventListener("resize", handleResize);
13
14       return () => {
15           window.removeEventListener("resize", handleResize);
16       };
17   });
```



Rules for Hooks

- Don’t put hooks in any `if` statements, `for` or `while` loops, or any other **conditionals**

  “We love hooks unconditionally”

- Custom hook functions should always start with “`use`”

- Don’t use hooks in classes

Lab 26 Overview