

# **React - The Complete Guide**

**By**

**Maximillian Schwarzmuller**

**Course Link**

<https://www.udemy.com/course/react-the-complete-guide-incl-redux/>

**Notes By**

**Naveen Kumar M**

<https://www.linkedin.com/in/naveen-kumar-m-5a6960119/>

**Source Code Link**

<https://github.com/mnaveensmn/react-projects>

## Table of Contents

<b>Effects, Reducers &amp; Context</b>	<b>8</b>
<b>What is Side Effect ?</b>	<b>9</b>
<b>useEffect() Hook</b>	<b>9</b>
Use Effect Example	10
useEffect and Dependencies	10
What to add & Not to add as Dependencies	11
setTimeout() and clearTimeout() in useEffect()	12
<b>UseReducer() Hook</b>	<b>12</b>
Understanding useReducer()	13
Reducer state in UseEffect() dependency	14
useReducer() vs useState - State Management	15
<b>Context</b>	<b>15</b>
Context Example	16
Step 1: Initializing the Context	16
Step 2: Context Provider	16
Step 3: Context consumer	17
Context consumer using hook	17
Making context dynamic	18
Using centralized state management using Context	18
Context Limitation	20
<b>Rules of Hooks</b>	<b>20</b>
useImperativeHandler Hook	21
<b>Practice Food Order App</b>	<b>21</b>
<b>Adding Cart Button</b>	<b>22</b>
<b>Adding a meals component</b>	<b>23</b>
<b>Adding a Form</b>	<b>24</b>
<b>Adding a Model via React Portal</b>	<b>26</b>
<b>Adding Cart Close Functionality</b>	<b>26</b>
<b>Adding Cart Context</b>	<b>27</b>
<b>Using Cart Context</b>	<b>27</b>
<b>Using Cart Reducer</b>	<b>28</b>
<b>Using and forwarding Ref's</b>	<b>28</b>
<b>Reducer logic for adding and removing items from cart</b>	<b>30</b>
<b>Implementing Animation to Cart Button</b>	<b>30</b>
<b>Video Demo</b>	<b>31</b>
<b>How React Works?</b>	<b>32</b>
<b>Understanding</b>	<b>32</b>
<b>Demo - Inserting the element without refreshing the UI</b>	<b>34</b>
<b>Child component re evaluation</b>	<b>34</b>
<b>Preventing Function Re-Creation using useCallback()</b>	<b>35</b>
<b>Component and state</b>	<b>36</b>
<b>State update and Scheduling</b>	<b>37</b>
<b>useMemo() hook</b>	<b>37</b>
<b>Class based Components</b>	<b>38</b>

<b>Class based components alternative to Functions</b>	38
First class-based component	39
Managing state in class-based component	40
Component Lifecycle – using useEffect() functionality in Class based component	41
componentDidUpdate()	41
componentDidMount()	42
Using context in class-based component	42
Class Based Component vs Functional component	43
Error Boundary	44
<b>HTTP Request</b>	45
Browser-side Apps Don't Directly Talk to Databases	45
Using fetch and then functions	46
Using Async Await	46
Handling the Loading with States	46
Handling error using try catch	48
Using UseEffect for sending Http Request	48
Http POST request	49
<b>Building Custom Hooks</b>	49
What are Custom Hooks ?	50
Creating Custom Hook	51
Configuring the custom hooks	51
Custom Hook to handling Http Requests	52
<b>Working with Form and User Input</b>	53
Dealing with Form Submission and Getting input values	55
Doing the validation on submission	56
Disabling the button based on the state	57
Refactoring the functionality	58
Custom Hook	59
Using reducer in custom hook	60
<b>Using Http in Food Order APP</b>	61
Reusing the useHttp hook	61
Demo	61
<b>Redux</b>	62
What is Cross-Component / App-Wide State?	63
Then Why do need a Redux when we already context Provider	63
How Redux Works?	65
Redux Core Concept	66
Redux Demo	66
Required redux package for react project	67

<b>Registering the redux store in react app</b>	<b>67</b>
<b>Use the redux store in react component</b>	<b>68</b>
<b>Use the redux store in react class based component</b>	<b>69</b>
<b>Counter App Demo</b>	<b>70</b>
<b>Passing data via dispatch action</b>	<b>70</b>
<b>Working with multiple state in redux</b>	<b>71</b>
<b>Redux Challenges &amp; Introducing Redux Toolkit</b>	<b>74</b>
<b>Redux Toolkit Example</b>	<b>74</b>
<b>Using Multiple States</b>	<b>77</b>
<b><i>Advanced Redux Concept</i></b>	<b>79</b>
<b>Using asynchronous code with Redux</b>	<b>80</b>
<b>React Dev Tool</b>	<b>84</b>
<b><i>React Router</i></b>	<b>84</b>
<b>Why routing?</b>	<b>85</b>
<b>Router Basic</b>	<b>86</b>
<b>Using Link to navigate to Route</b>	<b>87</b>
<b>Using NavLink to navigate to Route</b>	<b>87</b>
<b>Route Param</b>	<b>88</b>
<b>Using “Switch” and “exact” for configuring routes</b>	<b>89</b>
<b>Working with Nested Routes</b>	<b>91</b>
<b>Using Redirect</b>	<b>91</b>
<b>Route for handling unknown URL paths</b>	<b>92</b>
<b>Programmatic Navigation</b>	<b>93</b>
<b>Quote App Implementation So Far</b>	<b>93</b>
<b>Prompting User on Navigation</b>	<b>94</b>
<b>Working with Query Parameters</b>	<b>95</b>
<b>Removing the component based on URL</b>	<b>96</b>
<b>Writing more flexible route code</b>	<b>97</b>
<b><i>React Router 6</i></b>	<b>98</b>
Switch changed to Routes	98
Exact has been removed	98
element is introduced in Route tag.	98
Order of the Routes does not matter	98
NavLink activeClassName has been removed	99
Redirect changed to Navigate	99
While adding the nested routes, no need to specify the parent path	100
Alternate way specifying the alternate route	100
useHistory changed to useNavigate	101
Prompt is removed.	101
<b><i>Deploying React App</i></b>	<b>101</b>
<b>Deployment Step</b>	<b>101</b>
<b>Deployment Optimization</b>	<b>102</b>

<b>Building the code for production</b>	<b>103</b>
Upload Production Code to Server	103
<b>Adding Authentication to React App</b>	<b>107</b>
Adding User Signup	109
Adding User Sign In	112
Redirecting the login page after the password change	114
Protecting Frontend Page	114
Storing login token in storage browser	115
Auto logout	115
Handling the timer while logout	116
<b>Next JS</b>	<b>117</b>
Why Next JS?	117
Next JS Key Feature and Benefits	118
Create Next JS Project	118
Nested Pages	119
Dynamic Page	120
Extracting a Dynamic Parameter Value	121
Linking between Pages	121
Other way – using Link from next js	122
Preparing project pages	123
Adding a new meetup form	125
Adding a Navigations	126
Adding a navigation bar to entire pages	127
Meetup App - Implementation so far	128
Data Fetching for static pages	128
Static Site Generation (SSG)	129
Regenerate the code	130
Server-Side Rendering (SSR)	131
getStaticPaths()	131
Working with MongoDB	132
Inserting data to MongoDB	133
Loading Data from MongoDB	135
Loading a Dynamic Data in MeetupDetails page	135
Adding a Meta Data to Page before Deployment	136
<b>Animation</b>	<b>137</b>
Preparing the project for adding animation	137
Animating the model pop up	137
CSS Animation Property	139
Using ReactTransition group	140

<b>Adding Transition to Model Popup</b>	<b>142</b>
<b>Wrapping the Transition Component</b>	<b>143</b>
<b>Animation Timings</b>	<b>144</b>
<b>Transition Events</b>	<b>145</b>
<b>CSSTransition component</b>	<b>147</b>
<b>Using a classnames in CSSTransition component</b>	<b>148</b>
<b>Using TransitionGroup to manage multiple transitions</b>	<b>148</b>
<b>Other React Animation Library</b>	<b>149</b>
<b><i>Replacing Redux with React Hooks</i></b>	<b>149</b>
<b>Testing React App</b>	<b>152</b>
<b>What is Testing?</b>	<b>152</b>
<b>Different Kind of Test</b>	<b>153</b>
<b>What to test ?</b>	<b>153</b>
<b>Running the test</b>	<b>154</b>
<b>Writing a test</b>	<b>155</b>
<b>Grouping test together with Test Suites</b>	<b>155</b>
<b>Testing connected component</b>	<b>159</b>
<b>Testing Asynchronous Code</b>	<b>160</b>
<b>Working with Mock</b>	<b>161</b>
<b><i>React + Typescript</i></b>	<b>161</b>
<b>What and Why?</b>	<b>162</b>
<b>Installing Typescript</b>	<b>163</b>
<b>npm init -y</b>	<b>163</b>
<b>npm install typescript</b>	<b>164</b>
<b>Compiling type script to JavaScript</b>	<b>164</b>
<b>Base Types</b>	<b>165</b>
<b>Array and Objects</b>	<b>165</b>
<b>Type Inference</b>	<b>166</b>
<b>Union Type</b>	<b>166</b>
<b>Type alias</b>	<b>166</b>
<b>Function and Function Types</b>	<b>167</b>
<b>Generics</b>	<b>167</b>
<b><i>Working with React TypeScript Project</i></b>	<b>168</b>
<b>Creating a project</b>	<b>168</b>
<b>Working with component and TypeScript</b>	<b>169</b>
<b>Working with props and Typescript</b>	<b>170</b>
<b>Adding a Data Model</b>	<b>172</b>
<b>Sperate out by Adding a TodoItem component</b>	<b>173</b>
<b>Working with ref, useRef and Function Props</b>	<b>174</b>
<b>Managing State</b>	<b>176</b>

<b>Removing Todo Item</b>	<b>177</b>
<b>Using Context API</b>	<b>180</b>
<b>tsconfig.json</b>	<b>182</b>
<b><i>React Hooks</i></b>	<b>184</b>
<b>What are React Hooks ?</b>	<b>184</b>
<b>useState Hook</b>	<b>185</b>
<b>SetStateFunction with Previous state</b>	<b>186</b>
<b>Rules of Hooks</b>	<b>187</b>
<b>Performance Optimization</b>	<b>189</b>
<b><i>React Summary</i></b>	<b>190</b>
<b><i>What is React ?</i></b>	<b>190</b>
<b>Reactive Alternative</b>	<b>191</b>
<b>Creating a React project</b>	<b>192</b>
<b><i>React Eco System</i></b>	<b>192</b>

## Effects, Reducers & Context

## Effects, Reducers & Context

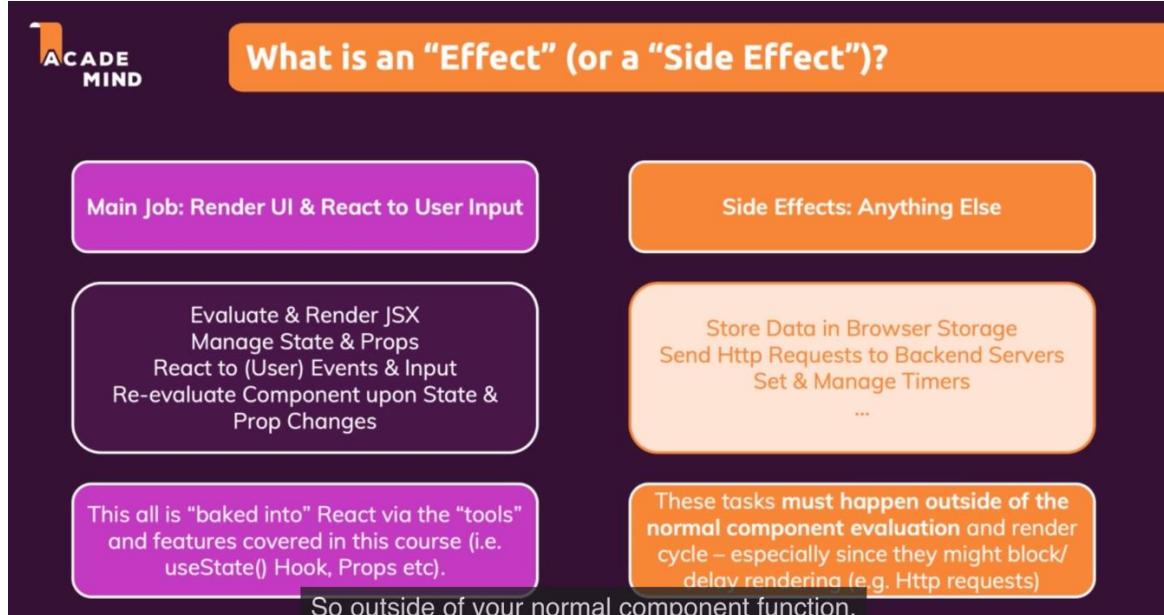
Advanced, yet super-important Features!

ACADE MIND

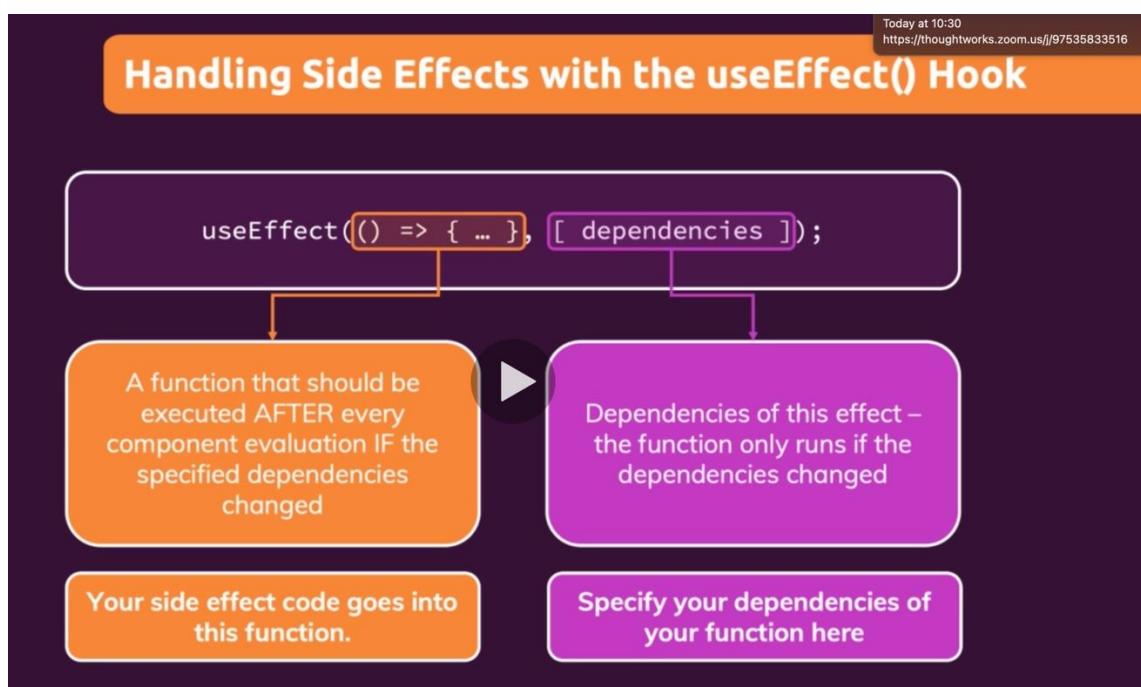
### Module Content

- Working with (Side) Effects
- Managing more Complex State with Reducers
- Managing App-Wide or Component-Wide State with Context

## What is Side Effect ?



## useEffect() Hook



```

Complexity is 13 You must be kidding
function App() { █
  const [isLoggedIn, setIsLoggedIn] = useState(false);

  useEffect(() => {
    const isUserLoggedIn = localStorage.getItem("isLoggedIn");
  }) █

  if (isUserLoggedIn === "1") {
    setIsLoggedIn(true);
  }
}, [];

const loginHandler = (email, password) => {
  // We should of course check email and password
  // But it's just a dummy/ demo anyways
  localStorage.setItem("isLoggedIn", "1");
  setIsLoggedIn(true);
};

const logoutHandler = () => {
  setIsLoggedIn(false);
  localStorage.removeItem("isLoggedIn")
};

```

localStorage is built-in storage in browser.

## Use Effect Example

```

Complexity is 13 You must be kidding
function App() { █
  const [isLoggedIn, setIsLoggedIn] = useState(false);

  useEffect(() => {
    const isUserLoggedIn = localStorage.getItem("isLoggedIn");
  }) █

  if (isUserLoggedIn === "1") {
    setIsLoggedIn(true);
  }
}, [];

const loginHandler = (email, password) => {
  // We should of course check email and password
  // But it's just a dummy/ demo anyways
  localStorage.setItem("isLoggedIn", "1");
  setIsLoggedIn(true);
};

const logoutHandler = () => {

```

This useEffect() function runs only once on starting the app, since we didn't specify the dependency.

## UseEffect and Dependencies

If we remove the second argument (without an array of dependencies) from the useEffect function, useEffect is executed every time component is reevaluated. It causes the infinite loop.

```
useEffect(() => {
  setFormIsValid(
    enteredEmail.includes('@') && enteredPassword.trim().length > 6
  );
});
```

We can have a function and state variable in useEffect dependency. Whenever dependencies changes useEffect function will be executed.

```
useEffect(() => {
  setFormIsValid(
    enteredEmail.includes('@') && enteredPassword.trim().length > 6
  );
}, [setFormIsValid, enteredEmail, enteredPassword]);
```

## What to add & Not to add as Dependencies

- You **DON'T need to add state updating functions** (as we did in the last lecture with `setFormIsValid`): React guarantees that those functions never change, hence you don't need to add them as dependencies (you could though)
- You also **DON'T need to add "built-in" APIs or functions** like `fetch()`, `localStorage` etc (functions and features built-into the browser and hence available globally): These browser APIs / global functions are not related to the React component render cycle and they also never change
- You also **DON'T need to add variables or functions** you might've **defined OUTSIDE of your components** (e.g. if you create a new helper function in a separate file): Such functions or variables also are not created inside of a component function and hence changing them won't affect your components (components won't be re-evaluated if such variables or functions change and vice-versa)

So long story short: You must add all "things" you use in your effect function **if those "things" could change because your component (or some parent component) re-rendered**. That's why variables or state defined in component functions, props or functions defined in component functions have to be added as dependencies!

```

1 import { useEffect, useState } from 'react';
2
3 let myTimer;
4
5 const MyComponent = (props) => {
6   const [timerIsActive, setTimerIsActive] = useState(false);
7
8   const { timerDuration } = props; // using destructuring to
9
10  useEffect(() => {
11    if (!timerIsActive) {
12      setTimerIsActive(true);
13      myTimer = setTimeout(() => {
14        setTimerIsActive(false);
15      }, timerDuration);
16    }
17  }, [timerIsActive, timerDuration]);
18}

```

- `timerIsActive` is added as a dependency because it's component state that may change when the component changes (e.g. because the state was updated)
- `timerDuration` is added as a dependency because it's a prop value of that component - so it may change if a parent component changes that value (causing this MyComponent component to re-render as well)
- `setTimerIsActive` is NOT added as a dependency because it's that exception: State updating functions could be added but don't have to be added since React guarantees that the functions themselves never change
- `myTimer` is NOT added as a dependency because it's not a component-internal variable (i.e. not some state or a prop value) - it's defined outside of the component and changing it (no matter where) wouldn't cause the component to be re-evaluated
- `setTimeout` is NOT added as a dependency because it's a built-in API (built-into the browser) - it's independent from React and your components, it doesn't change

## setTimeout() and clearTimeout() in useEffect()

```

Complexity is 5 Everything is cool!
useEffect(()=> {
  const identifier = setTimeout(()=> [
    console.log("Checking for input validity");
    setFormIsValid(
      enteredEmail.includes('@') && enteredPassword.trim().length > 6
    );
  ], 500);

  return ()=> {
    console.log("Clear timeout")
    clearTimeout(identifier);
  }
}, [enteredEmail,enteredPassword]);

```

- `setTimeout()` – Based on the dependency, useEffect function will execute. Every time dependencies value changes, useEffect function trigger immediately after the component function render. So, every time key typed useEffect will get triggered. This will end make the multiple call to this function. To overcome this problem, we can use the `setTimeout()` function to make some delay the execution upon dependency changes. New timer will set on every time dependency changes.
- return statement inside the `useEffect()` – UseEffect function can return the function that will take care of the clean up activity. This return function will run at start of the useEffect function except the first time. In this example we are clearing the timer using `clearTimeout()` function.
- This clean up function also execute when the component is removed from the dom.

## UseReducer() Hook

- State update depends on other state.
- Two more state belonging together.

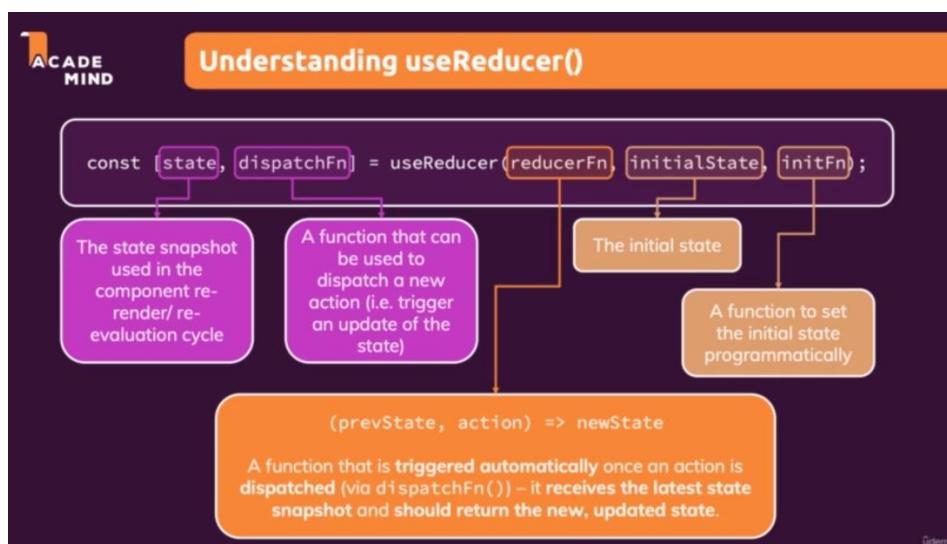
## Introducing useReducer() for State Management

Sometimes, you have **more complex state** – for example if it got **multiple states, multiple ways of changing it or dependencies** to other states

useState() then often **becomes hard or error-prone to use** – it's easy to write bad, inefficient or buggy code in such scenarios

useReducer() can be used as a **replacement** for useState() if you need "**more powerful state management**"

### Understanding useReducer()



```
const [email, dispatchEmail] = useReducer(emailReducer, {  
  value: '',  
  isValid: false  
});
```

Complexity is 6 It's time to do something...

```
const emailReducer = (state, action) =>{  
  if (action.type === 'USER_INPUT') {  
    return {value: action.value, isValid: action.value.includes('@')}  
  } if (action.type === 'INPUT_BLUR') {  
    return {value: state.value, isValid: false}  
  }  
  return {value: '', isValid: false}  
}
```

```
const emailChangeHandler = (event) => {
  dispatchEmail({type: 'USER_INPUT', val: event.target.value});
  setFormIsValid(
    event.target.value.includes('@') && enteredPassword.trim().length > 6
  );
};
```

```
const validateEmailHandler = () => {
  dispatchEmail({type: 'INPUT_BLUR'});
};
```

Reducer state in useEffect() dependency

```
const [email, dispatchEmail] = useReducer(emailReducer, {
  value: '',
  isValid: false
});
const [password, dispatchPassword] = useReducer(passwordReducer, {
  value: '',
  isValid: false
});
```

```
const {isValid:emailIsValid} = email;
const {isValid:passwordIsValid} = password;
```

Complexity is 5 Everything is cool!

```
useEffect(() => {
  const identifier = setTimeout(() => {
    console.log('Checking form validity!');
    setFormIsValid(
      passwordIsValid && passwordIsValid
    );
  }, 500);

  return () => {
    console.log('CLEANUP');
    clearTimeout(identifier);
  };
}, [emailIsValid, passwordIsValid]);
```

Here we are using the reducer's object property on useEffect dependency. We pass specific properties instead of the entire object as a dependency.

We could also write this code and it would **work in the same way**.

```
1 | useEffect(() => {  
2 |   // code that only uses someProperty ...  
3 | }, [someObject.someProperty]);
```

This works just fine as well!

But you should **avoid** this code:

```
1 | useEffect(() => {  
2 |   // code that only uses someProperty ...  
3 | }, [someObject]);
```

Why?

Because now the **effect function would re-run whenever**

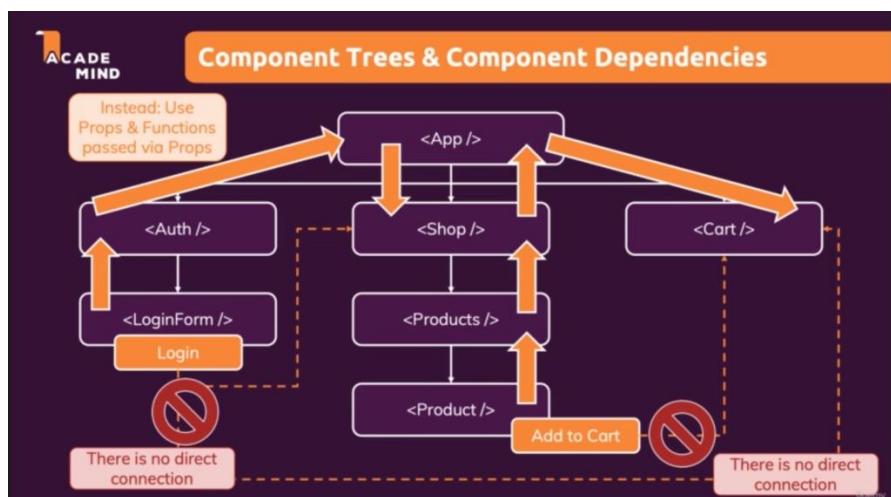
**ANY property of `someObject`** changes - not just the one property (`someProperty` in the above example) our effect might depend on.

useReducer() vs useState - State Management

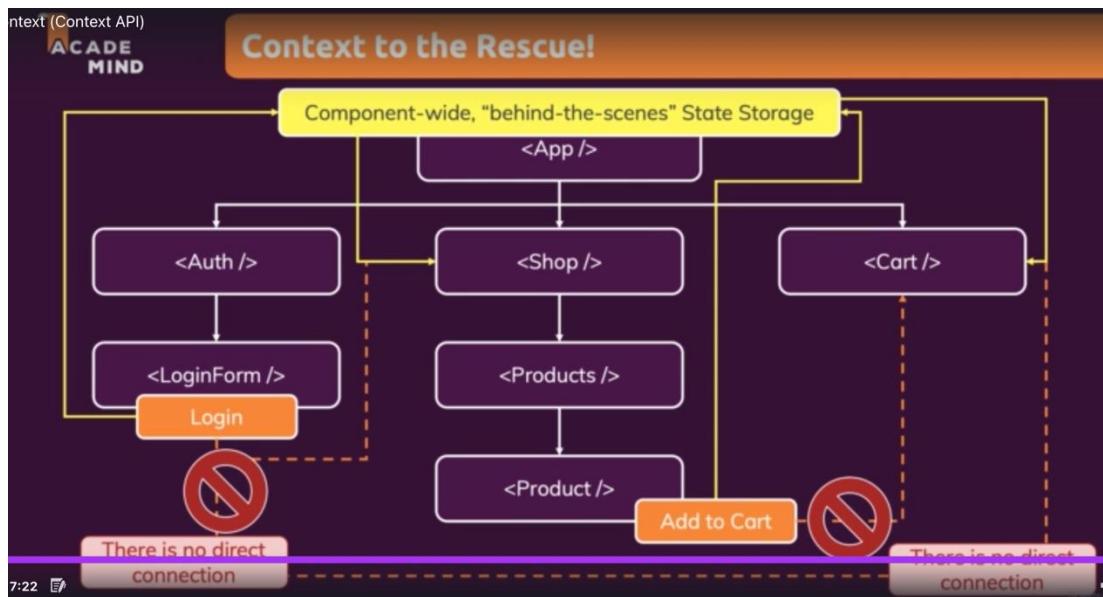


Context

When we have larger project lifting the state up and down can be tedious.



In this case we can use “Context”.



## Context Example

### Step 1: Initializing the Context

```
import React from 'react';
const AuthContext = React.createContext({
  isLoggedIn: false
});

export default AuthContext;
```

Here, AuthContext contains the component objects.

### Step 2: Context Provider

```
return (
  <AuthContext.Provider
    value={{
      isLoggedIn: isLoggedIn
    }}
  >
    <MainHeader onLogout={logoutHandler} />
    <main>
      {!isLoggedIn && <Login onLogin={loginHandler} />}
      {isLoggedIn && <Home onLogout={logoutHandler} />}
    </main>
  </AuthContext.Provider>
);
```

### Step 3: Context consumer

```
const Navigation = (props) => {  
  return (  
    <AuthContext.Consumer>  
      Complexity is 13 You must be kidding  
      {(ctx) => {  
        return (<nav className={classes.nav}>  
          <ul>  
            {ctx.isLoggedIn && (  
              <li>  
                <a href="/">Users</a>  
              </li>  
            )}  
            {ctx.isLoggedIn && (  
              <li>  
                <a href="/">Admin</a>  
              </li>  
            )}  
            {ctx.isLoggedIn && (  
              <li>  
                <button onClick={props.onLogout}>Logout</button>  
              </li>  
            )}  
          </ul>  
        </nav>  
      })  
    </AuthContext.Consumer>  
  );  
};
```

### Context consumer using hook

```
import React, {useContext} from 'react';  
import AuthContext from '../store/auth-context';  
  
import classes from './Navigation.module.css';  
  
Complexity is 13 You must be kidding  
const Navigation = (props) => {  
  const ctx = useContext(AuthContext);  
  return (  
    <nav className={classes.nav}>  
      <ul>  
        {ctx.isLoggedIn && (  
          <li>  
            <a href="/">Users</a>  
          </li>  
        )}  
        {ctx.isLoggedIn && (  
          <li>  
            <a href="/">Admin</a>  
          </li>  
        )}  
        {ctx.isLoggedIn && (  
          <li>  
            <button onClick={props.onLogout}>Logout</button>  
          </li>  
        )}  
      </ul>  
    </nav>  
  );  
};  
  
export default Navigation;
```

When we are forwarding prop values instead of using it, we can use context there.

Context use to manage the app wide state.

## Making context dynamic

Using the function as context.

```
return (
  <AuthContext.Provider
    value={{
      isLoggedIn: isLoggedIn,
      onLogout : logoutHandler
    }}
  >
    <MainHeader/>
    <main>
      {!isLoggedIn && <Login onLogin={loginHandler} />
      {isLoggedIn && <Home onLogout={logoutHandler} />}
    </main>
  </AuthContext.Provider>
);
}

const ctx = useContext(AuthContext);
return (
  <nav className={classes.nav}>
    <ul>
      {ctx.isLoggedIn && (
        <li>
          <a href="/">Users</a>
        </li>
      )}
      {ctx.isLoggedIn && (
        <li>
          <a href="/">Admin</a>
        </li>
      )}
      {ctx.isLoggedIn && (
        <li>
          <button onClick={ctx.onLogout}>Logout</button>
        </li>
      )}
    </ul>
  </nav>
)
```

Using centralized state management using Context

```
ReactDOM.render(
  <AuthContextProvider>
    <App />
  </AuthContextProvider>,
  document.getElementById("root")
);
```

```
import React, { useState, useEffect } from "react";

const AuthContext = React.createContext({
  isLoggedIn: false,
  onLogout: () => {},
  onLogin: (email, password) => {},
});

Complexity is 7 It's time to do something...
export const AuthContextProvider = (props) => {
  const [isLoggedIn, setIsLoggedIn] = useState(false);

  const logoutHandler = () => {
    localStorage.removeItem("isLoggedIn");
    setIsLoggedIn(false);
  };

  const loginHandler = () => {
    localStorage.setItem("isLoggedIn", "1");
    setIsLoggedIn(true);
  };

  useEffect(() => {
    const storedUserLoggedInInformation = localStorage.getItem(""

      if (storedUserLoggedInInformation === "1") {
        setIsLoggedIn(true);
      }
    }, []);

    return (
      <AuthContext.Provider
        value={{

          isLoggedIn: isLoggedIn,
          onLogout: logoutHandler,
          onLogin: loginHandler,
        }}
      >
        {props.children}
      </AuthContext.Provider>
    );
  };
};
```

```
Complexity is 9 It's time to do something...
function App() {
  const ctx = useContext(AuthContext)
  return (
    <React.Fragment>
      <MainHeader />
      <main>
        {!ctx.isLoggedIn && <Login />}
        {ctx.isLoggedIn && <Home />}
      </main>
    </React.Fragment>
  );
}
```

## Context Limitations

React Context is **NOT optimized** for high frequency changes!



We'll explore a better tool for that, later

React Context also **shouldn't be used to replace ALL component communications and props**



Component should still be configurable via props and short "prop chains" might not need any replacement

## Rules of Hooks

### Rules of Hooks

Only call React Hooks in **React Functions**

React Component Functions

Custom Hooks  
(covered later!)

Only call React Hooks at the **Top Level**

Don't call them  
in nested  
functions

Don't call them  
in any block  
statements

+ extra, unofficial Rule for **useEffect()**: **ALWAYS** add everything you refer to inside of useEffect() as a dependency!

## useImperativeHandler Hook

```

Complexity is 5 Everything is cool!
91 const submitHandler = (event) => {
92   event.preventDefault();
93   if (formIsValid) {
94     authCtx.onLogin(email.value, password.value);
95   } else if (!emailIsValid) {
96     emailInputRef.current.focus();
97   } else {
98     passwordInputRef.current.focus();
99   }
100 };
101
102 return (
103   <Card className={classes.login}>
104     <form onSubmit={submitHandler}>
105       <Input
106         ref={emailInputRef}
107         id="email"
108         label="E-Mail"
109         type="email"
110         isValid={emailIsValid}
111         value={email.value}
112         onChange={emailChangeHandler}
113         onBlur={validateEmailHandler}
114       />
115       <Input
116         ref={passwordInputRef}
117         id="password"
118         label="Password"
119         type="password"
120         isValid={passwordIsValid}
121         value={password.value}
122         onChange={passwordChangeHandler}
123         onBlur={validatePasswordHandler}
124       />
125     </form>
126   </Card>
127 );
128
129 
```

```

Complexity is 9 It's time to do something...
const Input = React.forwardRef((props, ref) => {
  const inputRef = useRef();
  const activate = () => {
    inputRef.current.focus();
  }
  useImperativeHandle(ref, () => {
    return {
      focus: activate,
    };
  });
  return (
    <div
      className={`${classes.control} ${props.isValid === false ? classes.invalid : ''}`}
    >
      <label htmlFor="email">{props.label}</label>
      <input
        ref={inputRef}
        type={props.type}
        id={props.id}
        value={props.value}
        onChange={props.onChange}
        onBlur={props.onBlur}
      />
    </div>
  );
};

export default Input;

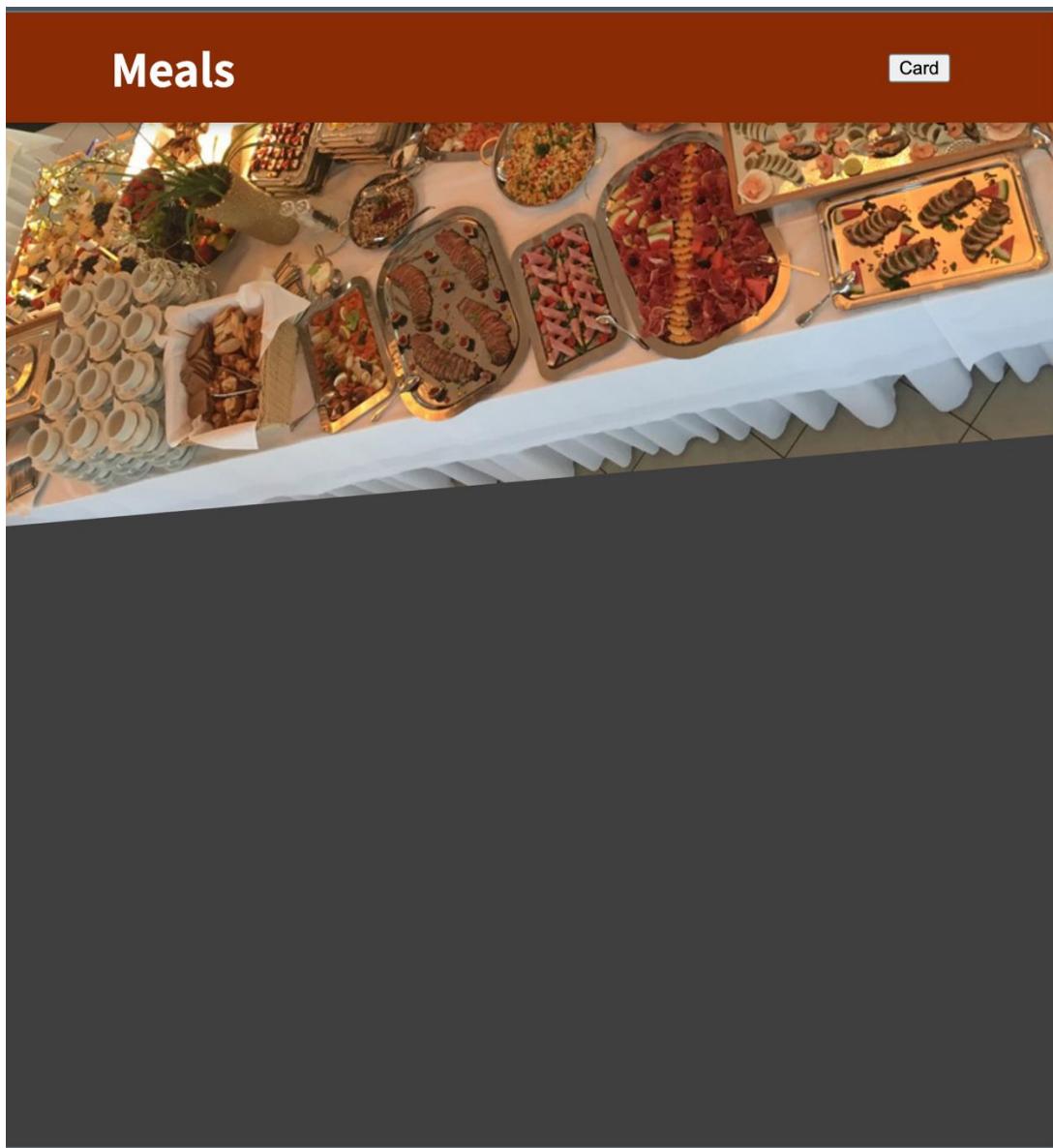
```

## Practice Food Order App

07-FOOD-ORDER-APP

```

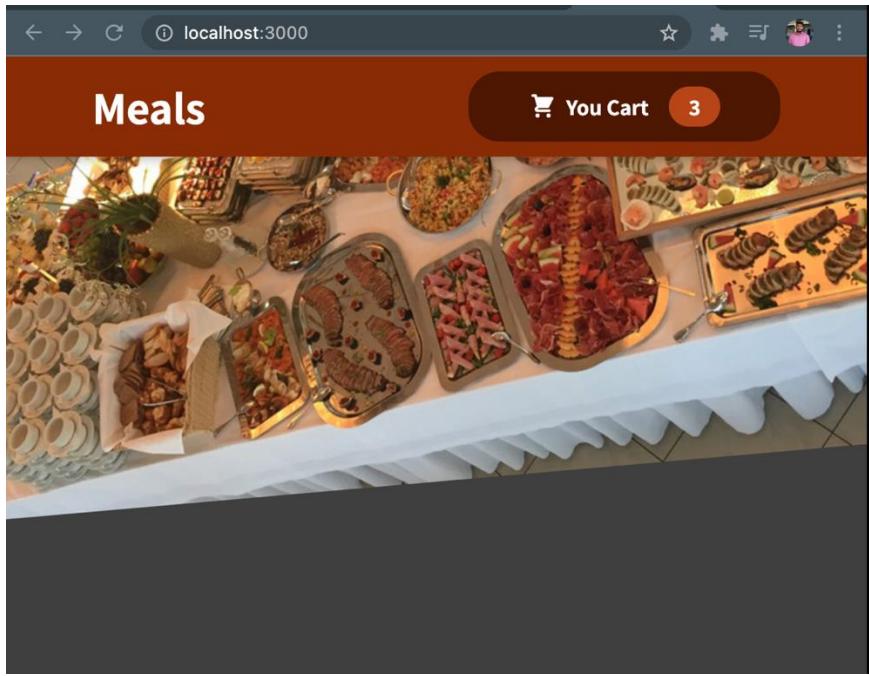
src > components > Layout > Header.js > Header
1 import { Fragment } from "react";
2 import mealsImage from "../../assets/meals.jpeg";
3 import classes from './Header.module.css';
4
5 Complexity is 8 It's time to do something...
6 const Header = (props) => {
7   return (
8     <Fragment>
9       <header className={classes.header}>
10         <h1>Meals</h1>
11         <button>Card</button>
12       </header>
13       <div className={classes['main-image']}>
14         <img src={mealsImage} alt="A table full of
15           </div>
16         </Fragment>
17       );
18
19       export default Header;
20 
```



## Adding Cart Button

```
└─ 07-FOOD-ORDER-APP
    ├─ node_modules
    ├─ public
    └─ src
        ├─ assets
        │   └─ meals.jpeg
        ├─ components
        │   └─ Cart
        │       └─ CartIcon.js
        └─ Layout
            ├─ Header.js
            ├─ Header.module.css
            └─ HeaderCartButton.js
                └─ HeaderCartButton.module.css
        └─ Meals
        └─ UI
            └─ App.js
            └─ index.css
            └─ index.js
```

```
1
2     import CartIcon from "../Cart/CartIcon";
3     import classes from './HeaderCartButton.module.css';
4
5     Complexity is 7 It's time to do something...
6     const HeaderCartButton = () => {
7         return (
8             <button className={classes.button}>
9                 <span className={classes.icon}>
10                    <CartIcon />
11                </span>
12                <span>You Cart</span>
13                <span className={classes.badge}>3</span>
14            </button>
15        );
16
17     export default HeaderCartButton;
```



## Adding a meals component

The screenshot shows a code editor with a file tree on the left and a code editor window on the right.

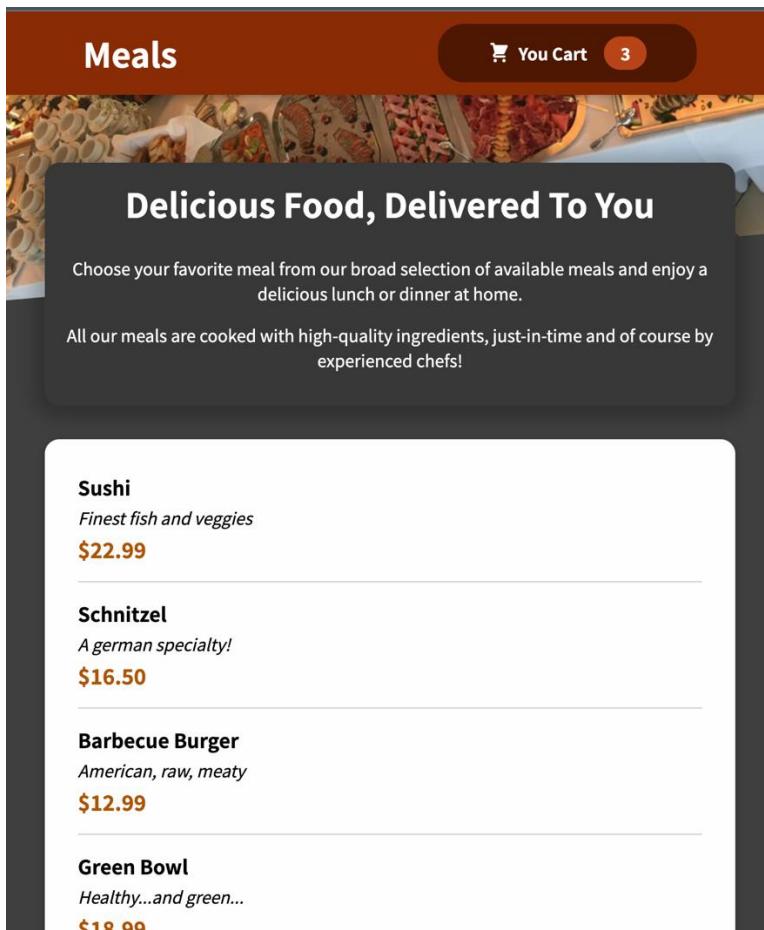
**File Tree:**

- 07-FOOD-O... (Project Root)
- node\_modules
- public
- src
  - assets (meals.jpeg)
  - components
    - Cart (CartIcon.js)
    - Layout (Header.js, Header.module.c..., HeaderCartButton..., HeaderCartButton...)
    - Meals
      - MealItem (MealItem.js, MealItem.module..., AvailableMeals.js, AvailableMeals...., Meals.js, MealsSummary.js, MealsSummary....)
  - UI (Card.js, Card.module.css)

**Code Editor (AvailableMeals.js):**

```
src > components > Meals > AvailableMeals.js > ...
23   },
24   {
25     id: 'm4',
26     name: 'Green Bowl',
27     description: 'Healthy...and green...',
28     price: 18.99,
29   },
30 ];
31
Complexity is 7 It's time to do something...
32 const AvailableMeals = () => {
33   const mealsList = DUMMY_MEALS.map((meal) => (
34     <MealItem
35       key={meal.id}
36       name={meal.name}
37       description={meal.description}
38       price={meal.price}
39     />
40   ));
41
42   return (
43     <section className={classes.meals}>
44       <Card>
45         <ul>{mealsList}</ul>
46       </Card>
47     </section>
48   );
49 }
```

Below the code editor are tabs for PROBLEMS, OUTPUT, and TERMINAL. The TERMINAL tab is selected.



## Adding a Form

The screenshot shows a code editor with a dark theme. On the left is the Explorer sidebar showing the project structure of "07-FOOD-ORDER-APP". The main editor area is displaying the content of the "MealItemForm.js" file. The code defines a functional component "MealItemForm" that takes "props" and returns a form with an input field and a button.

```
src > components > Meals > MealItem > MealItemForm.js > MealItemForm
1 import Input from '....../UI/Input';
2 import classes from './MealItemForm.module.css';
3
4 Complexity is 5 Everything is cool!
5 const MealItemForm = props => {
6   return (
7     <form className={classes.form}>
8       <Input
9         label="Amount"
10        input={{
11          id: "amound",
12          type: "number",
13          min: "1",
14          max: "5",
15          step: "1",
16          defaultValue: "1",
17        }}
18      />
19      <button>+ Add</button>
20    </form>
21  );
22
23 export default MealItemForm;
```

The screenshot shows the VS Code interface with the following details:

- EXPLORER** view: Shows the project structure under "07-FOOD-O...".
  - node\_modules
  - public
  - src
    - assets (meals.jpeg)
    - components
      - Cart (CartIcon.js)
      - Layout (Header.js, Header.module.css, HeaderCartButton.js, HeaderCartButton.module.css)
      - Meals
      - MeallItem (MeallItem.js, MeallItem.module.css, MeallItemForm.js)
- INPUT.JS**: The active file contains the following code:

```
src > components > UI > Input.js > Input
1 import classes from './Input.module.css';
2
3 const Input = props => {
4   return (
5     <div className={classes.input}>
6       <label htmlFor={props.input.id}>{props.label}</label>
7       <input {...props.input}/>
8     </div>
9   );
10 }
11
12 export default Input;
```

- INPUT MODULE.CSS**: The file contains the following CSS rule:

```
input { border: 1px solid #ccc; padding: 5px; }
```

The screenshot shows a web browser window at [localhost:3000](http://localhost:3000) displaying the meal delivery application.

**Header:**

- Navigation icons: back, forward, search, refresh.
- Address bar: localhost:3000.
- User icon: A small profile picture.
- Cart icon: You Cart (3).

**Section Headers:**

- Meals**
- Delicious Food, Delivered To You**

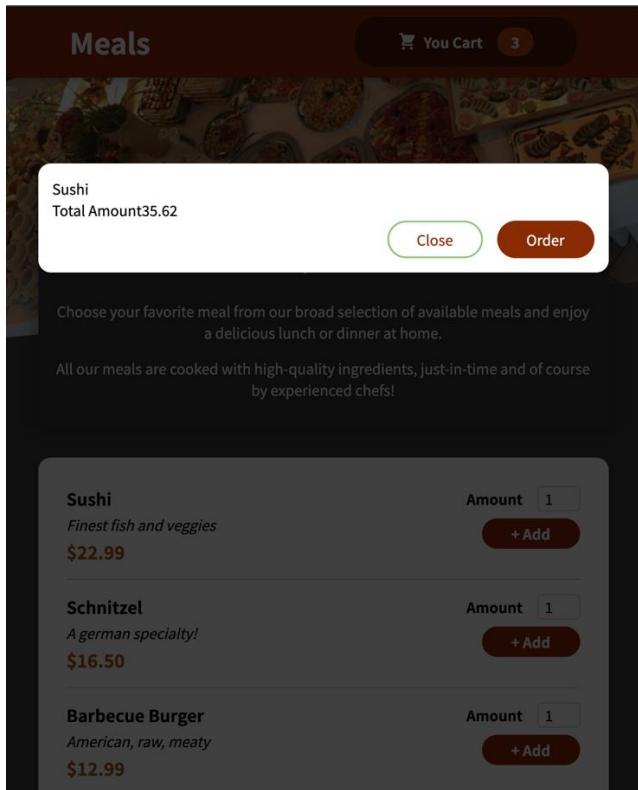
**Text Content:**

- Choose your favorite meal from our broad selection of available meals and enjoy a delicious lunch or dinner at home.
- All our meals are cooked with high-quality ingredients, just-in-time and of course by experienced chefs!

**Meal Listings:**

Meal Name	Description	Price	Amount	Action
Sushi	Finest fish and veggies	\$22.99	1	+ Add
Schnitzel	A german specialty!	\$16.50	1	+ Add
Barbecue Burger	American, raw, meaty	\$12.99	1	+ Add
Green Bowl	Healthy...and green...	\$18.99	1	+ Add

## Adding a Model via React Portal



```
import { Fragment } from 'react';
import classes from './Modal.module.css';
import ReactDOM from 'react-dom';

Complexity is 3 Everything is cool!
const BackDrop = props => {
  return <div className={classes.backdrop}></div>
}

Complexity is 4 Everything is cool!
const ModelOverlay = props => {
  return <div className={classes.modal}>
    <div className={classes.content}>{props.children}</div>
  </div>;
};

const portalElement = document.getElementById('overlays');

Complexity is 5 Everything is cool!
const Modal = props => {
  return (
    <Fragment>
      {ReactDOM.createPortal(<BackDrop />, portalElement)}
      {ReactDOM.createPortal(
        <ModelOverlay>{props.children}</ModelOverlay>,
        portalElement
      )}
    </Fragment>
  );
}

export default Modal;
```

## Adding Cart Close Functionality

```
Complexity is 10 It's time to do something...
function App() {
  const [cartIsShown, setCartIsShown] = useState(false);

  const showCartHandler = () => {
    setCartIsShown(true);
  }

  const hideCartHandler = () => {
    setCartIsShown(false);
  }

  return (
    <Fragment>
      {cartIsShown && <Cart onCloseCart={hideCartHandler}>/>}
      <Header onShowCart={showCartHandler}></Header>
      <main>
        <Meals />
      </main>
    </Fragment>
  );
}

export default App;
```

## Adding Cart Context

```
Complexity is 5 Everything is cool!
const CartProvider = (props) => { █
  const addItemToCartHandler = (items) =>{
    }

  const removeItemFromCartHandler = (id) => {};

  const cartContext = {
    items: [],
    totalAmount:0,
    addItem:addItemToCartHandler,
    removeItem:removeItemFromCartHandler
  };

  return <CartContext.Provider value={cartContext}>
    {props.children}
  </CartContext.Provider>
};

export default CartProvider;
```

```
const CartContext = React.createContext({
  items: [],
  totalAmount: 0,
  addItem: (item) => {},
  removeItem: (id) => {}
});

export default CartContext;
```

```
return (
  <CartProvider>
    {cartIsShown && <Cart onCloseCart={hideCartHandle}>
      <Header onShowCart={showCartHandler}/>
      <main>
        <Meals />
      </main>
    </CartProvider>
  );
}
```

## Using Cart Context

```
import { useContext } from "react";
import CartContext from "../../store/cart-context";
import CartIcon from "../Cart/CartIcon";
import classes from './HeaderCartButton.module.css';

Complexity is 9 It's time to do something...
const HeaderCartButton = (props) => { █
  const cartCtx = useContext(CartContext);
  const numberofCartItems = cartCtx.items.reduce((curNumber,item)=>{
    return curNumber + item.totalAmount;
},0);█

  return (
    <button className={classes.button} onClick={props.onClick}>
      <span className={classes.icon}>
        <CartIcon />
      </span>
      <span>You Cart</span>
      <span className={classes.badge}>{numberofCartItems}</span>
    </button>
  );
}

export default HeaderCartButton;
```

## Using Cart Reducer

```
import CartContext from "./cart-context";
import { useReducer } from "react";

const defaultCartState = {
  items: [],
  totalAmount: 0
};

Complexity is 6 It's time to do something...
const cartReducer = (state, action) => { █
  if (action.type === 'ADD') {
    const updateItems = state.items.concat(action.item);
    const updatedTotalAmount = state.totalAmount + action.item.price * a
    return {
      items: updateItems,
      totalAmount: updatedTotalAmount
    }
  } else if (action.type === 'REMOVE') {

  }
  return defaultCartState;
}

Complexity is 5 Everything is cool!
const CartProvider = (props) => { █
  const [cartState, dispatchCartAction] = useReducer(
    cartReducer,
    defaultCartState
  );

  const addItemToCartHandler = (item) => {
    dispatchCartAction({ type: "ADD", item: item });
  };

  const removeItemFromCartHandler = (id) => {
    dispatchCartAction({ type: "REMOVE", id: id });
  };

  const cartContext = {
    items: cartState.items,
    totalAmount: cartState.totalAmount,
    addItem: addItemToCartHandler,
    removeItem: removeItemFromCartHandler,
  };

  return (
    <CartContext.Provider value={cartContext}>
      {props.children}
    </CartContext.Provider>
  );
}
```

## Using and forwarding Ref's

```
import React from 'react';
import classes from './Input.module.css';

Complexity is 5 Everything is cool!
const Input = React.forwardRef((props, ref) => { █
  return (
    <div className={classes.input}>
      <label htmlFor={props.input.id}>{props.l...</l...
      <input ref ={ref} {...props.input}/>
    </div>
  );
});

export default Input;
```

```

import { useRef, useState } from 'react';
import Input from '../../../../../UI/Input';
import classes from './MealItemForm.module.css';

Complexity is 12 You must be kidding
const MealItemForm = props => { █

  const [amountIsValid, setAmountIsValid] = useState(true);
  const amountInputRef = useRef();

💡

Complexity is 5 Everything is cool!
  const submitHandler = event => { █
    event.preventDefault();

    const enteredAmount = amountInputRef.current.value;
    const enteredAmountNumber = +enteredAmount;

    if (
      enteredAmount.trim().length === 0 ||
      enteredAmountNumber < 1 ||
      enteredAmountNumber > 5
    ) {
      setAmountIsValid(false);
      return;
    }

    props.onAddToCart(enteredAmountNumber);
  }

  return (
    <form className={classes.form} onSubmit={submitHandler}>
      <Input
        ref={amountInputRef}
        label="Amount"
        input={{
          id: "amount_" + props.id,
          type: "number",
          min: "1",
          max: "5",
          step: "1",
          defaultValue: "1",
        }}
      />
      <button>+ Add</button>
      {!amountIsValid && <p>Please enter a valid amount</p>}
    </form>
  );
}

```



## Reducer logic for adding and removing items from cart

```
const cartReducer = (state, action) => {  
  if (action.type === 'ADD') {  
    const updatedTotalAmount =  
      state.totalAmount + action.item.price * action.item.amount;  
  
    const existingCartItemIndex = state.items.findIndex(  
      (item) => item.id === action.item.id  
    );  
  
    const existingCartItem = state.items[existingCartItemIndex];  
    let updatedItems;  
    let updatedItem;  
  
    if (existingCartItem) {  
      updatedItem = {  
        ...existingCartItem,  
        amount: existingCartItem.amount + action.item.amount,  
      };  
      updatedItems = [...state.items];  
      updatedItems[existingCartItemIndex] = updatedItem;  
    } else {  
      updatedItems = state.items.concat(action.item);  
    }  
    return {  
      items: updatedItems,  
      totalAmount: updatedTotalAmount,  
    };  
  } else if (action.type === 'REMOVE') {  
    const existingCartItemIndex = state.items.findIndex(  
      (item) => item.id === action.id  
    );  
    const existingCartItem = state.items[existingCartItemIndex];  
    const updatedTotalAmount = state.totalAmount - existingCartItem.price;  
    let updatedItems;  
    if (existingCartItem.amount === 1) {  
      updatedItems = state.items.filter(item => item.id !== action.id);  
    } else {  
      const updatedItem = [  
        ...existingCartItem,  
        amount: existingCartItem.amount - 1,  
      ];  
      updatedItems = [...state.items];  
      updatedItems[existingCartItemIndex] = updatedItem;  
    }  
    return {  
      items: updatedItems,  
      totalAmount: updatedTotalAmount,  
    };  
  }  
};
```



## Implementing Animation to Cart Button

```
const btnClasses = `${classes.button} ${btnIsHighlighted ? classes.bump: ''}`  
  
Complexity is 6 It's time to do something...  
useEffect(()=>{  
  if (items.length === 0) {  
    return;  
  }  
  setBtnIsHighlighted(true);  
  
  const timer = setTimeout(() => {  
    setBtnIsHighlighted(false);  
  }, 300);  
  
  return ()=> {  
    clearTimeout(timer);  
  };  
, [items]);  
  
return (  
  <button className={btnClasses} onClick={props.onClick}>  
    <span className={classes.icon}>
```

**Meals**

You Cart 4

**Delicious Food, Delivered To You**

Choose your favorite meal from our broad selection of available meals and enjoy a delicious lunch or dinner at home.

All our meals are cooked with high-quality ingredients, just-in-time and of course by experienced chefs!

**Sushi**  
*Finest fish and veggies*  
**\$22.99**

**Schnitzel**  
*A german specialty!*  
**\$16.50**

**Barbecue Burger**  
*American, raw, meaty*  
**\$12.99**

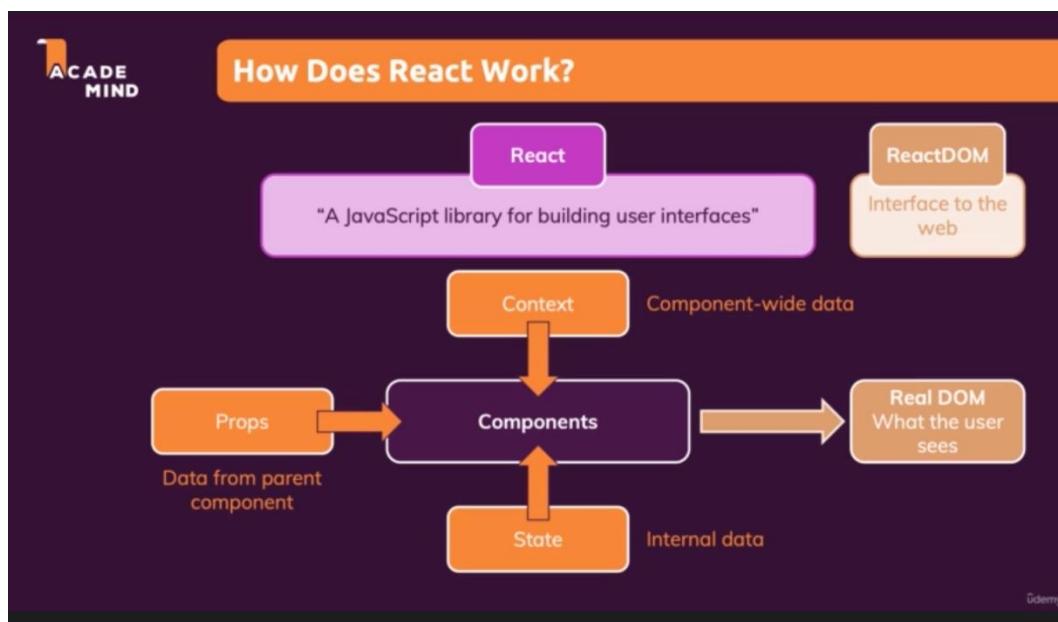
## How React Works?

ACADE MIND

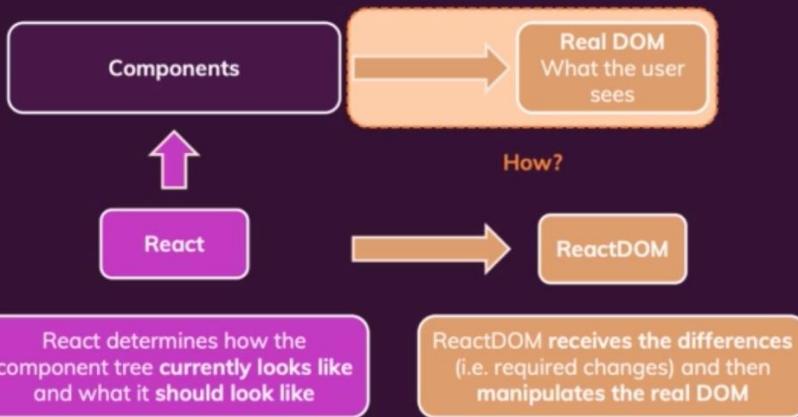
### Module Content

- How Does React Work Behind The Scenes?
- Understanding the Virtual DOM & DOM Updates
- Understanding State & State Updates

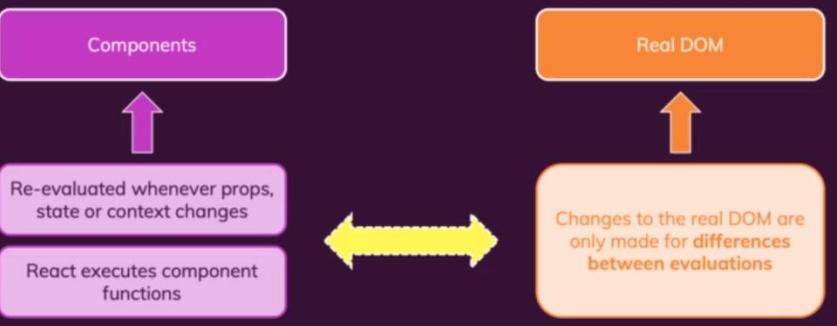
## Understanding



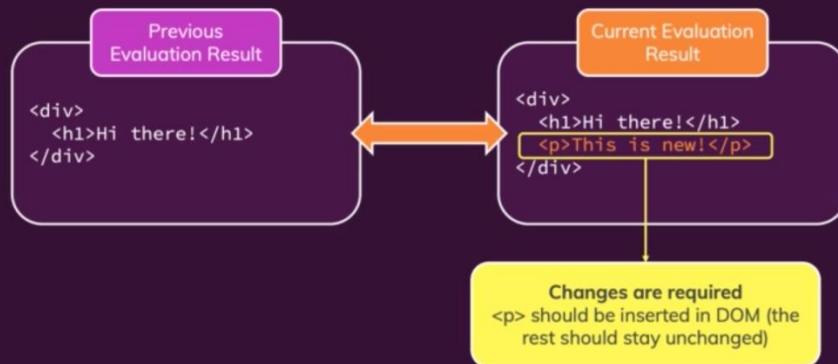
## How Does React Work?



## Re-Evaluating Components != Re-Rendering the DOM



## Virtual DOM Diffing



## Demo - Inserting the element without refreshing the UI

The screenshot shows the browser's developer tools with the "Elements" tab selected. The DOM tree on the left displays the HTML structure of the application. The main content area on the right shows a UI component with a heading "Hi there!" and a button labeled "Toogle Paragraph". Below the component, the browser's status bar indicates "Complexity is 9 It's time to do something...".

```
Complexity is 9 It's time to do something...
function App() { [redacted]

  const [showParagraph, setShowParagraph] = useState(false);

  const toogleParagraphHandler = () => {
    setShowParagraph((showParagraph) => !showParagraph);
  }

  const newParagraph = <p>New Paragraph</p>;
  return (
    <div className="app">
      <h1>Hi there!</h1>
      {showParagraph && newParagraph}
      <Button onClick={toogleParagraphHandler}>Toogle Paragraph</Button>
    </div>
  );
}

export default App;
```

Child component re evaluation

Child component re-executed when parent component state changes.

The screenshot shows the browser's developer tools with the "Elements" tab selected. The DOM tree on the left displays the HTML structure of the application. The main content area on the right shows a UI component with a heading "Hi there!" and a button labeled "Toogle Paragraph". A yellow arrow points from the "show" prop of the child component back to the parent component's state update logic. Below the component, the browser's status bar indicates "Complexity is 8 It's time to do something...".

```
Complexity is 8 It's time to do something...
function App() { [redacted]

  const [showParagraph, setShowParagraph] = useState(false);

  const toogleParagraphHandler = () => {
    setShowParagraph((showParagraph) => !showParagraph);
  }

  return (
    <div className="app">
      <h1>Hi there!</h1>
      <DemoOutput show={showParagraph} /> ←
      <Button onClick={toogleParagraphHandler}>Toogle Paragraph</Button> ←
    </div>
  );
}

export default App;
```

If we use the React.memo() it prevents re-evaluation of the component and its child component if state or object passed is not changes.

The screenshot shows a code editor with the following details:

- Code Editor:** App.js (selected file).

```
import Button from './Components/UI/Button/Button';
import DemoOutput from './Components/Demo/DemoOutput';
import './App.css';

Complexity is 8 it's time to do something...
function App() {
  const [showParagraph, setShowParagraph] = useState(false);

  const toggleParagraphHandler = () => {
    setShowParagraph((showParagraph) => !showParagraph)
  }

  return (
    <div className="app">
      <h1>Hi there!</h1>
      <DemoOutput show={false} />
      <button onClick={toggleParagraphHandler}>Toggle P:</button>
    </div>
  );
}

export default App;
```
- Terminal:** Compiled with warnings.

```
src/App.js
Line 8:10: 'showParagraph' is
assigned a value but never used
no-unused-vars
```
- Debug Console:** No output shown.

Here we used memo in Button component also, but button component executed when no value changes. That's because, the function object we passed through onClick props, will be created every time App component re-evaluated.

```
> false === false
<- true
> 'hi' === 'hi'
<- true
> [1, 2, 3] === [1, 2, 3]
<- false
> |
```

## Preventing Function Re-Creation using useCallback()

- useCallback() hook help us to save a function to help preventing recreation of it when component function re-evaluated.
- Saves the object in memory and reuse it when component function re-evaluated.
- First argument is function that we want to store in memory and second argument is dependency array.

Complexity is 8 It's time to do something...

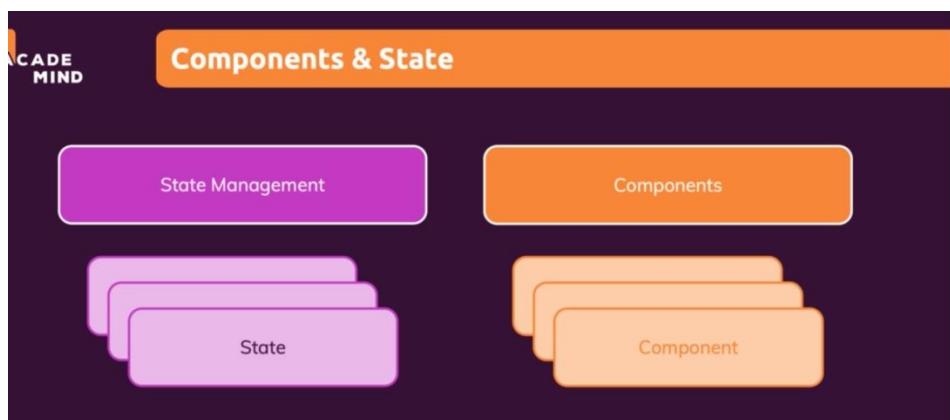
```
function App() { █  
  
  const [showParagraph, setShowParagraph] = useState(false);  
  
  const toggleParagraphHandler = useCallback(() => {  
    setShowParagraph((showParagraph) => !showParagraph);  
  }, []);  
  
  return (  
    <div className="app">  
      <h1>Hi there!</h1>  
      <DemoOutput show={false} />  
      <Button onClick={toggleParagraphHandler}>Toggle Paragraph</Button>  
    </div>  
  );  
}  
  
export default App;
```



Using callback with dependencies changes.

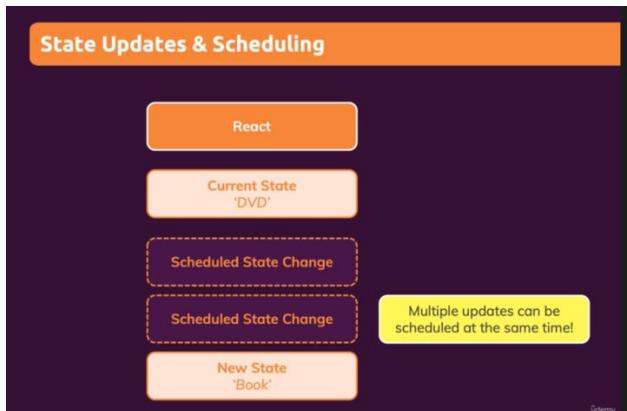
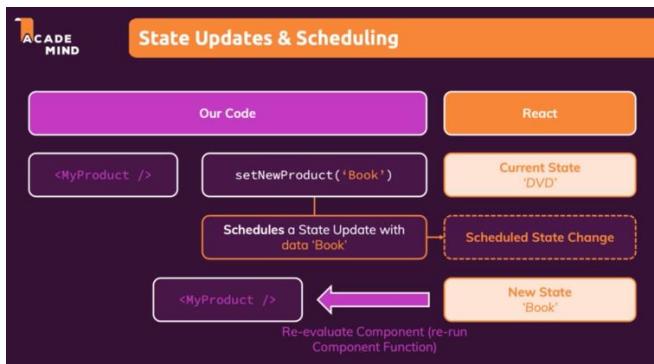
```
import React, { useState, useCallback } from 'react';  
import Button from './components/UI/Button/Button';  
import DemoOutput from './components/Demo/DemoOutput';  
import './App.css';  
  
Complexity is 11 You must be kidding  
function App() { █  
  
  const [showParagraph, setShowParagraph] = useState(false);  
  const [allowToggle, setAllowToggle] = useState(false);  
  
  Complexity is 3 Everything is cool!  
  const toggleParagraphHandler = useCallback(() => { █  
    if (allowToggle){  
      setShowParagraph((showParagraph) => !showParagraph);  
    }  
  },[allowToggle]);  
  
  const allowToggleHandler = () => {  
    setAllowToggle(true);  
  }  
}
```

Component and state



State is managed by react and connect with component.

## State update and Scheduling



```
const toggleParagraphHandler = useCallback(() => {
  if (allowToggle) {
    setShowParagraph((prevShowParagraph) => !prevShowParagraph);
  }
}, [allowToggle]);

const allowToggleHandler = () => {
  setAllowToggle(true);
};

return (
  <div className="app">
    <DemoList title={listTitle} items={listItems} />
    <Button onClick={changeTitleHandler}>Change List Title</Button>
  </div>
);
```

Due to the multiple state change, using the function in the state function helps. It always takes the previous state to update the state. Which means it always look into the latest state before updating the state.

State patching happens when multiple states are updated together.

### useMemo() hook

```
function App() {
  const [listTitle, setListTitle] = useState('My List');

  const changeTitleHandler = useCallback(() => {
    setListTitle('New Title');
  }, []);

  const listItems = useMemo(() => [5, 3, 1, 10, 9], []);
  //-----^

  return (
    <div className="app">
      <DemoList title={listTitle} items={listItems} />
      <Button onClick={changeTitleHandler}>Change List Title</Button>
    </div>
  );
}
```

```
const DemoList = (props) => {
  const { items } = props;

  const sortedList = useMemo(() => [
    items.sort((a, b) => a - b),
  ], [items]);
  console.log('DemoList RUNNING');

  return (
    <div className={classes.list}>
      {sortedList}
    </div>
  );
};
```

- useMemo() hook used to prevent the recreation of the object or function when component function reevaluated.

## Class based Components

**Module Content**

- What & Why?
- Working with Class-based Components
- Error Boundaries

Class based components alternative to Functions

**Class-based Components: An Alternative To Functions**

Default & Most Modern Approach!	Functional Components
function Product(props) { return <h2>A Product!</h2> }	Class-based Components
Components are regular JavaScript functions which return renderable results (typically JSX)	class Product extends Component { render() { return <h2>A Product!</h2> } }  Components can also be defined as JS classes where a render() method defines the to-be-rendered output

**Traditionally (React < 16.8), you had to use Class-based Components to manage “State”**

**React 16.8 introduced “React Hooks” for Functional Components**

**Class-based Components Can't Use React Hooks!**

Functional component and class-based component can work together.

## First class-based component

```
import classes from './User.module.css';
import {Component} from 'react';

Complexity is 3 Everything is cool!
class User extends Component { █
    Complexity is 3 Everything is cool!
    render() { █
        return <li className={classes.user}>{this.props.name}</li>;
    }
}

// const User = (props) => {
//     return <li className={classes.user}>{props.name}</li>;
// };

export default User;
```

- Should extend 'Component' class
- Render function returns the JSX code.
- Need to use the 'this' keyword to access the props in class-based component.

## Managing state in class-based component

Complexity is 9 It's time to do something...

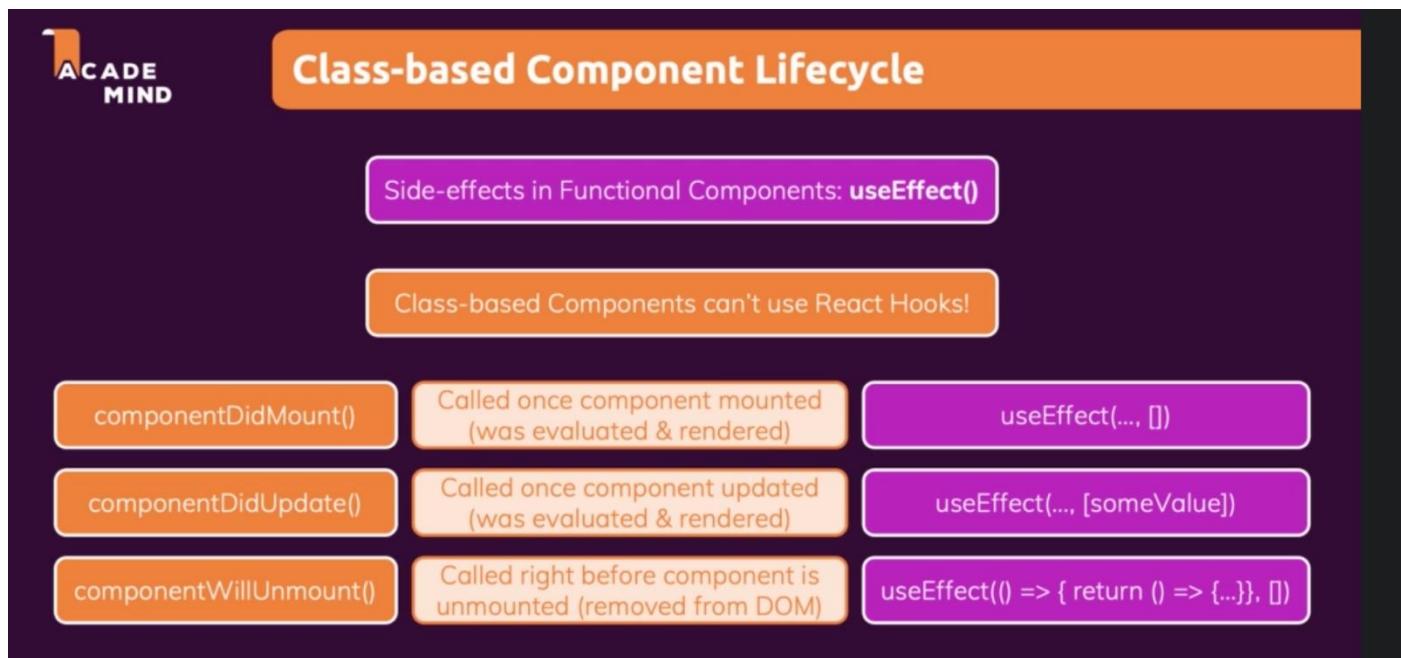
```
class Users extends Component {  
  constructor(){  
    super();  
    this.state = {  
      showUsers:true,  
      more: 'Test'  
    }  
  }  
}
```

Complexity is 3 Everything is cool!

```
toggleUserHandler() {  
  this.setState((curState) => {  
    return {showUsers : !curState.showUsers}  
  });  
}
```

Complexity is 9 It's time to do something...

```
render() {  
  const usersList = (  
    <ul>  
      {DUMMY_USERS.map((user) => (  
        <User key={user.id} name={user.name} />  
      ))}  
    </ul>  
  );  
  return (  
    <div className={classes.users}>  
      <button onClick={this.toggleUsersHandler.bind(this)}>  
        {this.state.showUsers ? 'Hide' : 'Show'} Users  
      </button>  
      {this.state.showUsers && usersList}  
    </div>  
  );  
}
```



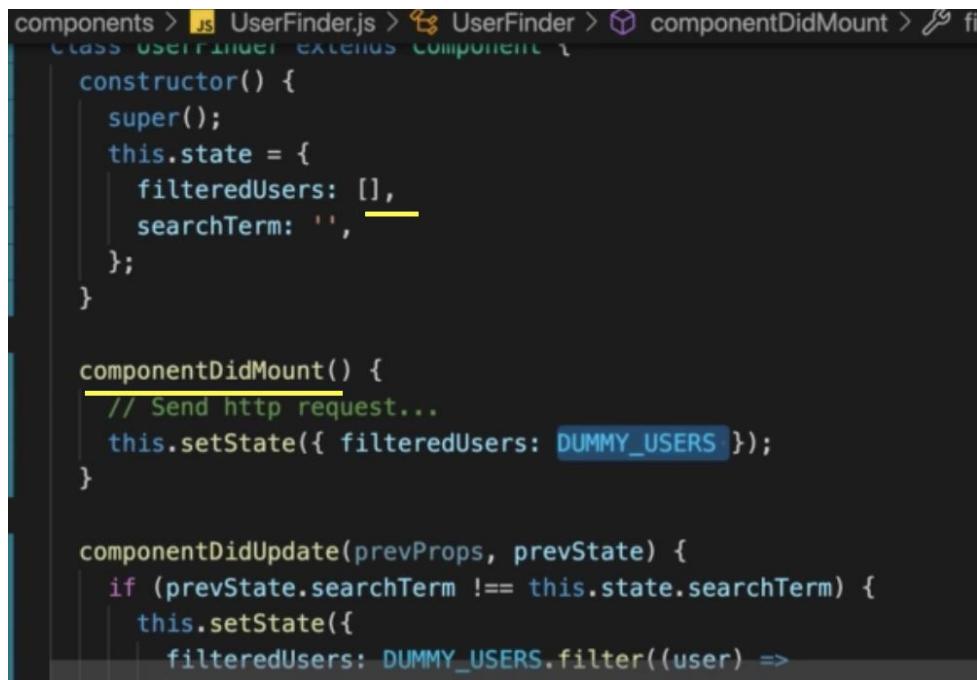
```
class UserFinder extends Component {
  constructor() {
    this.state = {
      filteredUsers: DUMMY_USERS,
      searchTerm: '',
    };
  }

  componentDidUpdate(prevProps, prevState) {
    if (prevProps.searchTerm !== this.state.searchTerm) {}
    this.setState({
      filteredUsers: DUMMY_USERS.filter((user) =>
        user.name.includes(searchTerm)
      ),
    });
  }
}
```

componentDidUpdate()

```
componentDidUpdate(prevProps, prevState) {
  if (prevState.searchTerm !== this.state.searchTerm) {}
  this.setState({
    filteredUsers: DUMMY_USERS.filter((user) =>
      user.name.includes(this.state.searchTerm)
    ),
  });
}
```

```
componentDidMount()
```



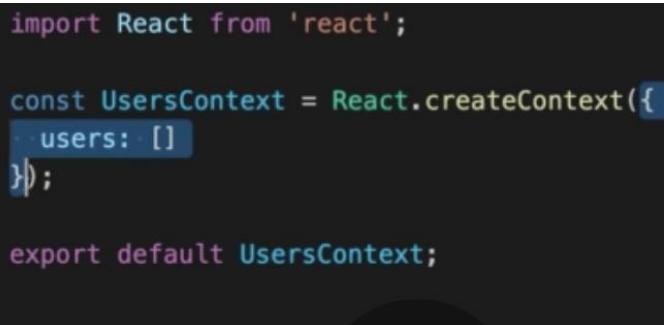
```
components > UserFinder.js > UserFinder > componentDidMount > file
class UserFinder extends Component {
  constructor() {
    super();
    this.state = {
      filteredUsers: [],
      searchTerm: '',
    };
  }

  componentDidMount() {
    // Send http request...
    this.setState({ filteredUsers: DUMMY_USERS });
  }

  componentDidUpdate(prevProps, prevState) {
    if (prevState.searchTerm !== this.state.searchTerm) {
      this.setState({
        filteredUsers: DUMMY_USERS.filter((user) =>
```

componentDidMount() function will be executed when component initialized.

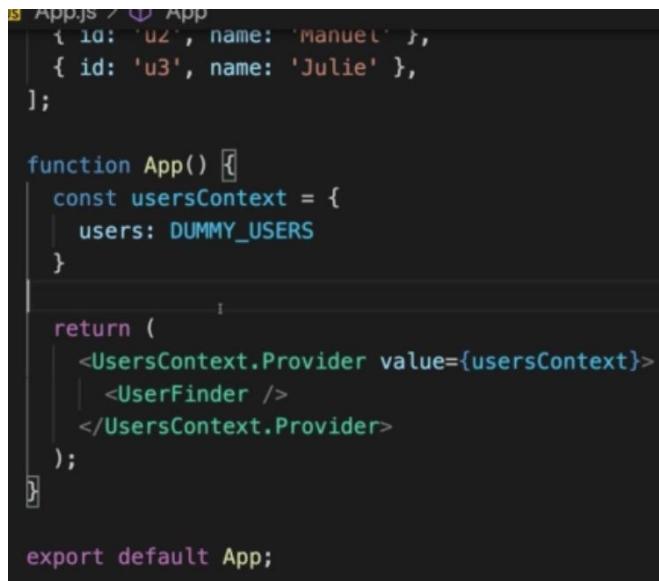
Using context in class-based component



```
import React from 'react';

const UsersContext = React.createContext({
  users: []
});

export default UsersContext;
```



```
App.js > App
  { id: 'u2', name: 'Manuel' },
  { id: 'u3', name: 'Julie' },
];

function App() {
  const usersContext = {
    users: DUMMY_USERS
  }

  return (
    <UsersContext.Provider value={usersContext}>
      <UserFinder />
    </UsersContext.Provider>
  );
}

export default App;
```

Only one context per class.

```
import UsersContext from '../store/users-context';

class UserFinder extends Component {
  static contextType = UsersContext;

  constructor() {
    super();
    this.state = {
      filteredUsers: [],
      searchTerm: '',
    };
  }
}
```



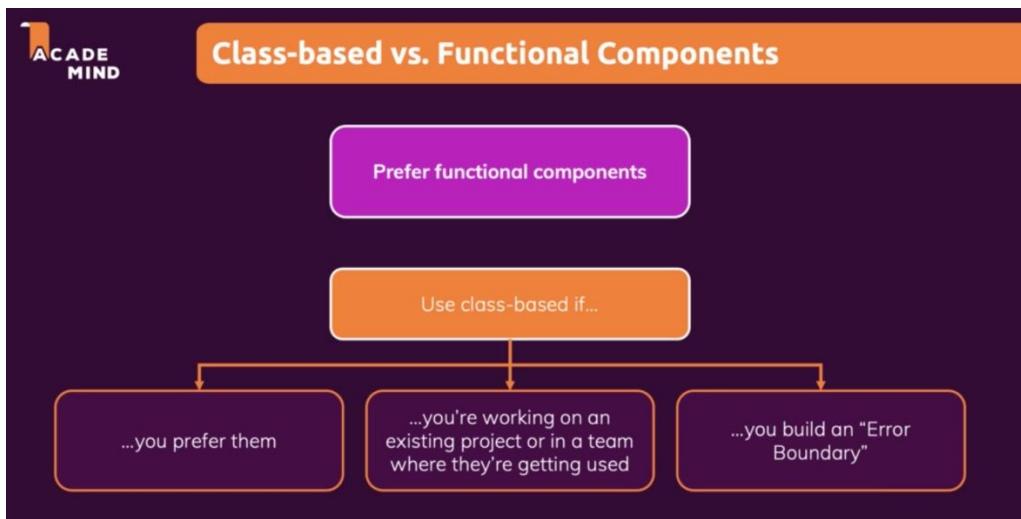
```
componentDidMount() {
  // Send http request...
  this.setState({ filteredUsers: this.context.users });
}

componentDidUpdate(prevProps, prevState) {
  if (prevState.searchTerm !== this.state.searchTerm) {
    this.setState([
      filteredUsers: this.context.users.filter((user) =>
        user.name.includes(this.state.searchTerm)
      ),
    ]);
  }
}
```

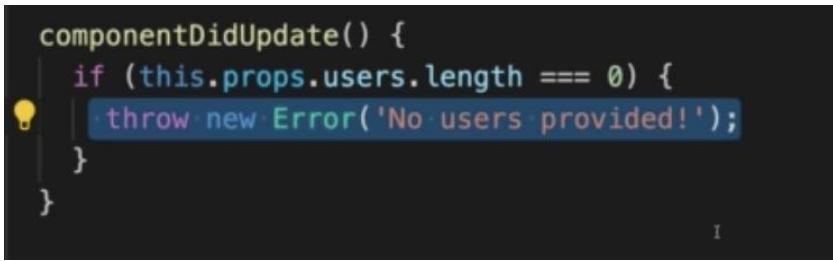


Functional components are leaner and flexible when compared to the class based component.

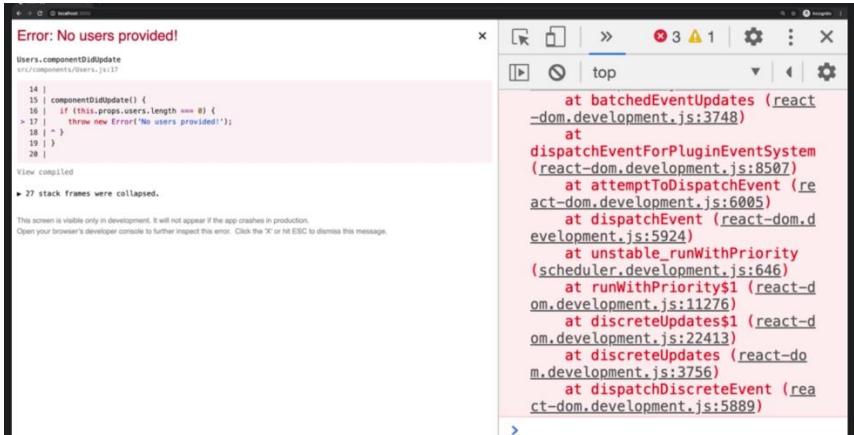
Class Based Component vs Functional component



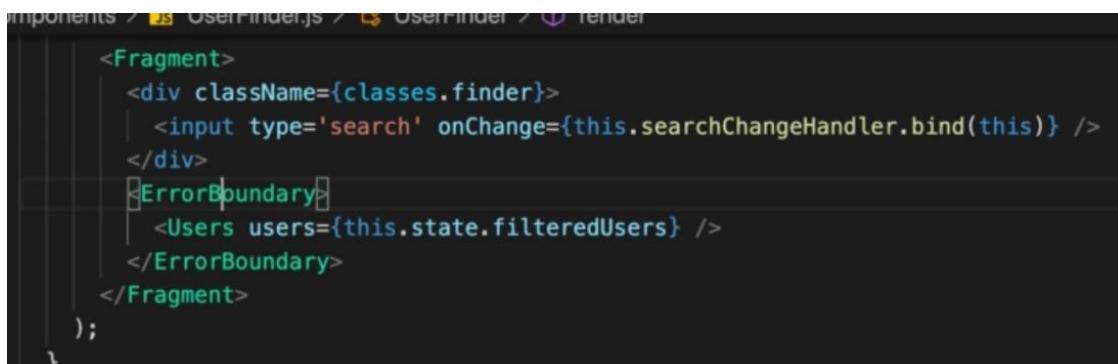
## Error Boundary



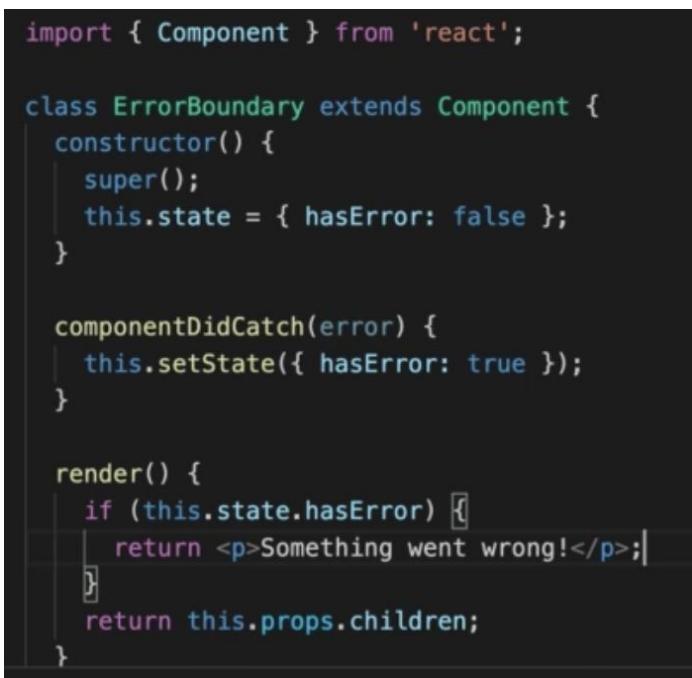
```
componentDidUpdate() {
  if (this.props.users.length === 0) {
    throw new Error('No users provided!');
  }
}
```



Having error in one component and handling it in another component.



```
<Fragment>
  <div className={classes.finder}>
    <input type='search' onChange={this.searchChangeHandler.bind(this)} />
  </div>
  <ErrorBoundary>
    <Users users={this.state.filteredUsers} />
  </ErrorBoundary>
</Fragment>
};
```



```
import { Component } from 'react';

class ErrorBoundary extends Component {
  constructor() {
    super();
    this.state = { hasError: false };
  }

  componentDidCatch(error) {
    this.setState({ hasError: true });
  }

  render() {
    if (this.state.hasError) {
      return <p>Something went wrong!</p>;
    }
    return this.props.children;
  }
}
```

Error boundary cannot be implemented in functional component. Error boundary is limitation of functional component.

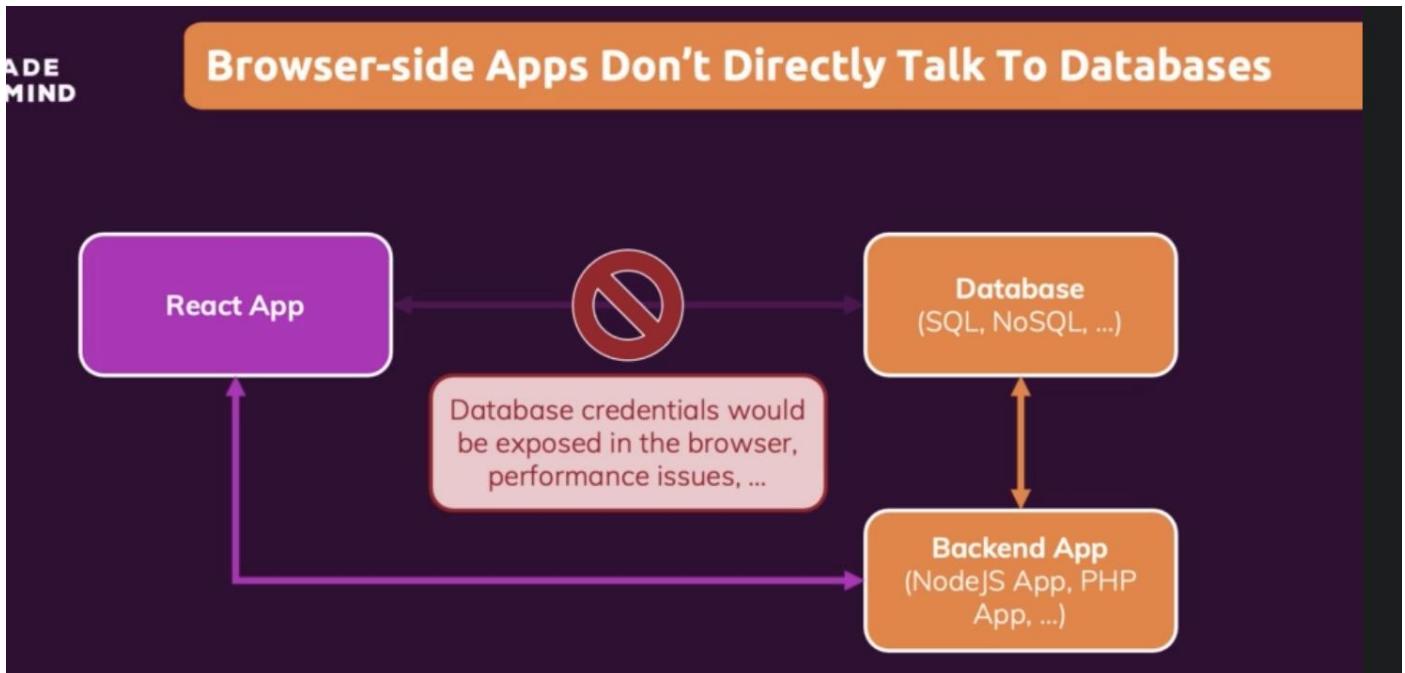
## HTTP Request

ACADE MIND

### Module Content

- How do React Apps Interact with Databases?
- Sending Http Requests & Using Responses
- Handling Errors & Loading State

Browser-side Apps Don't Directly Talk to Databases



## Using fetch and then functions

```
const [movies, setMovies] = useState([]);

Complexity is 6 It's time to do something...
function fetchMovieHandler() { █
  fetch('https://swapi.dev/api/films').then(response => {
    return response.json();
  })
  .then(data => { █
    const transformedMovies = data.results.map(movieData => {
      return {
        id: movieData.episode_id,
        title: movieData.title,
        openingText: movieData.opening_crawl,
        releaseDate: movieData.release_date
      }
    })
    setMovies(transformedMovies)
  });
}
```

fetch() supports both GET and POST request. By default, GET request.

## Using Async Await

```
Complexity is 3 Everything is cool!
async function fetchMovieHandler() { █
  const response = await fetch('https://swapi.dev/api/films');
  const data = await response.json();
  const transformedMovies = data.results.map(movieData => {
    return {
      id: movieData.episode_id,
      title: movieData.title,
      openingText: movieData.opening_crawl,
      releaseDate: movieData.release_date
    }
  })
  setMovies(transformedMovies)
}
```

## Handling the Loading with States

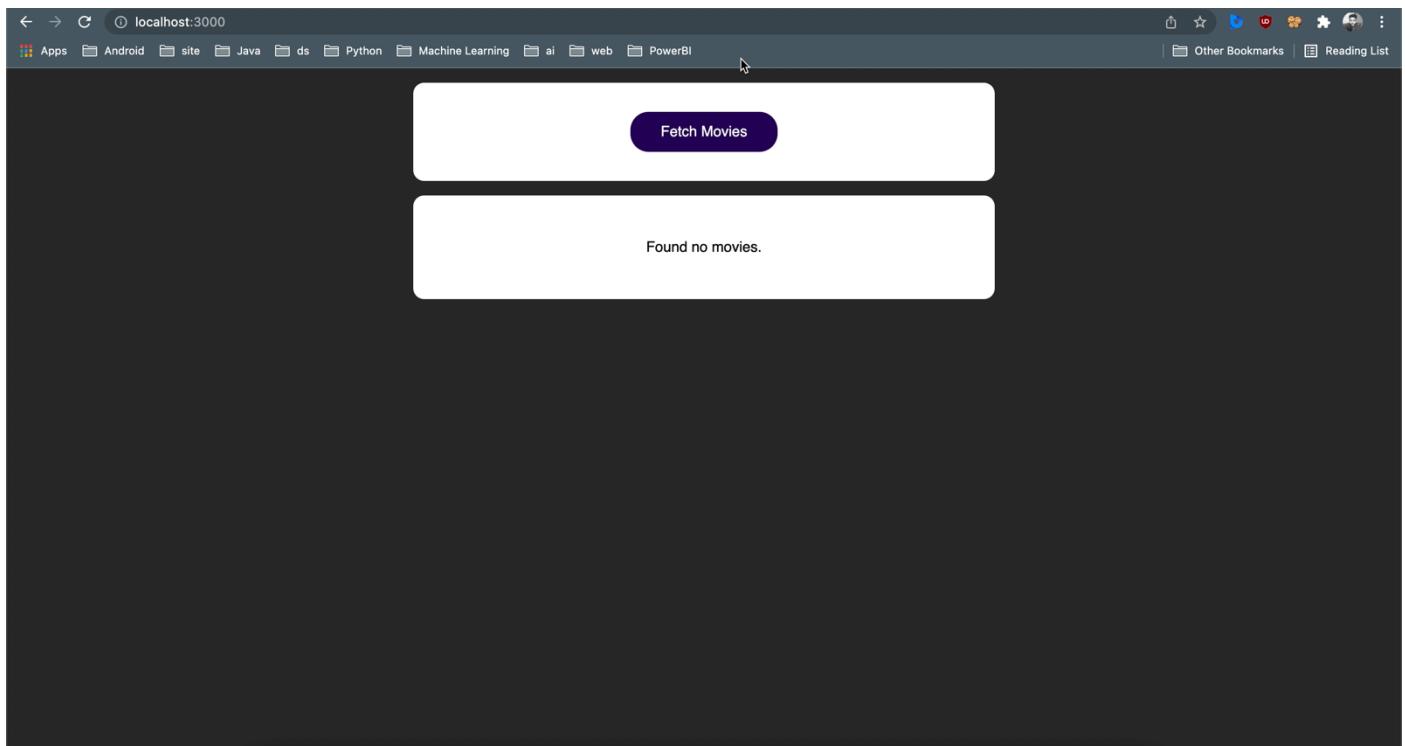
```

const [movies, setMovies] = useState([]);
const [isLoading, setIsLoading] = useState(false);

Complexity is 3 Everything is cool!
async function fetchMovieHandler() {
  setIsLoading(true)
  const response = await fetch('https://swapi.dev/api/films');
  const data = await response.json();
  const transformedMovies = data.results.map(movieData => {
    return {
      id: movieData.episode_id,
      title: movieData.title,
      openingText: movieData.opening_crawl,
      releaseDate: movieData.release_date
    }
  })
  setMovies(transformedMovies)
  setIsLoading(false)
}

return (
  <React.Fragment>
    <section>
      <button onClick={fetchMovieHandler}>Fetch Movies</button>
    </section>
    <section>
      {!isLoading && movies.length > 0 && <MoviesList movies={movies} />}
      {!isLoading && movies.length === 0 && <p>Found no movies.</p>}
      {isLoading && <p>Loading...</p>}
    </section>
  </React.Fragment>
);

```



## Handling error using try catch

```
Complexity is 7 It's time to do something...
async function fetchMovieHandler() { █
  setIsLoading(true)
  setError(null)
  try {
    const response = await fetch('https://swapi.dev/api/films');
    if (!response.ok) {
      throw new Error('Something went wrong!')
    }
    const data = await response.json();

    const transformedMovies = data.results.map(movieData => {
      return {
        id : movieData.episod_id,
        title: movieData.title,
        openingText: movieData.opening_crawl,
        releaseDate: movieData.release_date
      }
    })
    setMovies(transformedMovies)
  } catch (error) {
    setError(error.message)
  }
  setIsLoading(false)
}
```

## Using UseEffect for sending Http Request

```
const [movies, setMovies] = useState([]);
const [isLoading, setIsLoading] = useState(false);
const [error, setError] = useState(null);

Complexity is 7 It's time to do something...
const fetchMovieHandler = useCallback(async () => { █
  setIsLoading(true)
  setError(null)
  try {
    const response = await fetch('https://swapi.dev/api/films');
    if (!response.ok) {
      throw new Error('Something went wrong!')
    }
    const data = await response.json();

    const transformedMovies = data.results.map(movieData => {
      return {
        id : movieData.episod_id,
        title: movieData.title,
        openingText: movieData.opening_crawl,
        releaseDate: movieData.release_date
      }
    })
    setMovies(transformedMovies)
  } catch (error) {
    setError(error.message)
  }
  setIsLoading(false)
}, []);

useEffect(()=>{
  fetchMovieHandler();
}, [fetchMovieHandler])
|
```

## Http POST request

```
async function addMovieHandler(movie) {
  console.log(movie);
  const response = await fetch('https://react-http-5c577-default-firebase.com/movies.json'
    method: 'POST',
    body: JSON.stringify(movie),
    headers: {
      'Content-Type': 'application/json'
    }
  );
  const data = await response.json()
  console.log(data)
  fetchMoviesHandler()
}
```

```
const fetchMoviesHandler = useCallback(async () => {
  setIsLoading(true);
  setError(null);
  try {
    const response = await fetch('https://react-http-5c577-default-firebase.com/movies.json');
    if (!response.ok) {
      throw new Error('Something went wrong!');
    }

    const data = await response.json();

    const loadedMovies = []

    for (const key in data) {
      loadedMovies.push({
        id: key,
        title: data[key].title,
        openingText: data[key].openingText,
        releaseDate: data[key].releaseDate,
      })
    }

    setMovies(loadedMovies);
  } catch (error) {
    setError(error.message);
  }
  setIsLoading(false);
}
```

## Building Custom Hooks



## Rules of Hooks

Only call React Hooks in **React Functions**

React Component Functions

Custom Hooks (covered later!)

Only call React Hooks at the **Top Level**

Don't call them in nested functions

Don't call them in any block statements

+ extra, unofficial Rule for **useEffect()**: ALWAYS add everything you refer to inside of useEffect() as a dependency!

## Module Content

What & Why?

Building a Custom Hook

Custom Hook Rules & Practices

What are Custom Hooks ?

### What are “Custom Hooks”?

Outsource **stateful** logic into **re-usable** functions



Unlike “regular functions”, custom hooks can use other React hooks and React state

## Creating Custom Hook

```
import {useState, useEffect} from 'react';

Complexity is 6 It's time to do something...
const useCounter = () => {
  const [counter, setCounter] = useState(0);

  Complexity is 5 Everything is cool!
  useEffect(() => {
    const interval = setInterval(() => {
      setCounter((prevCounter) => prevCounter + 1);
    }, 1000);

    return () => clearInterval(interval);
  }, []);
};

export default useCounter;
```

```
import Card from './Card';
import useCounter from '../hooks/use-counter';

Complexity is 3 Everything is cool!
const ForwardCounter = () => {
  const counter = useCounter();

  return <Card>{counter}</Card>;
};

export default ForwardCounter;
```

States defined in useCounter() (hook) are tied to component where it is being called. For each call, unique states will be defined.

## Configuring the custom hooks

```
Complexity is 10 It's time to do something...
const useCounter = (forwards = true) => {
  const [counter, setCounter] = useState(0);

  Complexity is 8 It's time to do something...
  useEffect(() => {
    Complexity is 5 Everything is cool!
    const interval = setInterval(() => {
      if (forwards) {
        setCounter((prevCounter) => prevCounter + 1)
      } else {
        setCounter((prevCounter) => prevCounter - 1)
      }
    }, 1000);

    return () => clearInterval(interval);
  }, [forwards]);
  return counter;
};

export default useCounter;
```

```
Complexity is 3 Everything is cool!
const BackwardCounter = () => {
  const counter = useCounter(false);
  return <Card>{counter}</Card>;
};

export default BackwardCounter;
```

## Custom Hook to handling Http Requests

```
Complexity is 11 You must be kidding
const useHttp = () => { █

  const [isLoading, setIsLoading] = useState(false);
  const [error, setError] = useState(null);

  Complexity is 9 It's time to do something...
  const sendRequest = useCallback(async (requestConfig, applyData) => { █
    setIsLoading(true);
    setError(null);
    try {
      const response = await fetch(
        requestConfig.url, [
          method: requestConfig.method ? requestConfig.method: 'GET',
          headers: requestConfig.headers ? requestConfig.headers: {},
          body: requestConfig.body ? JSON.stringify(requestConfig.body): null
        ]);
      if (!response.ok) {
        throw new Error('Request failed!');
      }
      const data = await response.json();
      applyData(data)
    } catch (err) {
      setError(err.message || 'Something went wrong!');
    }
    setIsLoading(false);
  }, []);
  return {
    isLoading,
    error,
    sendRequest
  };
};

export default useHttp;
```

```
Complexity is 10 It's time to do something...
function App() { █

  const [tasks, setTasks] = useState([]);

  const transformTasks = useCallback((data) => {
    const loadedTasks = [];
    for (const taskKey in data) {
      loadedTasks.push({ id: taskKey, text: data[taskKey].text });
    }
    setTasks(loadedTasks);
  }, []);

  const {isLoading, error, sendRequest: fetchTasks} = useHttp();

  useEffect(() => {
    fetchTasks({ url: "https://react-http-5c577-default-firebase.com/tasks.json" },
      transformTasks);
  }, []);

  const taskAddHandler = (task) => {
    setTasks((prevTasks) => prevTasks.concat(task));
  };

  return (
    <React.Fragment>
      <NewTask onAddTask={taskAddHandler} />
      <Tasks
        items={tasks}
        loading={isLoading}
        error={error}
        onFetch={fetchTasks}
      />
    </React.Fragment>
  );
};

export default App;
```

```

Complexity is 8 It's time to do something...
const NewTask = (props) => {
  const {isLoading, error, sendRequest} = useHttp();

  const createTask = (taskText, taskData) => {
    const generatedId = taskData.name; // firebase-specific => "name" contains generated id
    const createdTask = { id: generatedId, text: taskText };
    props.onAddTask(createdTask);
  }

  const enterTaskHandler = async (taskText) => {
    sendTaskRequest(
      {
        url: 'https://react-http-5c577-default.firebaseio.com/tasks.json',
        method: 'POST',
        body: { text: taskText },
        headers: {
          'Content-Type': 'application/json',
        }
      },
      createTask.bind(null,taskText)
    )
  };

  return (
    <Section>
      <TaskForm onEnterTask={enterTaskHandler} loading={isLoading} />
      {error && <p>{error}</p>}
    </Section>
  );
};

export default NewTask;

```

## Working with Form and User Input

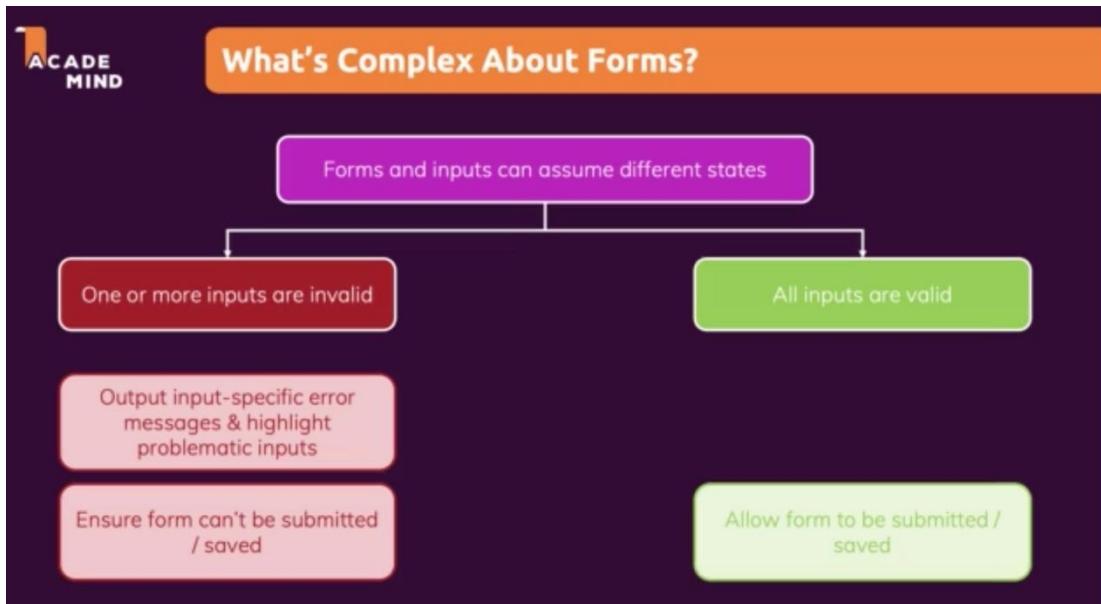


**ACADE MIND**

## Module Content

- What's Complex About Forms?
- Handling Inputs & Forms with React
- Simplifications

Dotenv



## Dealing with Form Submission and Getting input values

```
Complexity is 10 It's time to do something...
const SimpleInput = (props) => { [

  const nameInputRef = useRef();
  const [enteredName, setEnteredName] = useState('');

  const nameInputChangeHandler = event => {
    setEnteredName(event.target.value)
  }

  const formSubmissionHandler = event => {
    event.preventDefault();
    const enteredValue = nameInputRef.current.value;
    console.log(enteredValue);

    //nameInputRef.current.value = '' => NOT IdleDeadline, DON'T MANIPULATE
    setEnteredName(''); _____
  }

  return (
    <form onSubmit={formSubmissionHandler}>
      <div className='form-control'>
        <label htmlFor='name'>Your Name</label>
        <input
          ref={nameInputRef}
          type='text'
          id='name'
          onChange={nameInputChangeHandler}
        />
      </div>
      <div className="form-actions">
        <button>Submit</button>
      </div>
    </form>
  );
}

export default SimpleInput;
```

## Doing the validation on submission

```
Complexity is 16 You must be kidding
const SimpleInput = (props) => { █

  const nameInputRef = useRef();
  const [enteredName, setEnteredName] = useState('');
  const [enteredNameIsValid, setEnteredNameIsValid] = useState(true);
  const [enteredNameTouched, setEnteredNameTouched] = useState(false);

  const nameInputChangeHandler = event => {
    💡 setEnteredName(event.target.value)
  }

Complexity is 3 Everything is cool!
const formSubmissionHandler = event => { █
  event.preventDefault();
  setEnteredNameTouched(true)
  if (enteredName.trim() === '') {
    setEnteredNameIsValid(false);
    return;
  }
  setEnteredNameIsValid(true);

  console.log(enteredName);
  const enteredValue = nameInputRef.current.value;
  console.log(enteredValue);

  //nameInputRef.current.value = '' => NOT IdleDeadline, DON'T MANIPULATE -
  setEnteredName('');
}

const nameInputIsInvalid = !enteredNameIsValid && enteredNameTouched;

const nameInputClasses = nameInputIsInvalid
  ? 'form-control invalid'
  : 'form-control';

return (
  <form onSubmit={formSubmissionHandler}>
    <div className={nameInputClasses}>
      <label htmlFor='name'>Your Name</label>
      <input
        ref={nameInputRef}
        type='text'
        id='name'
        onChange={nameInputChangeHandler}
        value={enteredName}
      />
    </div>
  </form>
)
```

onBlur function will be triggered when input is loose focus.

```
return (
  <form onSubmit={formSubmissionHandler}>
    <div className={nameInputClasses}>
      <label htmlFor='name'>Your Name</label>
      <input
        ref={nameInputRef}
        type='text'
        id='name'
        onChange={nameInputChangeHandler}
        value={enteredName}
        onBlur={nameInputBlurHandler}
      />
      {nameInputIsValid && <p className="error-text">Name must not be empty<
    </div>
    <div className="form-actions">
      <button>Submit</button>
    </div>
  </form>
);
```

Complexity is 3 Everything is cool!

```
const nameInputBlurHandler = event => {
  setEnteredNameTouched(true);
  if (enteredName.trim() === '') {
    setEnteredNameIsValid(false);
    return;
  }
}
```

Disabling the button based on the state

```
return (
  <form onSubmit={formSubmissionHandler}>
    <div className={nameInputClasses}>
      <label htmlFor='name'>Your Name</label>
      <input
        type='text'
        id='name'
        onChange={nameInputChangeHandler}
        value={enteredName}
        onBlur={nameInputBlurHandler}
      />
      {nameInputIsValid && <p className="error-text">Name must not be empty<
    </div>
    <div className="form-actions">
      <button disabled={!formIsValid}>Submit</button>
    </div>
  </form>
);
```

## Refactoring the functionality

```
import { useState } from "react";

Complexity is 17 You must be kidding
const SimpleInput = (props) => { █

  const [enteredName, setEnteredName] = useState('');
  const [enteredNameTouched, setEnteredNameTouched] = useState(false);

  const enteredNameIsValid = enteredName.trim() !== '';
  const nameInputIsInvalid = !enteredNameIsValid && enteredNameTouched;

  const nameInputBlurHandler = event => {
    setEnteredNameTouched(true);
  }

  const nameInputChangeHandler = event => {
    setEnteredName(event.target.value);
  }

Complexity is 3 Everything is cool!
const formSubmissionHandler = event => { █
  event.preventDefault();
  setEnteredNameTouched(true)
  if (!enteredNameIsValid) {
    return;
  }
  setEnteredName('');
  setEnteredNameTouched(false)
}

const nameInputClasses = nameInputIsInvalid
  ? 'form-control invalid'
  : 'form-control';

return (
  <form onSubmit={formSubmissionHandler}>
    <div className={nameInputClasses}>
      <label htmlFor='name'>Your Name</label>
      <input
        type='text'
        id='name'
        onChange={nameInputChangeHandler}
        value={enteredName}
        onBlur={nameInputBlurHandler}
      />
      {nameInputIsInvalid && <p className="error-text">Name must not be empty</p>}
    </div>
    <div className="form-actions">
      <button>Submit</button>
    </div>
  </form>
)
```

## Custom Hook

```
Complexity is 6 It's time to do something...
const useInput = (validateValue) => { █

    const [enteredValue, setEnteredValue] = useState('');
    const [isTouched, setIsTouched] = useState(false);

    const valueIsValid = validateValue(enteredValue);
    const hasError = !valueIsValid && isTouched;

    const valueChangeHandler = event => {
        setEnteredValue(event.target.value);
    }

    const inputBlurHandler = event => {
        setIsTouched(true);
    }

    const reset = () => {
        setEnteredValue('')
        setIsTouched(false)
    } █

    return {
        value: enteredValue,
        isValid : valueIsValid,
        hasError,
        valueChangeHandler,
        inputBlurHandler,
        reset
    };
};

export default useInput;
```

## Using reducer in custom hook

```
import { useReducer } from "react";

const initialInputState = {
  value: '',
  isTouched: false
}
```

Complexity is 8 It's time to do something...

```
const inputStateReducer = (state, action) => {
  if (action.type === 'INPUT') {
    return {value: action.value, isTouched: state.isTouched};
  }
  if (action.type === 'BLUR') {
    return {value: state.value, isTouched: true};
  }
  if (action.type === 'RESET') {
    return {value: '', isTouched: false};
  }
  return initialInputState
}
```

Complexity is 6 It's time to do something...

```
const useInput = (validateValue) => {
  const [inputState, dispatch] = useReducer(inputStateReducer, initialInputState);

  const valueIsValid = validateValue(inputState.value);
  const hasError = !valueIsValid && inputState.isTouched;

  const valueChangeHandler = event => {
    dispatch({type: 'INPUT', value: event.target.value});
  }

  const inputBlurHandler = event => {
    dispatch({type: 'BLUR'});
  }

  const reset = () => {
    dispatch({type: 'RESET'});
  }

  return {
    value: inputState.value,
```

## Using Http in Food Order APP

Reusing the useHttp hook

```
import { useState , useCallback} from "react";

Complexity is 11 You must be kidding
const useHttp = () => { [REDACTED]

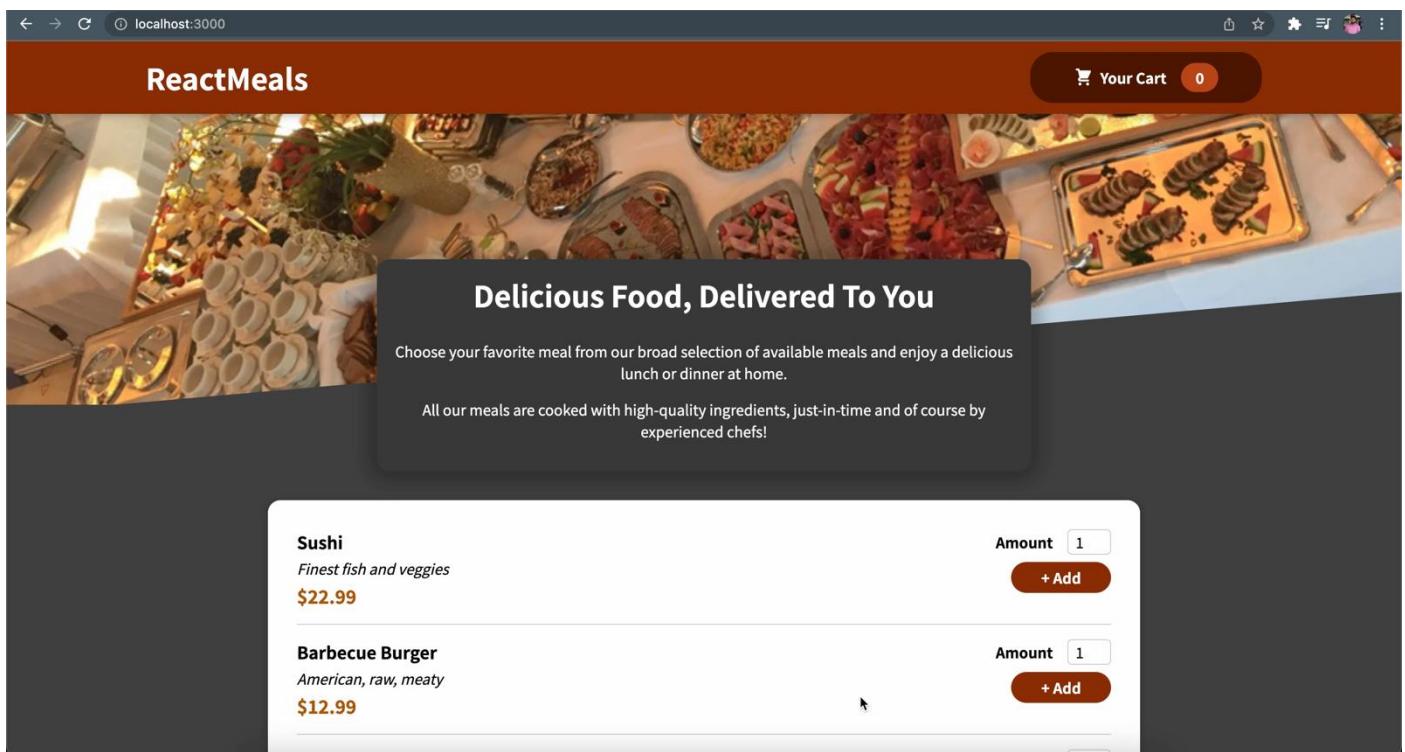
  const [isLoading, setIsLoading] = useState(false);
  const [error, setError] = useState(null);

  Complexity is 9 It's time to do something...
  const sendRequest = useCallback(async (requestConfig,applyData) => { [REDACTED]
    setIsLoading(true);
    setError(null);
    try {
      const response = await fetch(
        requestConfig.url,{ [REDACTED]
          method: requestConfig.method ? requestConfig.method: 'GET',
          headers: requestConfig.headers ? requestConfig.headers: {},
          body: requestConfig.body ? JSON.stringify(requestConfig.body): null
        });
      if (!response.ok) {
        throw new Error('Request failed!');
      }
      const data = await response.json();
      applyData(data)
    } catch (err) {
      setError(err.message || 'Something went wrong!');
    }
    setIsLoading(false);
  },[]);

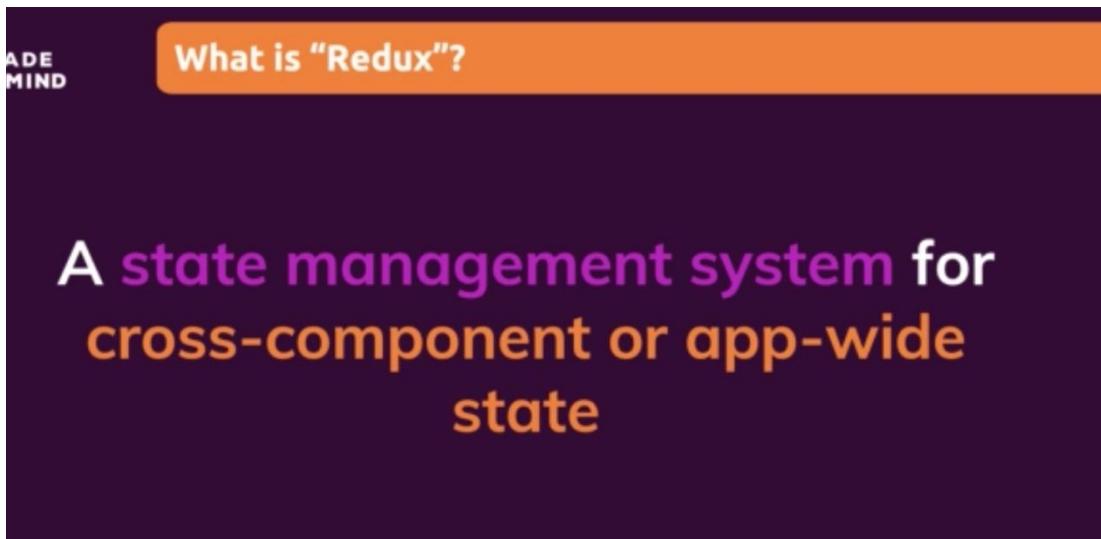
  return {
    isLoading,
    error,
    sendRequest
  };
};

export default useHttp;
```

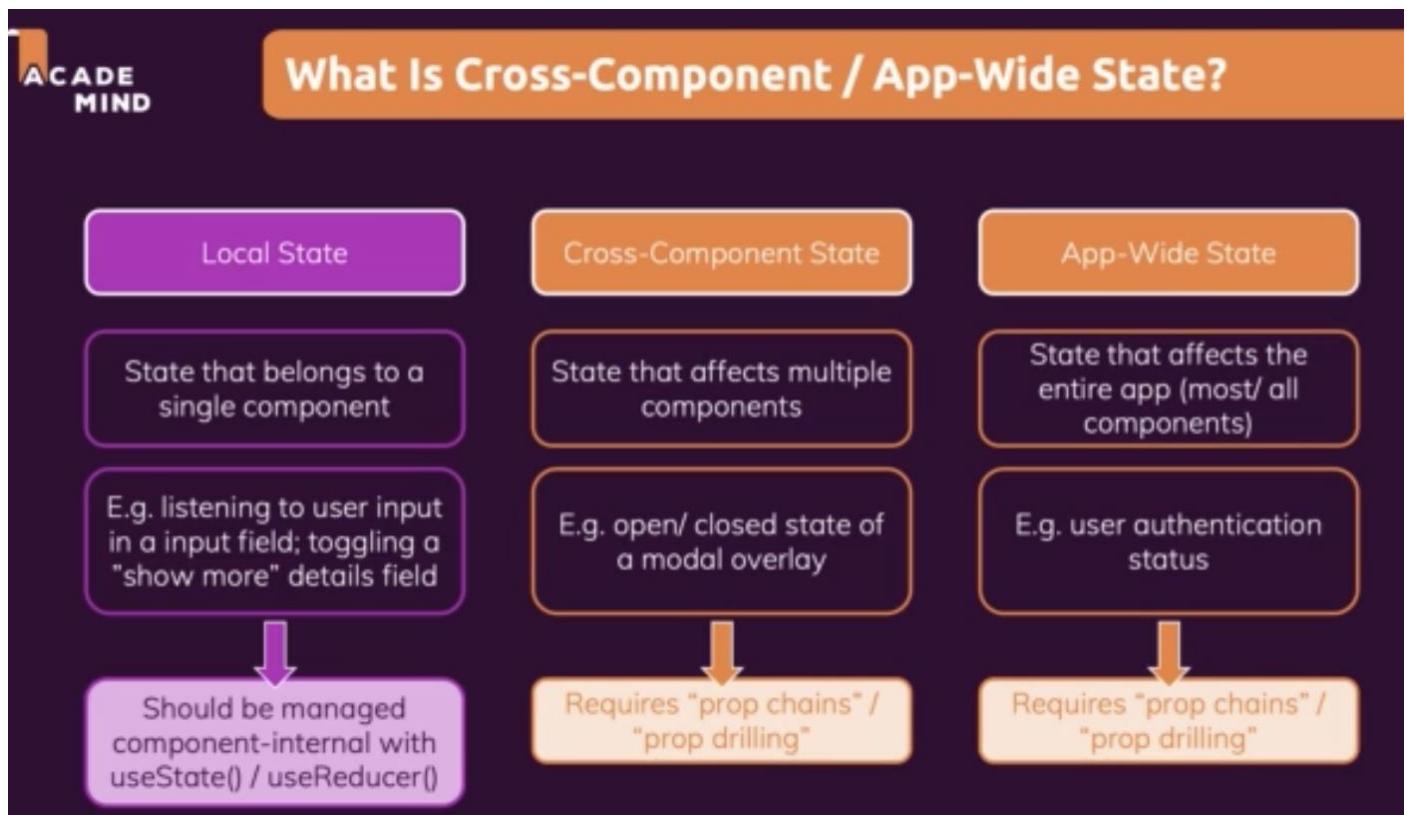
Demo



## Redux

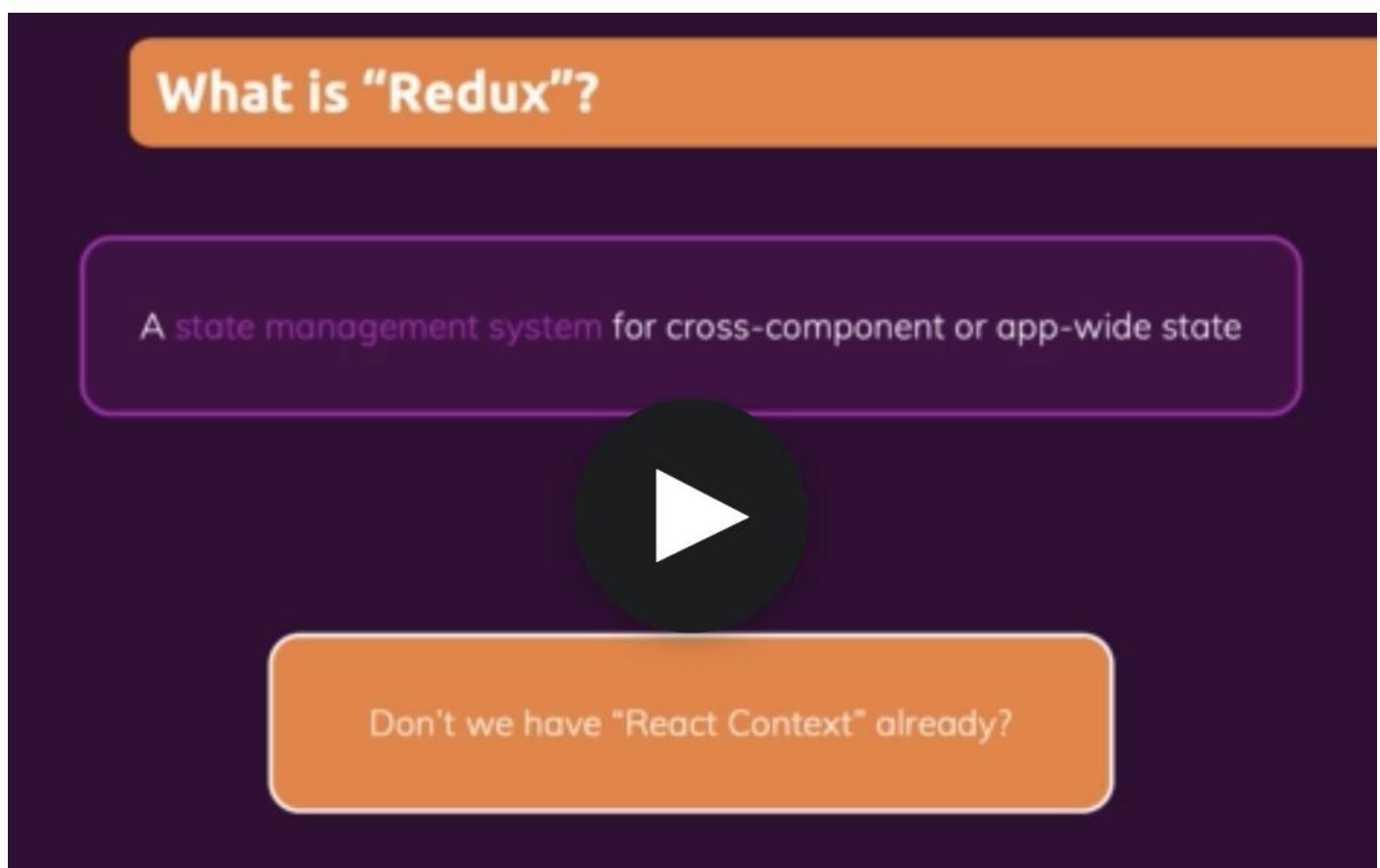


What is Cross-Component / App-Wide State?



Redux used to manage the cross-component state similar to react context.

Then Why do need a Redux when we already have Context Provider



## React Context – Complex Setup

```
● ● ●  
return (  
  <AuthContextProvider>  
    <ThemeContextProvider>  
      <UIInteractionContextProvider>  
        <MultiStepFormContextProvider>  
          <UserRegistration />  
        </MultiStepFormContextProvider>  
      </UIInteractionContextProvider>  
    </ThemeContextProvider>  
  </AuthContextProvider>  
)
```

```
function AllContextProvider() {  
  const [isAuth, setIsAuth] = useState(false);  
  const [isEvaluatingAuth, setIsEvaluatingAuth] = useState(false);  
  const [activeTheme, setActiveTheme] = useState('default');  
  const [ ... ] = useState(...);  
  
  function loginHandler(email, password) { ... };  
  
  function signupHandler(email, password) { ... };  
  
  function changeThemeHandler(newTheme) { ... };  
  
  ...  
  
  return (  
    <AllContext.Provider>
```

Context provider become large on large enterprise application.

## React Context – Performance



sebmarkbage commented on 18 Dec 2018

Member



...

My personal summary is that new context is ready to be used for low frequency unlikely updates (like locale/theme). It's also good to use it in the same way as old context was used. i.e. for static values and then propagate updates through subscriptions. It's not ready to be used as a replacement for all Flux-like state propagation.

55 4

Redux is flux like state propagation.

## React Context – Potential Disadvantages

Complex Setup / Management

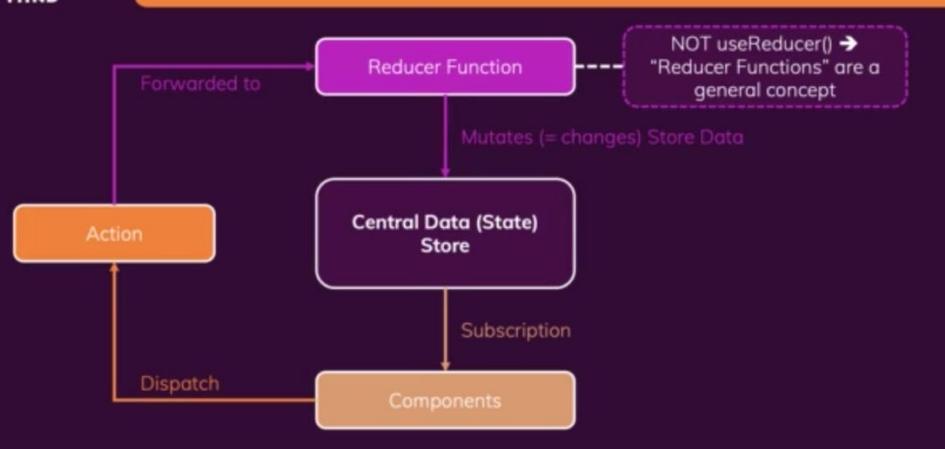
Performance

In more complex apps, managing React Context can lead to deeply nested JSX code and / or huge "Context Provider" components

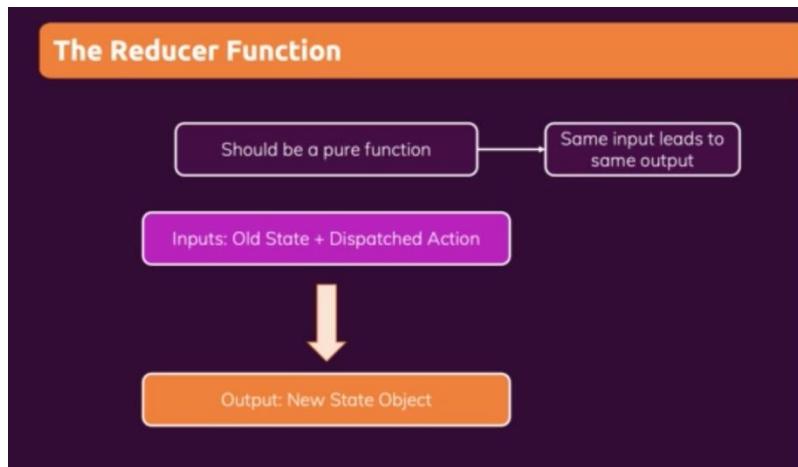
React Context is not optimized for high-frequency state changes

How Redux Works?

### Core Redux Concepts



## Redux Core Concept



## Redux Demo

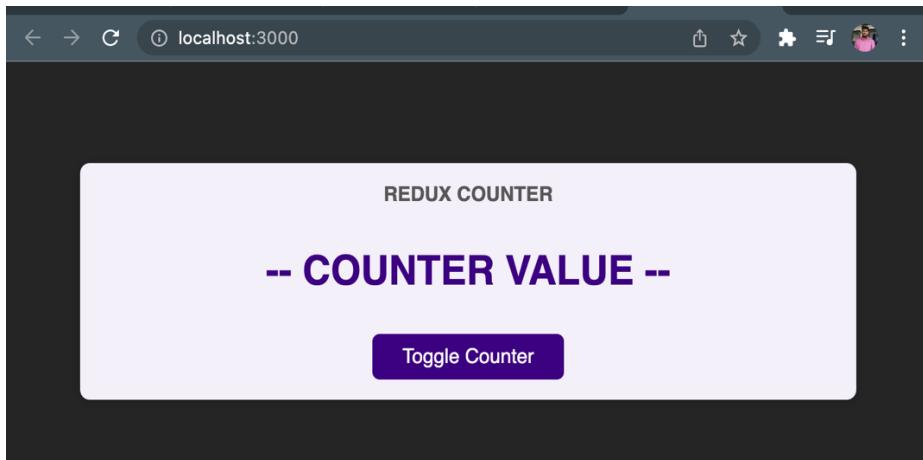
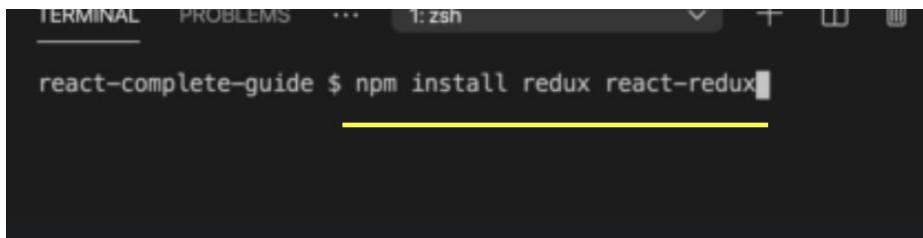
```
JS redux-demo.js X  TS package.json
16-redux-demo > JS redux-demo.js > ...
1  const redux = require('redux')
2
3  Complexity is 5 Everything is cool!
4  const counterReducer = (state = { counter: 0 }, action) => {
5      if (action.type === 'increment') {
6          return {
7              counter: state.counter + 1,
8          };
9      }
10     if (action.type === 'decrement') {
11         return {
12             counter: state.counter - 1,
13         };
14     };
15
16     const store = redux.createStore(counterReducer);
17     // console.log(store.getState());
18
19     const counterSubscriber = () => {
20         const latestState = store.getState();
21         console.log(latestState);
22     };
23
24     store.subscribe(counterSubscriber);
25
26     store.dispatch({type: 'increment'});
27     store.dispatch({type: 'increment'});
28     store.dispatch({type: 'decrement'});
```

PROBLEMS OUTPUT TERMINAL

TERMINAL ZSH - 16-REDUX-DE

```
→ 16-redux-demo node redux-demo.js
{ counter: 1 }
{ counter: 2 }
{ counter: 1 }
→ 16-redux-demo
```

Required redux package for react project



Registering the redux store in react app

```
'-redux-starter-project > src > js index.js
1  import React from 'react';
2  import ReactDOM from 'react-dom';
3  import { Provider } from "react-redux";
4  import './index.css';
5  import App from './App';
6  import store from './store/index';
7
8  ReactDOM.render(
9    <Provider store={store}>
10      <App />
11    </Provider>,
12    document.getElementById("root")
13  );
14  |
```

```

redux-starter-project > src > store > index.js > ...
import { createStore } from 'redux';

Complexity is 5 Everything is cool!
const counterReducer = (state = { counter: 0 }, action) => {
  if (action.type === 'increment') {
    return {
      counter: state.counter + 1
    };
  }
  if (action.type === 'decrement') {
    return {
      counter: state.counter - 1
    };
  }
};

const store = createStore(counterReducer);

export default store;

```

Use the redux store in react component

```

import classes from './Counter.module.css';
import { useSelector, useDispatch } from "react-redux";

Complexity is 13 You must be kidding
const Counter = () => {
  const dispatch = useDispatch(); —
  const counter = useSelector(state => state.counter); —
  const toggleCounterHandler = () => {};
  const incrementHandler = () => {
    dispatch({type: 'increment'}) —
  }
  const decrementHandler = () => {
    dispatch({type: 'decrement'}) —
  }

  return (
    <main className={classes.counter}>
      <h1>Redux Counter</h1>
      <div className={classes.value}>{counter}</div>
      <div>
        <button onClick={incrementHandler}>Increment</button>
        <button onClick={decrementHandler}>Decrement</button>
      </div>
      <button onClick={toggleCounterHandler}>Toggle Counter</button>
    </main>
  );
};

export default Counter;

```

Use the redux store in react class based component

```
Complexity is 8 It's time to do something...
class Counter extends Component { █

  incrementHandler () {
    this.props.increment();
  }

  decrementHandler() {
    this.props.decrement();
  }

  toggleCounterHandler() {
  }

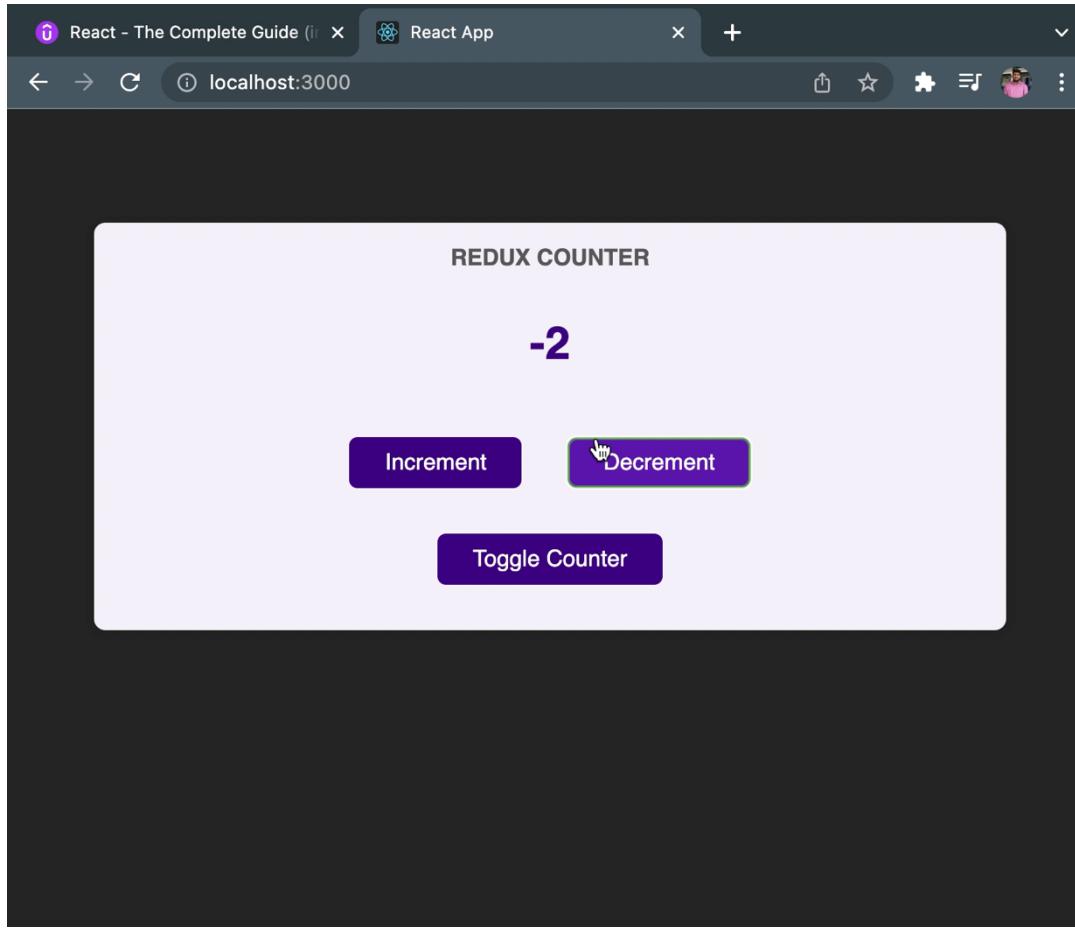
Complexity is 9 It's time to do something...
render() { █
  return (
    <main className={classes.counter}>
      <h1>Redux Counter</h1>
      <div className={classes.value}>{this.props.counter}</div>
      <div>
        <button onClick={this.incrementHandler.bind(this)}>Increment</button>
        <button onClick={this.decrementHandler.bind(this)}>Decrement</button>
      </div>
      <button onClick={this.toggleCounterHandler}>Toggle Counter</button>
    </main>
  );
}

const mapStateToProps = state => {
  return {
    counter: state.counter
  }
}

Complexity is 4 Everything is cool!
const mapDispatchToProps = dispatch => { █
  return {
    increment: ()=> dispatch({type:'increment'}),
    decrement: () => dispatch({type:'decrement'})
  }
}

export default connect(mapStateToProps, mapDispatchToProps)(Counter);
```

## Counter App Demo



## Passing data via dispatch action

```
Complexity is 15 You must be kidding
const Counter = () => { 
  const dispatch = useDispatch();
  const counter = useSelector(state => state.counter);

  const toggleCounterHandler = () => {};

  const incrementHandler = () => {
    dispatch({type: 'increment'});
  }

  const increaseHandler = () => {
    dispatch({ type: "increase", value: 5 });
  }

  const decrementHandler = () => {
    dispatch({type: 'decrement'});
  }

  return (
    <main className={classes.counter}>
      <h1>Redux Counter</h1>
      <div className={classes.value}>{counter}</div>
      <div>
        <button onClick={incrementHandler}>Increment</button>
        <button onClick={increaseHandler}>Increase by 5</button>
        <button onClick={decrementHandler}>Decrement</button>
      </div>
      <button onClick={toggleCounterHandler}>Toggle Counter</button>
    </main>
  );
};
```

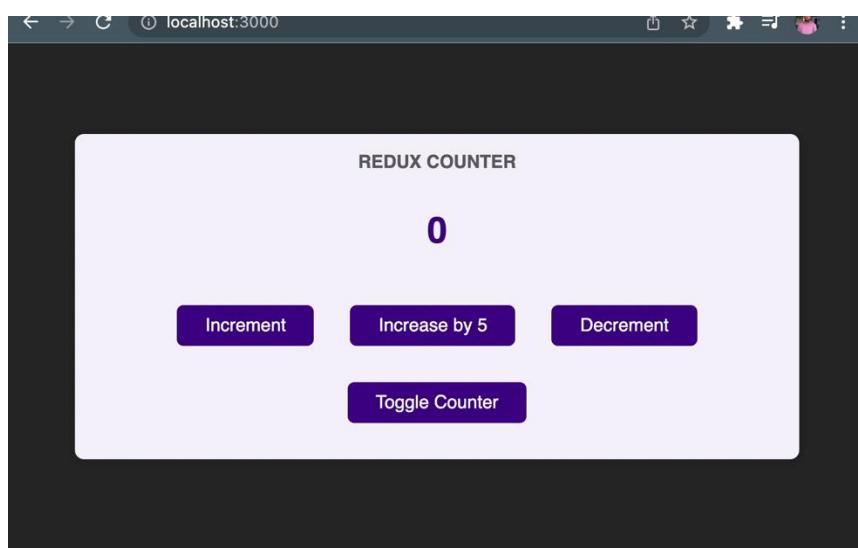
```
Complexity is 8 It's time to do something...
const counterReducer = (state = { counter: 0 }, action) => {
  if (action.type === 'increment') {
    return {
      counter: state.counter + 1
    };
  }
  if (action.type === 'increase') {
    return {
      counter: state.counter + action.value
    };
  }
  if (action.type === 'decrement') {
    return {
      counter: state.counter - 1
    };
  }
  return {
    counter: state.counter
  };
};

const store = createStore(counterReducer);

export default store;
```



Working with multiple state in redux



```
import {createStore} from 'redux';

const initialState = { counter: 0, showCounter: true }

Complexity is 10 It's time to do something...
const counterReducer = (state = initialState, action) => {
    if (action.type === 'increment') {
        return {
            counter: state.counter + 1,
            showCounter: state.showCounter
        };
    }
    if (action.type === 'increase') {
        return {
            counter: state.counter + action.value,
            showCounter: state.showCounter
        };
    }
    if (action.type === 'decrement') {
        return {
            counter: state.counter - 1,
            showCounter: state.showCounter
        };
    }
    if (action.type === 'toggle') {
        return {
            counter: state.counter,
            showCounter: !state.showCounter
        };
    }
    return state;
};

const store = createStore(counterReducer);

export default store;
```

```
import classes from './Counter.module.css';
import { useSelector, useDispatch } from "react-redux";

Complexity is 17 You must be kidding
const Counter = () => {
  const dispatch = useDispatch();
  const counter = useSelector(state => state.counter);
  const show = useSelector(state => state.showCounter); }

  const toggleCounterHandler = () => {
    dispatch({type: 'toggle'})
  };

  const incrementHandler = () => {
    dispatch({type: 'increment'})
  }

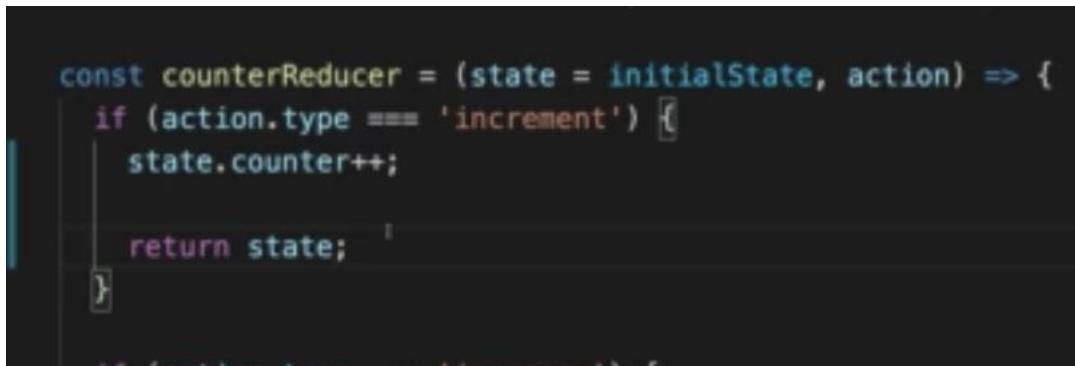
  const increaseHandler = () => {
    dispatch({ type: "increase", value: 5 });
  }

  const decrementHandler = () => {
    dispatch({type: 'decrement'})
  }

  return (
    <main className={classes.counter}>
      <h1>Redux Counter</h1>
      {show && <div className={classes.value}>{counter}</div>}
      <div>
        <button onClick={incrementHandler}>Increment</button>
        <button onClick={increaseHandler}>Increase by 5</button>
        <button onClick={decrementHandler}>Decrement</button>
      </div>
      <button onClick={toggleCounterHandler}>Toggle Counter</button>
    </main>
  );
};

export default Counter;
```

Never mutate the state when working with redux like below. Always overwrite the state.

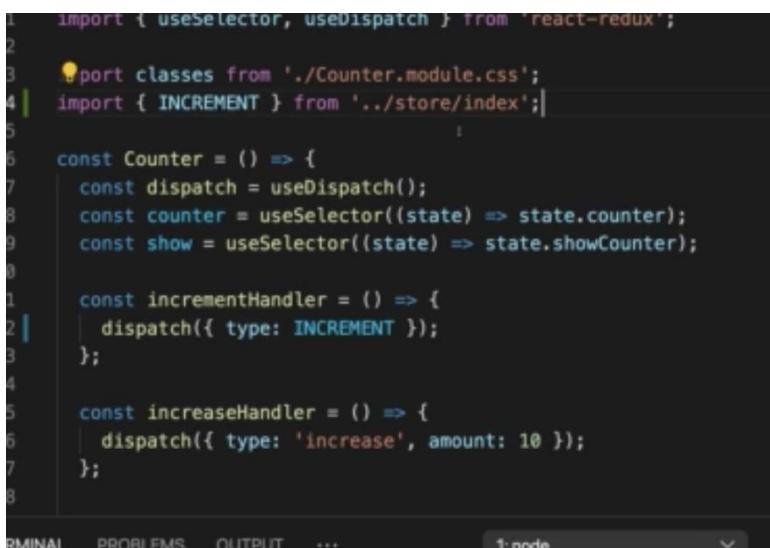


```
const counterReducer = (state = initialState, action) => {
  if (action.type === 'increment') {
    state.counter++;

    return state;
  }
}
```

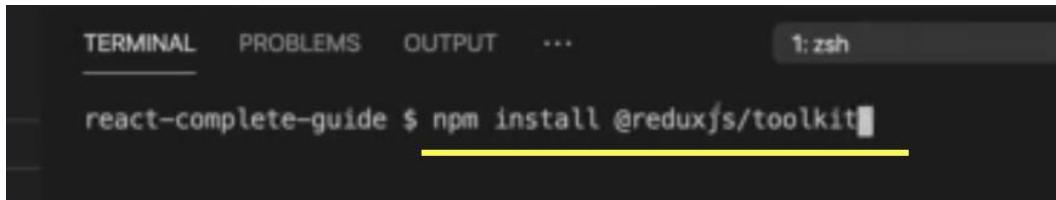
## Redux Challenges & Introducing Redux Toolkit

Using constant for action type



```
1 import { useSelector, useDispatch } from 'react-redux';
2
3 import classes from './Counter.module.css';
4 import { INCREMENT } from '../store/index';
5
6 const Counter = () => {
7   const dispatch = useDispatch();
8   const counter = useSelector((state) => state.counter);
9   const show = useSelector((state) => state.showCounter);
10
11   const incrementHandler = () => {
12     dispatch({ type: INCREMENT });
13   };
14
15   const increaseHandler = () => {
16     dispatch({ type: 'increase', amount: 10 });
17   };
18 }
```

TERMINAL PROBLEMS OUTPUT ... 1:node



```
TERMINAL PROBLEMS OUTPUT ... 1:zsh
react-complete-guide $ npm install @reduxjs/toolkit
```

## Redux Toolkit Example

```
import { createSlice ,configureStore} from '@reduxjs/toolkit';

const initialState = { counter: 0, showCounter: true }

const counterSlice = createSlice({
  name: "counter",
  initialState,
  reducers: {
    increment(state) {
      state.counter++;
    },
    decrement(state) {
      state.counter--;
    },
    increase(state, action) {
      state.counter = state.counter + action.payload;
    },
    toggleCounter(state) {
      state.showCounter = !state.showCounter;
    },
  },
});

const store = configureStore({
  reducer: counterSlice.reducer
});

export const counterAction = counterSlice.actions;

export default store;
```

```
nx start project > src > components > Counter.js > ...
import classes from './Counter.module.css';
import { useSelector, useDispatch } from "react-redux";
import { counterAction } from '../store';
|
Complexity is 17 You must be kidding
const Counter = () => {
  const dispatch = useDispatch();
  const counter = useSelector(state => state.counter);
  const show = useSelector(state => state.showCounter);

  const toggleCounterHandler = () => {
    dispatch(counterAction.toggleCounter())
  };

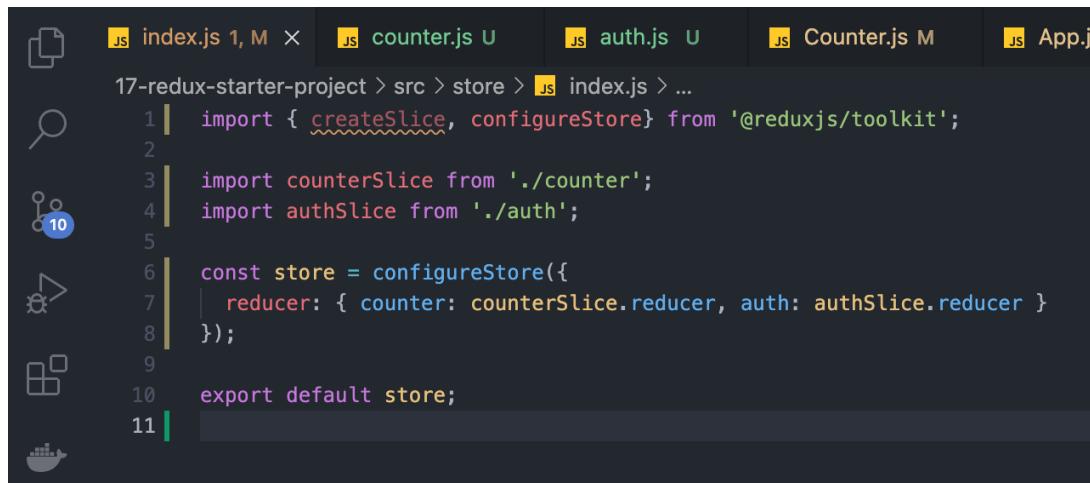
  const incrementHandler = () => {
    dispatch(counterAction.increment())
  }

  const increaseHandler = () => {
    dispatch(counterAction.increase(5));
  }

  const decrementHandler = () => {
    dispatch(counterAction.decrement())
  }

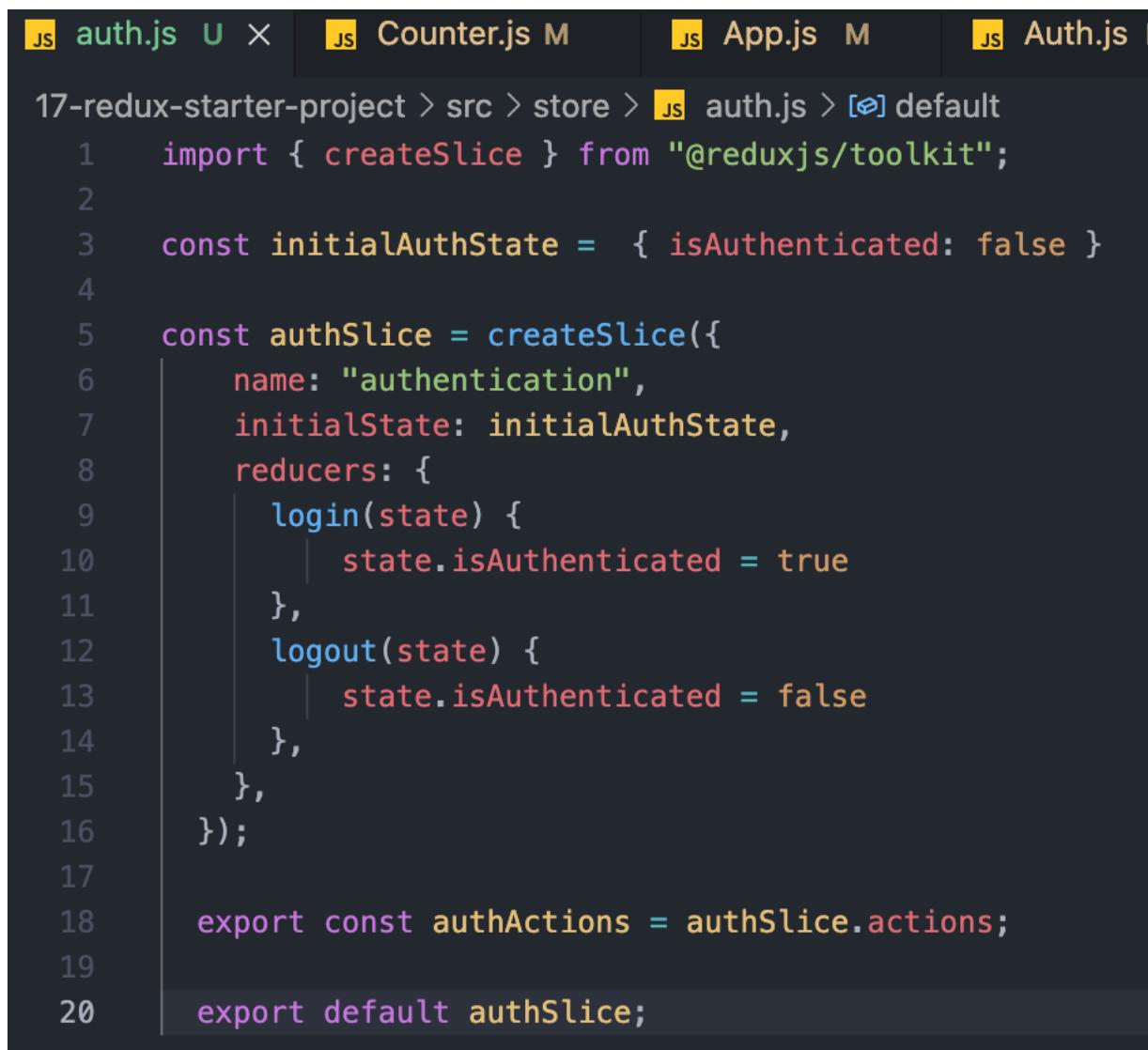
  return (
    <main className={classes.counter}>
      <h1>Redux Counter</h1>
      {show && <div className={classes.value}>{counter}</div>}
      <div>
        <button onClick={incrementHandler}>Increment</button>
        <button onClick={increaseHandler}>Increase by 5</button>
      </div>
    </main>
  );
}
```

## Using Multiple States



A screenshot of a code editor showing the `index.js` file from a Redux starter project. The file imports `createSlice` and `configureStore` from `@reduxjs/toolkit`. It then imports `counterSlice` and `authSlice` from their respective files. A `store` is configured using `configureStore` with a reducer object containing `counter` and `auth` keys, each pointing to its respective reducer. Finally, the `store` is exported.

```
17-redux-starter-project > src > store > index.js > ...
1 | import { createSlice, configureStore } from '@reduxjs/toolkit';
2 |
3 | import counterSlice from './counter';
4 | import authSlice from './auth';
5 |
6 | const store = configureStore({
7 |   reducer: { counter: counterSlice.reducer, auth: authSlice.reducer }
8 | });
9 |
10| export default store;
11|
```

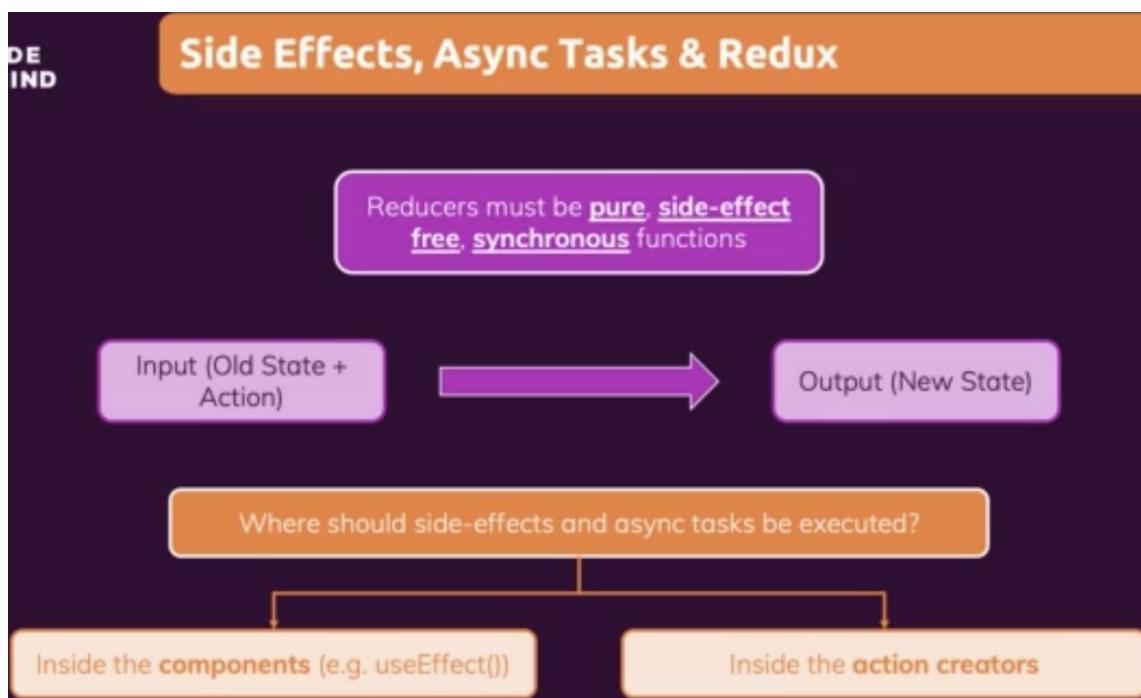


A screenshot of a code editor showing the `auth.js` file from a Redux starter project. The file imports `createSlice` from `@reduxjs/toolkit`. It defines an initial state object `initialAuthState` with a `isAuthenticated` key set to `false`. A `authSlice` is created using `createSlice`, specifying a name of `"authentication"`, an initialState of `initialAuthState`, and reducers for `login` and `logout` actions. The `login` reducer sets `state.isAuthenticated` to `true`, and the `logout` reducer sets it back to `false`. The slice is then exported along with its actions.

```
17-redux-starter-project > src > store > auth.js > [✖] default
1 | import { createSlice } from "@reduxjs/toolkit";
2 |
3 | const initialAuthState = { isAuthenticated: false }
4 |
5 | const authSlice = createSlice({
6 |   name: "authentication",
7 |   initialState: initialAuthState,
8 |   reducers: {
9 |     login(state) {
10|       state.isAuthenticated = true
11|     },
12|     logout(state) {
13|       state.isAuthenticated = false
14|     },
15|   },
16| });
17 |
18| export const authActions = authSlice.actions;
19 |
20| export default authSlice;
```

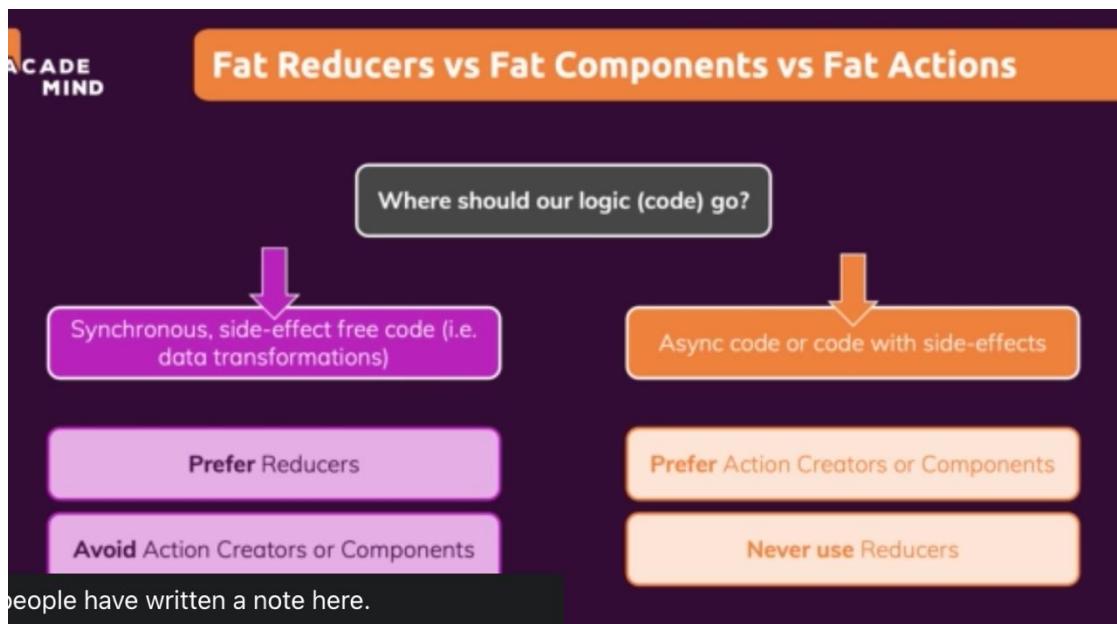
```
17-redux-starter-project > src > components > Header.js > ...
3 import { authActions } from '../store/auth.js';
4
5 Complexity is 15 You must be kidding
6 const Header = () => {
7
8     const dispatch = useDispatch();
9
10    const isAuthenticated = useSelector(state => state.auth.isAuthenticated)
11
12    const logoutHandler = (event) => {
13        dispatch(authActions.logout());
14    }
15    return (
16        <header className={classes.header}>
17            <h1>Redux Auth</h1>
18            {isAuthenticated && (
19                <nav>
20                    <ul>
21                        <li>
22                            <a href='/'>My Products</a>
23                        </li>
24                        <li>
25                            <a href='/'>My Sales</a>
26                        </li>
27                        <li>
28                            <button onClick={logoutHandler}>Logout</button>
29                        </li>
30                    </ul>
31                </nav>
32            )} 
33        </header>
34    );
}
```

## Advanced Redux Concept



```
npm install @reduxjs/toolkit  
npm install react-redux
```

## Using asynchronous code with Redux



```
import Cart from './components/Cart/Cart';
import Layout from './components/Layout/Layout';
import Products from './components/Shop/Products';
import {useSelector} from 'react-redux';
import { useEffect } from 'react';

Complexity is 9 It's time to do something...
function App() { }

  const cartIsVisible = useSelector(state => state.ui.cartIsVisible);
  const cart = useSelector((state)=> state.cart);

  useEffect(() => {
    fetch('https://react-http-5c577-default-rtdb.firebaseio.com/cart.json',{ 
  }, [cart]);

  return (
    <Layout>
      {cartIsVisible && <Cart />}
      <Products />
    </Layout>
  );
}

export default App;
```

Adding the notification.

```
const dispatch = useDispatch();
const cartIsVisible = useSelector(state => state.ui.cartIsVisible);
const cart = useSelector((state)=> state.cart);
const notification = useSelector((state) => state.ui.notification);

Complexity is 7 It's time to do something...
useEffect(() => { █
  Complexity is 3 Everything is cool!
  const sendCartData = async () => { █
    dispatch(
      uiActions.showNotification({
        status: "Pending",
        title: "Sending...",
        message: "Sending cart data!",
      })
    );
    const response = await fetch(
      "https://react-http-5c577-default-firebase.firebaseio.com/cart.json",
      { method: "PUT", body: JSON.stringify(cart) }
    );
    if (!response.ok) {
      throw new Error();
    }
    dispatch(
      uiActions.showNotification({
        status: "success",
        title: "Success!",
        message: "Sent cart data successfull!",
      })
    );
  };
}

if (isInitial) {
  isInitial = false;
  return;
}

sendCartData().catch((error)=>{
  dispatch(uiActions.showNotification({
    status: 'error',
    title: 'Error!',
    message: 'Sent cart data failed!'
  }));
});

}, [cart,dispatch]);
```

## What is a “Thunk”?

A function that delays an action until later

An action creator function that does NOT return the action itself but another function which eventually returns the action

```
import { uiActions } from "./store/ui-slice";
import Notification from './components/UI/Notification';
import { sendCartData } from "./store/cart-slice";

let isInitial = true;

Complexity is 15 You must be kidding
function App() { █

  const dispatch = useDispatch();
  const cartIsVisible = useSelector(state => state.ui.cartIsVisible);
  const cart = useSelector((state)=> state.cart);
  const notification = useSelector((state) => state.ui.notification);

  Complexity is 3 Everything is cool!
  useEffect(() => { █
    if (isInitial) {
      isInitial = false;
      return;
    }
    dispatch(sendCartData(cart));
  }, [cart,dispatch]); █

  return (
    <Fragment>...
    </Fragment>
  );
}

export default App;
```

Complexity is 8 It's time to do something...

```
export const sendCartData = (cartData) => {
```

Complexity is 6 It's time to do something...

```
    return async (dispatch)=> {
```

```
        dispatch(
```

```
            uiActions.showNotification({
```

status: "Pending",  
 title: "Sending...",  
 message: "Sending cart data!",

```
            })
```

```
    );
```

Complexity is 3 Everything is cool!

```
    const sendRequest = async() => {
```

```
        const response = await fetch(
```

```
            "https://react-http-5c577-default-firebaseio.com/cart.json"
```

```
            { method: "PUT", body: JSON.stringify(cartData) }
```

```
        );
```

```
        if (!response.ok) {
```

```
            throw new Error();
```

```
        }
```

```
}
```

```
try {
```

```
    await sendRequest();
```

```
    dispatch(
```

```
        uiActions.showNotification({
```

status: "success",  
 title: "Success!",  
 message: "Sent cart data successfull!",

```
        })
```

```
    );
```

```
} catch (error) {
```

```
    dispatch(
```

```
        uiActions.showNotification({
```

status: "error",  
 title: "Error!",  
 message: "Sent cart data failed!",

```
        })
```

```
    );
```

```
};
```

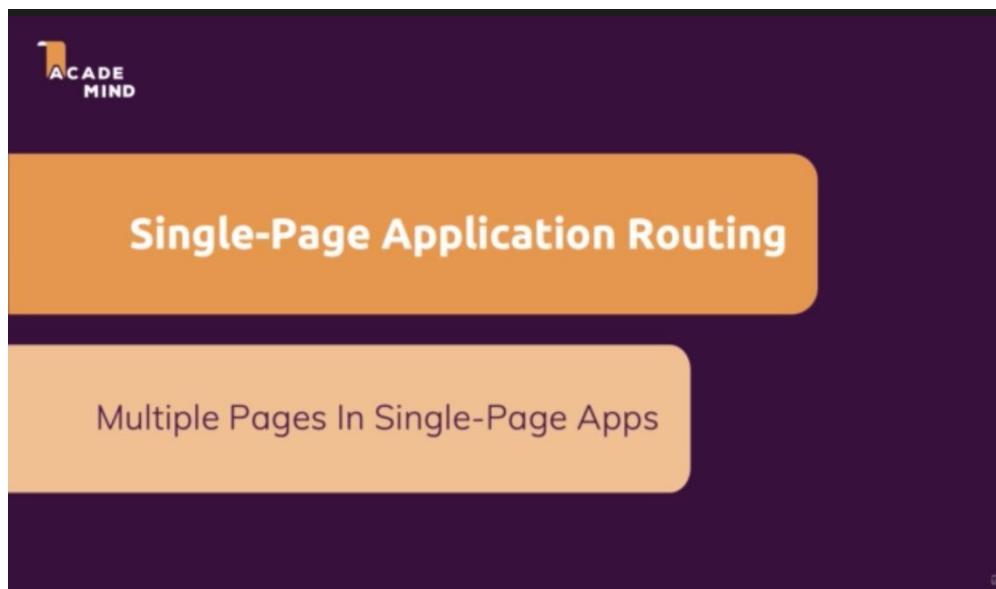
```
};
```



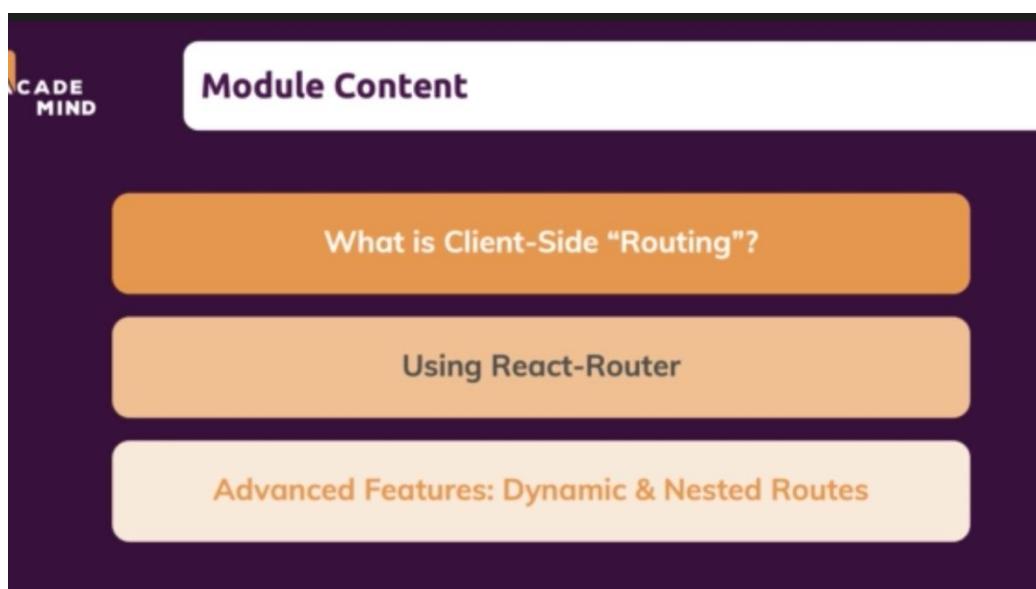
# Redux DevTools

Offered by: Redux DevTools

★★★★★ 545 | [Developer Tools](#) | 1,000,000+ users

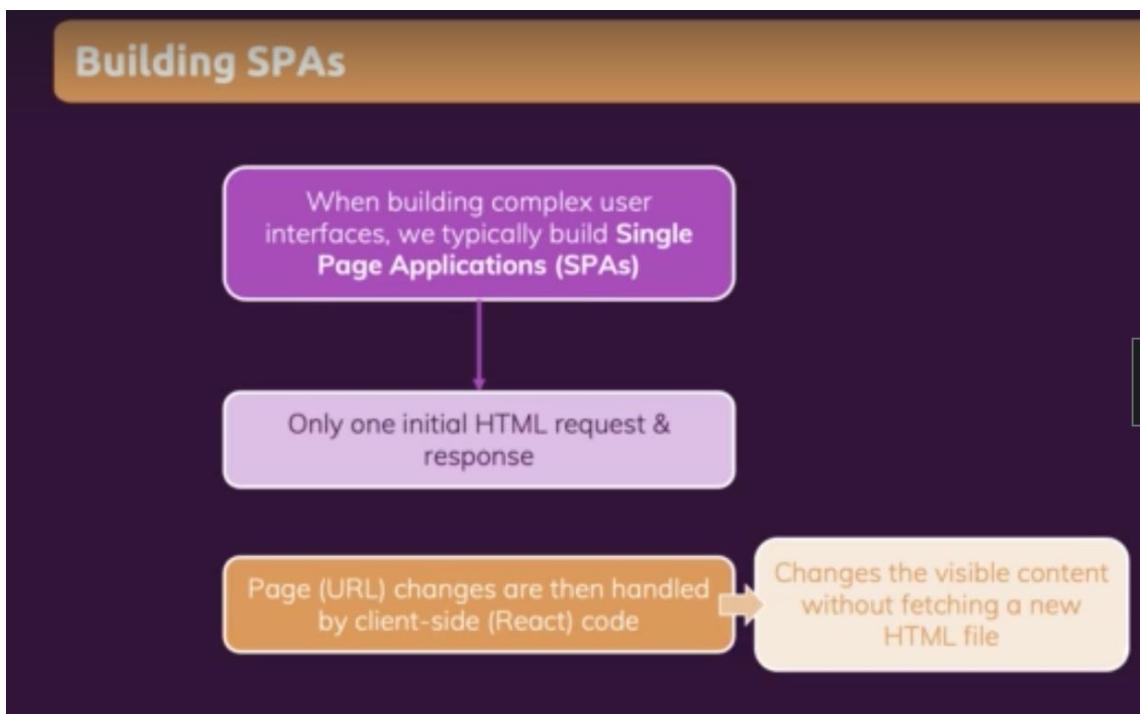
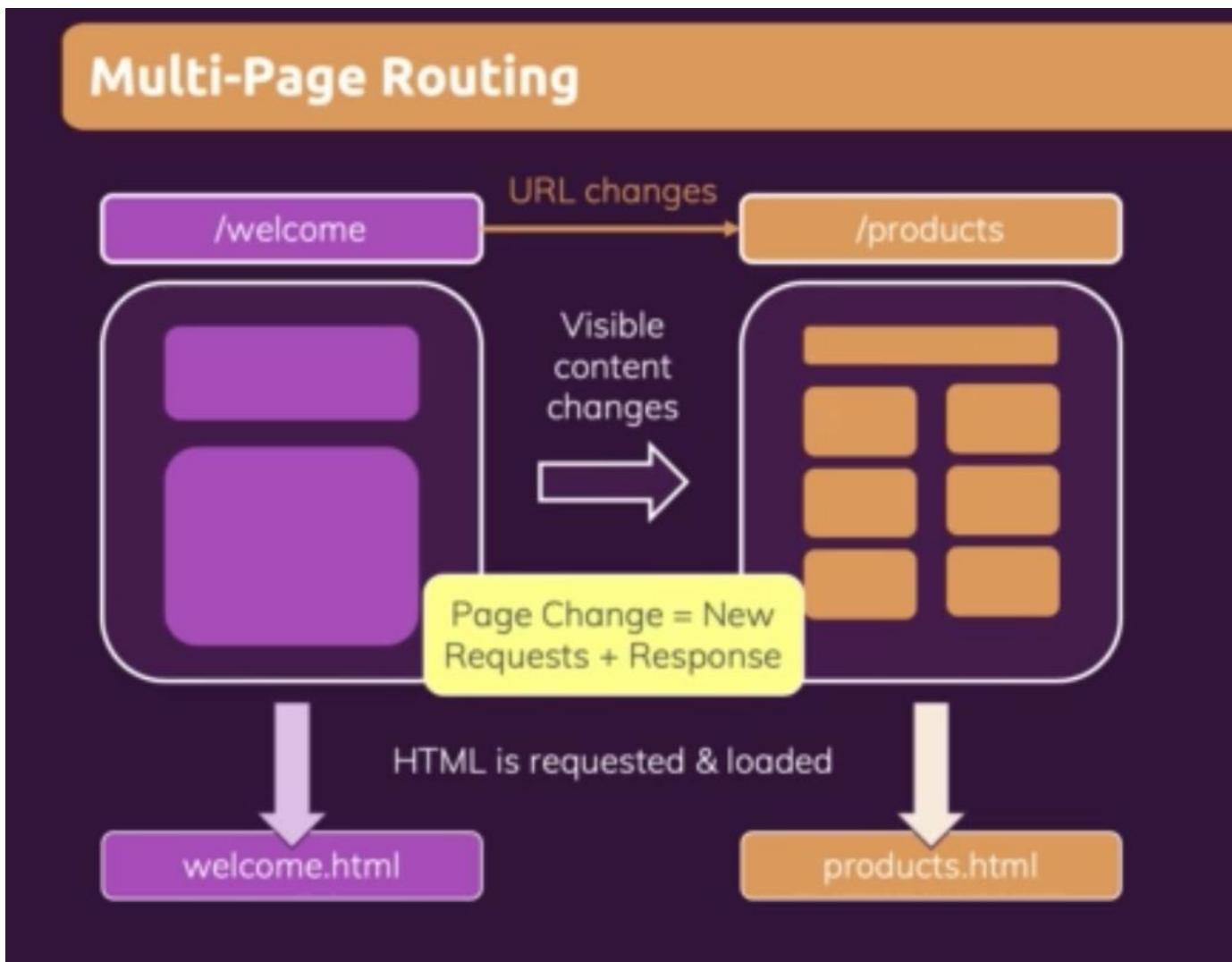


The thumbnail for a course on Single-Page Application Routing. It features a dark purple header with the Academind logo. Below this is an orange callout box containing the title "Single-Page Application Routing". The main content area is a light orange box containing the text "Multiple Pages In Single-Page Apps". At the bottom right of the main box, there is some very small, illegible text.



The thumbnail for a module content section of a React Router course. It has a dark purple header with the Academind logo. Below it is a white callout box containing the title "Module Content". Below this are three orange callout boxes, each containing a topic: "What is Client-Side “Routing”?", "Using React-Router", and "Advanced Features: Dynamic & Nested Routes".

Why routing?



## Router Basic

```
import { Route } from "react-router-dom";
import Products from "./components/Products";
import Welcome from "./components/Welcome";

Complexity is 7 It's time to do something...
function App() {
  return (
    <div>
      <Route path="/welcome">
        <Welcome />
      </Route>
      <Route path="/products">
        <Products />
      </Route>
    </div>
  );
}

export default App;
```

```
react-router > src > index.js
▶ import ReactDOM from 'react-dom';
▶ import './index.css';
▶ import App from './App';
▶ import {BrowserRouter} from 'react-router-dom'
```

```
ReactDOM.render(
  <BrowserRouter>
    <App />
  </BrowserRouter>,
  document.getElementById("root")
);
```

## Using Link to navigate to Route

The screenshot shows the code for `MainHeader.js` on the left and a browser preview on the right.

`MainHeader.js` code:

```
import { Link } from "react-router-dom";

Complexity is 9 It's time to do something...
const MainHeader=()=>{
  return <header>
    <nav>
      <ul>
        <li>
          <Link to="/welcome">Welcome</Link>
        </li>
        <li>
          <Link to="/products">Products</Link>
        </li>
      </ul>
    </nav>
  </header>;
}

export default MainHeader;
```

Browser preview (localhost:3000/welcome):

- Welcome
- Products

## Welcome Page

Using the Link will prevent the page reload.

## Using NavLink to navigate to Route

The screenshot shows the code for `MainHeader.js` on the left and a browser preview on the right.

`MainHeader.js` code:

```
import { NavLink } from "react-router-dom";
import classes from './MainHeader.module.css'
Complexity is 9 It's time to do something...
const MainHeader=()=>{
  return (
    <header className={classes.header}>
      <nav>
        <ul>
          <li>
            <NavLink activeClassName={classes.active} to="/welcome">
              Welcome
            </NavLink>
          </li>
          <li>
            <NavLink activeClassName={classes.active} to="/products">
              Products
            </NavLink>
          </li>
        </ul>
      </nav>
    </header>
  );
}

export default MainHeader;
```

Browser preview (localhost:3000/welcome):

Header navigation bar with two items: "Welcome" and "Products". The "Welcome" link is underlined, indicating it is active.

## Welcome Page

NavLink is like Link. It helps to add a CSS class to active link.

## Route Param

```
import { Route } from "react-router-dom";
import MainHeader from "./components/MainHeader";
import ProductDetail from "./pages/ProductDetail";
import Products from "./pages/Products";
import Welcome from "./pages/Welcome";

Complexity is 11 You must be kidding
function App() { █
  return (
    <div>
      <MainHeader/>
      <main>
        <Route path="/welcome">
          | <Welcome/>
        </Route>
        <Route path="/products">
          | <Products/>
        </Route>
        <Route path="/product-detail/:productId">
          | <ProductDetail/> █
        </Route>
      </main>
    </div>
  );
}

export default App;
```

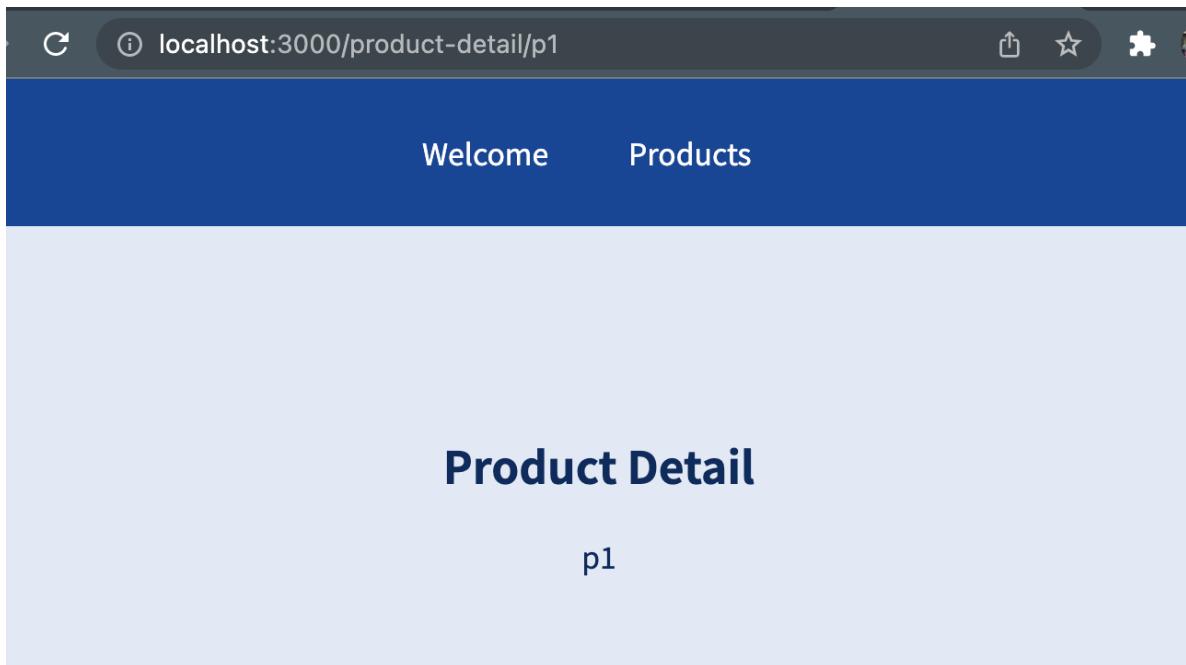
```
import { useParams } from "react-router-dom";
```

Complexity is 5 Everything is cool!

```
const ProductDetail = () => { █
  const params = useParams();

  return (
    <section>
      <h1>Product Detail</h1>
      <p>{params.productId}</p>
    </section> █
  );
};

export default ProductDetail;
```

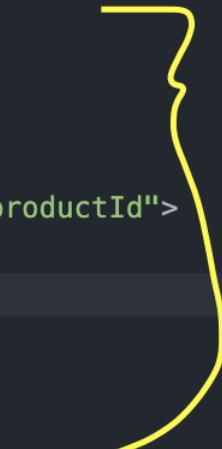


Using “Switch” and “exact” for configuring routes

Switch considers the <Route> orders

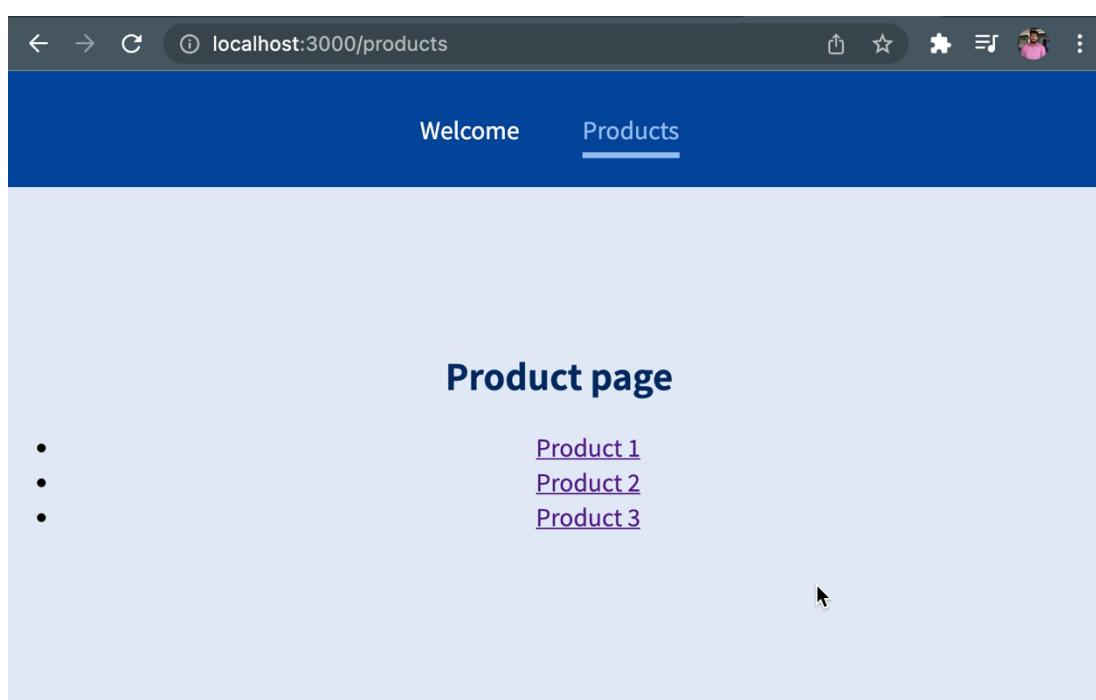
Whichever <Route> matching first only that will be displayed.

```
Complexity is 12 You must be kidding
function App() {
  return (
    <div>
      <MainHeader />
      <main>
        <Switch>
          <Route path="/welcome">
            <Welcome />
          </Route>
          <Route path="/products/:productId">
            <ProductDetail />
          </Route>
          <Route path="/products">
            <Products />
          </Route>
        </Switch>
      </main>
    </div>
  );
}
```



If we don't want the <Switch> to consider the order, we can use "exact".

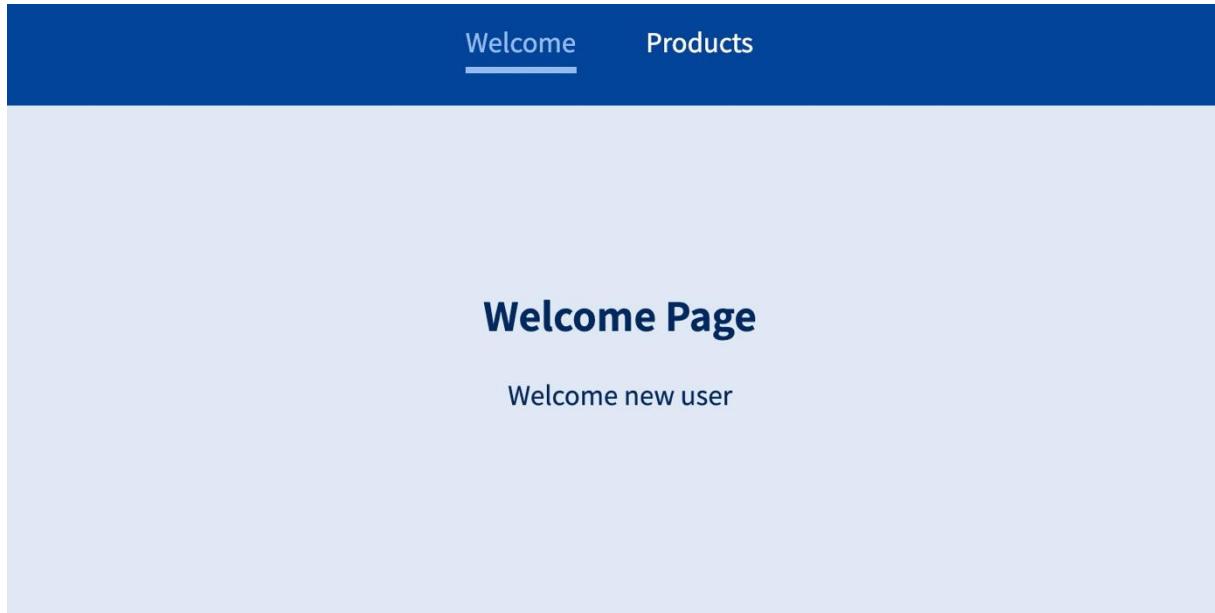
```
Complexity is 12 You must be kidding
function App() { 
  return (
    <div>
      <MainHeader />
      <main>
        <Switch>
          <Route path="/welcome">
            <Welcome />
          </Route>
          <Route path="/products" exact>
            <Products />
          </Route>
          <Route path="/products/:productId">
            <ProductDetail />
          </Route>
        </Switch>
      </main>
    </div>
  );
}
```



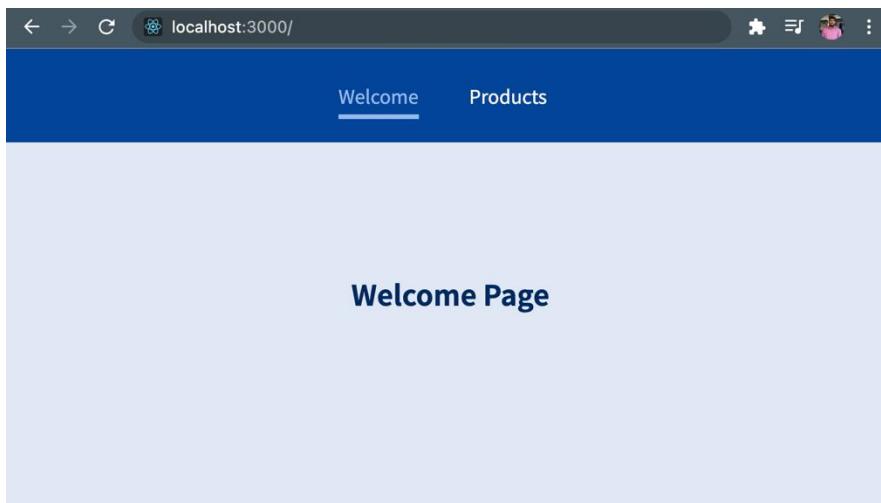
## Working with Nested Routes

```
import { Route } from "react-router-dom";
Complexity is 6 It's time to do something...
const Welcome = () => {
  return (
    <section>
      <h1>Welcome Page</h1>
      <Route path="/welcome/new-user">
        <p>Welcome new user</p>
      </Route>
    </section>
  );
}

export default Welcome;
```



## Using Redirect



```
import { Redirect, Route, Switch } from "react-router-dom";
import MainHeader from "./components/MainHeader";
import ProductDetail from "./pages/ProductDetail";
import Products from "./pages/Products";
import Welcome from "./pages/Welcome";

Complexity is 14 You must be kidding
function App() {   
  return (
    <div>
      <MainHeader />
      <main>
        <Switch>
          <Route path="/" exact>
            <Redirect to="/welcome"/>
          </Route>
          <Route path="/welcome">
            <Welcome />
          </Route>
          <Route path="/products" exact>
            <Products />
          </Route>
          <Route path="/products/:productId">
            <ProductDetail />
          </Route>
        </Switch>
      </main>
    </div>
  );
}

export default App;
```

## Route for handling unknown URL paths

```
import QuoteDetail from "./pages/quotedetails";
import AllQuotes from "./pages/allquotes";
import NewQuote from "./pages/newquote";

Complexity is 14 You must be kidding
function App() {   
  return (
    <Layout>
      <Switch>
        <Route path="/" exact>
          <Redirect to="/quotes" />
        </Route>
        <Route path="/quotes" exact>
          <AllQuotes />
        </Route>
        <Route path="/quotes/:quoteId">
          <QuoteDetail />
        </Route>
        <Route path="/new-quote">
          <NewQuote />
        </Route>
        <Route path="*">
          <NotFound />
        </Route>
      </Switch>
    </Layout>
  );
}

export default App;
```

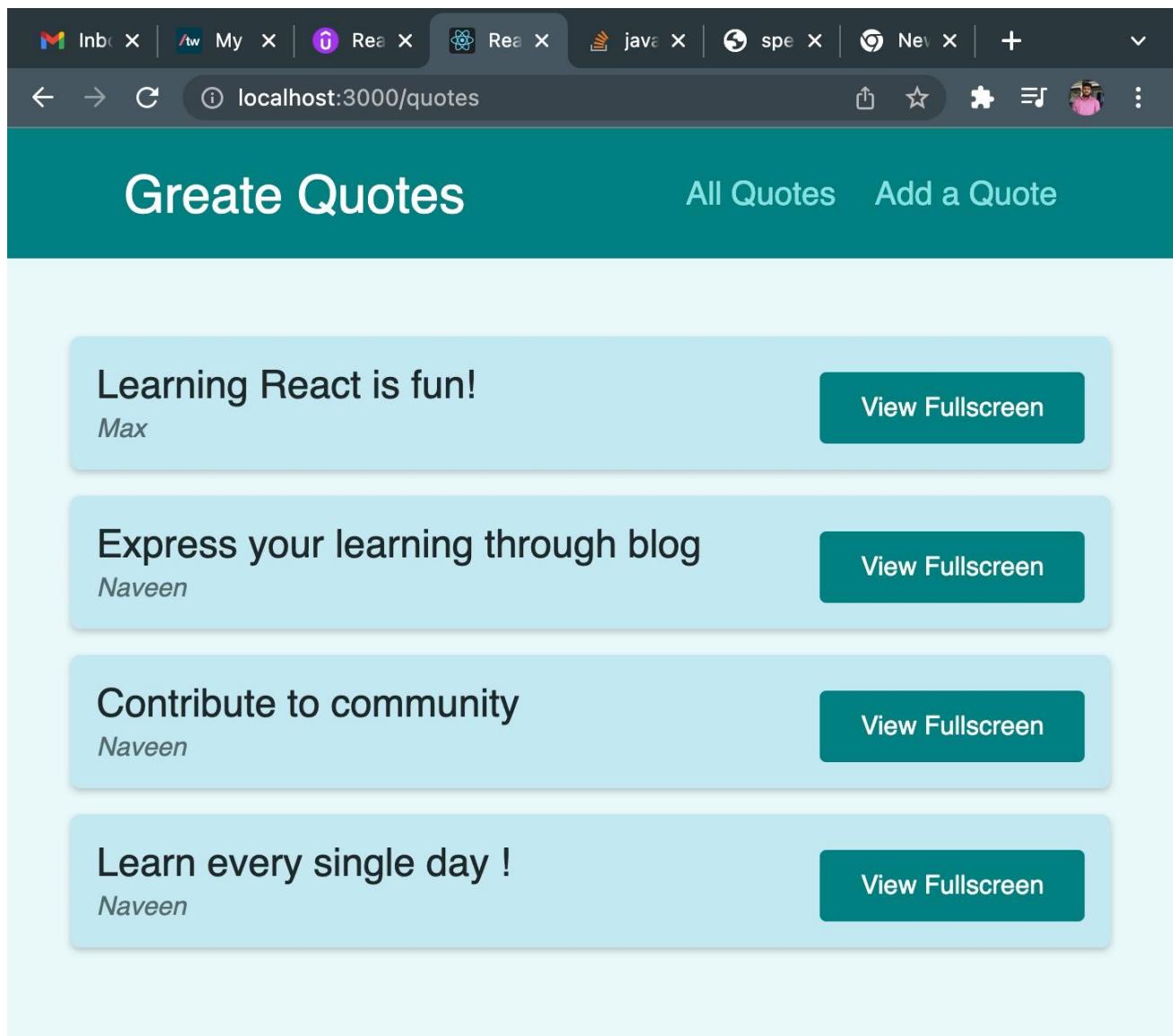
## Programmatic Navigation

```
Complexity is 4 Everything is cool!
const NewQuote = () => {
  const history = useHistory();

  const addQuoteHandler = quoteData => {
    console.log(quoteData);
    history.push('/quotes')
  }

  return <QuoteForm onAddQuote={addQuoteHandler} />;
}
export default NewQuote;
```

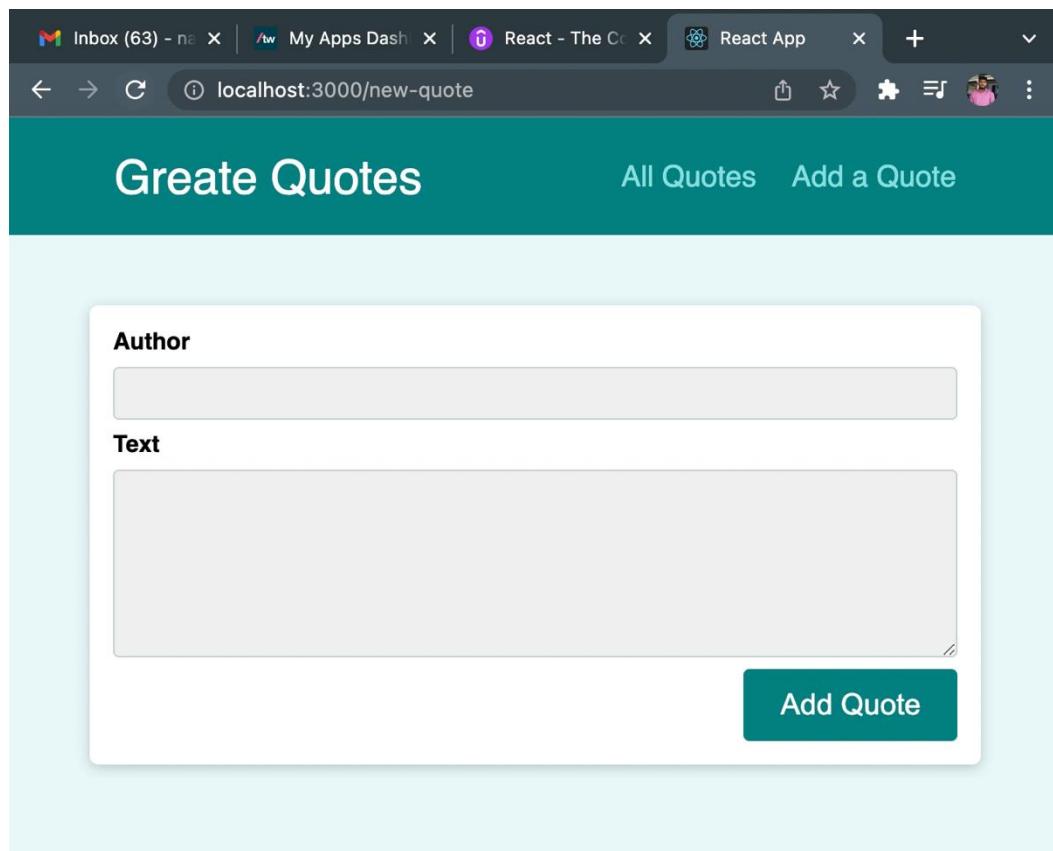
## Quote App Implementation So Far



## Prompting User on Navigation

```
const formFocusedHandler = () => {
  setIsEntered(true)
}

return (
  <Fragment>
    <Prompt
      when={isEntered}
      message={(location) =>
        "Are you sure you want to leave ? All your entered data will be lost!"
      }
    ></Prompt>
    <Card>
      <form
        onFocus={formFocusedHandler}
```



## Working with Query Parameters

```
act-router-demo-project > src > components > quotes > js QuoteList.js > ts QuoteList > ts ISorting
import { Fragment } from 'react';
import { useHistory, useLocation } from 'react-router-dom';

import QuoteItem from './QuoteItem';
import classes from './QuoteList.module.css';

Complexity is 9 It's time to do something...
const sortQuotes = (quotes, ascending) => {
  Complexity is 7 It's time to do something...
  return quotes.sort((quoteA, quoteB) => {
    if (ascending) {
      return quoteA.id > quoteB.id ? 1 : -1;
    } else {
      return quoteA.id < quoteB.id ? 1 : -1;
    }
  });
};

Complexity is 11 You must be kidding
const QuoteList = (props) => {

  const history = useHistory();
  const location = useLocation();

  const queryParam = new URLSearchParams(location.search);
  const isSortingAscending = queryParam.get('sort') === 'asc';
  const sortedQuotes = sortQuotes(props.quotes, isSortingAscending);

  const changeSortingHandler = () => {
    history.push("/quotes?sort=" + (isSortingAscending ? "desc" : "asc"));
  };

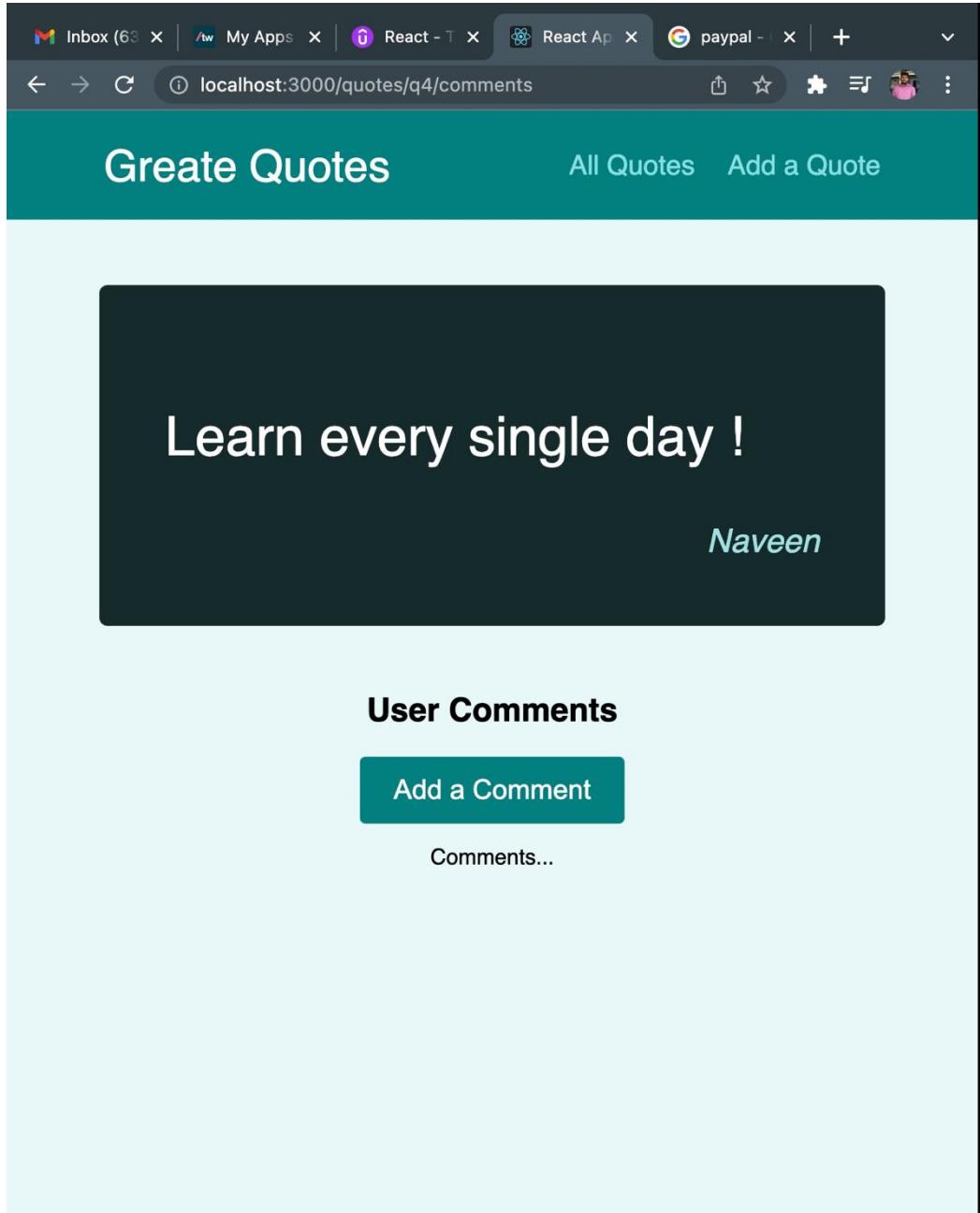
  return (
    <Fragment>
      <div className={classes.sorting}>
        <button onClick={changeSortingHandler}>
          Sort isSortingAscending ? "Descending" : "Ascending"
        </button>
      </div>
      <ul className={classes.list}>
        {sortedQuotes.map((quote) => (
          <QuoteItem
            key={quote.id}
            id={quote.id}
            author={quote.author}
            text={quote.text}
          />
        ))}
      </ul>
    </Fragment>
  );
}
```

The screenshot shows a web browser window with the URL `localhost:3000/quotes?sort=desc`. The page has a dark teal header with the title "Create Quotes" and navigation links "All Quotes" and "Add a Quote". Below the header is a button labeled "Sort Ascending". The main content area displays four quote cards, each with a "View Fullscreen" button. The quotes are:

- Learn every single day ! by Naveen
- Contribute to community by Naveen
- Express your learning through blog by Naveen
- Learning React is fun! by Max

Removing the component based on URL

```
return (
  <Fragment>
    <HighlightedQuote text={quote.text} author={quote.author} />
    <Route path={`/quotes/${params.quoteId}`} exact>
      <div className="centered">
        <Link
          className="btn--flat"
          to={`/quotes/${params.quoteId}/comments`}
        >
          Load Comments
        </Link>
      </div>
    </Route>
    <Route path="/quotes/:quoteId/comments">
      <Comments />
    </Route>
  </Fragment>
);
```



Writing more flexible route code

```
const history = useHistory();
const location = useLocation();

const queryParam = new URLSearchParams(location.search);
const isSortingAscending = queryParam.get('sort') === 'asc';
const sortedQuotes = sortQuotes(props.quotes, isSortingAscending);

const changeSortingHandler = () => {
  history.push({
    pathname: location.pathname,
    search: `?sort=${isSortingAscending ? "desc" : "asc"}`
  });
}
```

```

Complexity is 15. You must be kidding.
const QuoteDetail = () => {
  const match = useRouteMatch();
  const params = useParams();

  const quote = DUMMY_QUOTES.find(quote => quote.id === params.quoteId);

  if (!quote) {
    return <p>No quote found</p>
  }

  return (
    <Fragment>
      <HighlightedQuote text={quote.text} author={quote.author} />
      <Route path={match.path} exact>
        <div className="centered">
          <Link className="btn--flat" to={`${match.url}/comments`}>
            Load Comments
          </Link>
        </div>
      </Route>
      <Route path={`${match.path}/comments`}>
        <Comments />
      </Route>
    </Fragment>
  );
}

```

## React Router 6

```

import MainHeader from './components/MainHeader';

function App() {
  return (
    <div>
      <MainHeader />
      <main>
        <Routes>
          <Route path='/welcome' element={<Welcome />} />
          <Route path='/products' element={<Products />} />
          <Route path='/products/:productId' element={<ProductDetail />} />
        </Routes>
      </main>
    </div>
  );
}

export default App;

```

Switch changed to Routes

Exact has been removed

element is introduced in Route tag.

Order of the Routes does not matter

NavLink activeClassName has been removed, the alternate is given below,

```
>
NavLink activeClassName=((navData) => navData.isActive ? classes.active : '') to='
Welcome
/NavLink>
i>
>
NavLink activeClassName=((navData) => navData.isActive ? classes.active : '') to='
Products
/NavLink>
i>
```

Redirect changed to Navigate, Pls refer below,

```
function App() {
  return (
    <div>
      <MainHeader />
      <main>
        <Routes>
          <Route path="/" element={<Navigate to="/welcome" />} />
          <Route path="/welcome" element={<Welcome />} />
          <Route path="/products" element={<Products />} />
          <Route path="/products/:productId" element={<ProductDetail />} />
        </Routes>
      </main>
    </div>
  );
}

export default App;
```

We can add “replace” prop along with “to” prop. It will replace the component while navigating.

In version 6, Route tag is mandatory to enclosed with Routes.

```
App.js  Welcome.js  MainHeader.js
1 | import { Routes, Route } from 'react-router-dom';
2 |
3 | const Welcome = () => {
4 |   return (
5 |     <section>
6 |       <h1>The Welcome Page</h1>
7 |       <Routes>
8 |         <Route path="/welcome/new-user" element={<p>Welcome, new user!</p>} />
9 |       </Routes>
10 |     </section>
11 |   );
12 |
13 |
14 | export default Welcome;
15 |
```

While adding the nested routes, no need to specify the parent path. It is a relative path.

```
const Welcome = () => {
  return (
    <section>
      <h1>The Welcome Page</h1>
      <Routes>
        <Route path="new-user" element={<p>Welcome, new user!</p>} />
      </Routes>
    </section>
  );
}
```

Alternate way specifying the alternate route

```
function App() {
  return (
    <div>
      <MainHeader />
      <main>
        <Routes>
          <Route path="/" element={<Navigate to="/welcome" />} />
          <Route path="/welcome/*" element={<Welcome />}>
            <Route path="new-user" element={<p>Welcome, new user!</p>} />
          </Route>
          <Route path="/products" element={<Products />} />
          <Route path="/products/:productId" element={<ProductDetail />} />
        </Routes>
      </main>
    </div>
  );
}
```

Using Outlet as placeholder for nested Route element

```
import { Link, Outlet } from 'react-router-dom';

const Welcome = () => {
  return (
    <section>
      <h1>The Welcome Page</h1>
      <Link to="new-user">New User</Link>
      <Outlet />
    </section>
  );
};

export default Welcome;
```

useHistory changed to useNavigate

```
import { Link, useNavigate } from 'react-router-dom';

const Products = () => {
  const navigate = useNavigate();
  navigate('/welcome', {replace: true});
  return (
    <section>
      <h1>The Products Page</h1>
      <ul>
        <li>
```

We can specify number for navigate back and forth.

```
import { Link, useNavigate } from 'react-router-dom';

const Products = () => {
  const navigate = useNavigate();
  navigate(1);
  return (
    <section>
      <h1>The Products Page</h1>
      <ul>
```

Promt is removed.

## Deploying React App

### Deployment Step



# Lazy Loading

Load code only when it's needed

20-react-router-demo-project > src > `App.js` > `App`

```

1 import React, { Suspense } from 'react';
2 import {Redirect, Route, Switch} from 'react-router-dom';
3 import Layout from './components/layout/Layout';
4 import LoadingSpinner from './components/UI>LoadingSpinner';
5
6 const NewQuote = React.lazy(()=>import('./pages/NewQuotes'));
7 const AllQuotes = React.lazy(()=>import('./pages/AllQuotes'));
8 const NotFound = React.lazy(()=>import('./pages/NotFound'));
9 const QuoteDetails = React.lazy(()=>import('./pages/QuoteDetails'));
10
11 Complexity is 17 You must be kidding
12 function App() {
13   return (
14     <Layout>
15       <Suspense
16         fallback={
17           <div className="centered">
18             <LoadingSpinner />
19           </div>
20         }
21       >
22         <Switch>
23           <Route path="/" exact>
24             <Redirect to="/quotes" />
25           </Route>
26           <Route path="/quotes" exact>
27             <AllQuotes />
28           </Route>
29           <Route path="/quotes/:quoteId">
30             <QuoteDetails />
31           </Route>
32           <Route path="/new-quote">
33             <NewQuote />
34           </Route>
35           <Route path="*>
36             <NotFound />
37           </Route>
38         </Switch>
39       </Suspense>
40     </Layout>
41   );
42
43   export default App;
44 
```



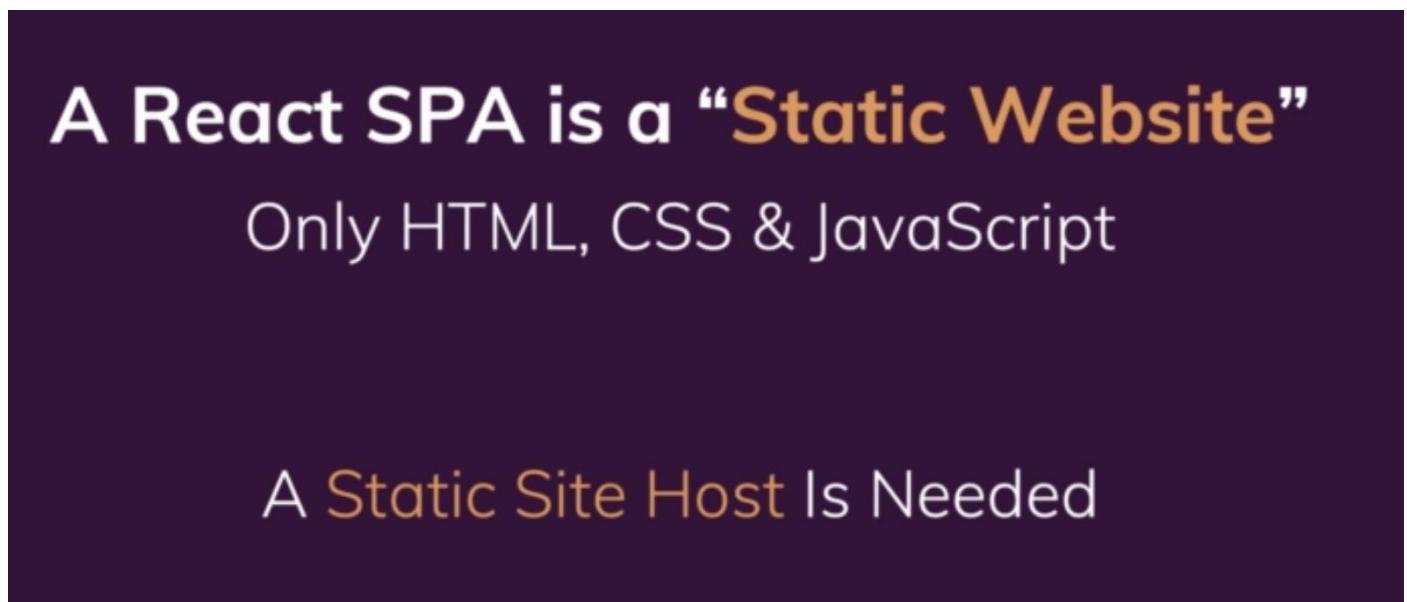
## Building the code for production

The screenshot shows a terminal window within a code editor interface. The terminal tab is active, displaying the command sequence for a production build:

```
/Users/naveen.kumar1/.zshrc:68: no matches found: load?  
→ REACT git:(master) ✘ cd 20-react-router-demo-project  
→ 20-react-router-demo-project git:(master) ✘ npm run build  
  
> react-complete-guide@0.1.0 build  
> react-scripts build  
  
Creating an optimized production build...  
■
```

The left sidebar shows a file tree with several projects listed under a parent folder. The current project is "20-react-router-de...", with its "build" and "node\_modules" folders selected.

Upload Production Code to Server



```
→ 20-react-router-demo-project git:(master) ✘ npm install -g firebase-tools
npm WARN deprecated har-validator@5.1.5: this library is no longer supported
npm WARN deprecated uuid@3.4.0: Please upgrade to version 7 or higher. Older versions are
be problematic. See https://v8.dev/blog/math-random for details.
npm WARN deprecated uuid@3.4.0: Please upgrade to version 7 or higher. Older versions are
be problematic. See https://v8.dev/blog/math-random for details.
npm WARN deprecated request@2.88.2: request has been deprecated, see https://gith

added 723 packages, and audited 724 packages in 40s

30 packages are looking for funding
  run `npm fund` for details

14 vulnerabilities (6 low, 8 moderate)

To address issues that do not require attention, run:
  npm audit fix

To address all issues (including breaking changes), run:
  npm audit fix --force

Run `npm audit` for details.
→ 20-react-router-demo-project git:(master) ✘
```

## × Set up Firebase Hosting

### 1 Install Firebase CLI

To host your site with Firebase Hosting, you need the Firebase CLI (a command line tool).

Run the following [npm](#) command to install the CLI or update to the latest CLI version.

```
$ npm install -g firebase-tools
```



Doesn't work? Take a look at the [Firebase CLI reference](#) or change your [npm permissions](#).

Also show me the steps to add the Firebase JavaScript SDK to my web app

The SDK includes Cloud Firestore, Authentication, Performance Monitoring and more. It can be added now or later.

[Next](#)

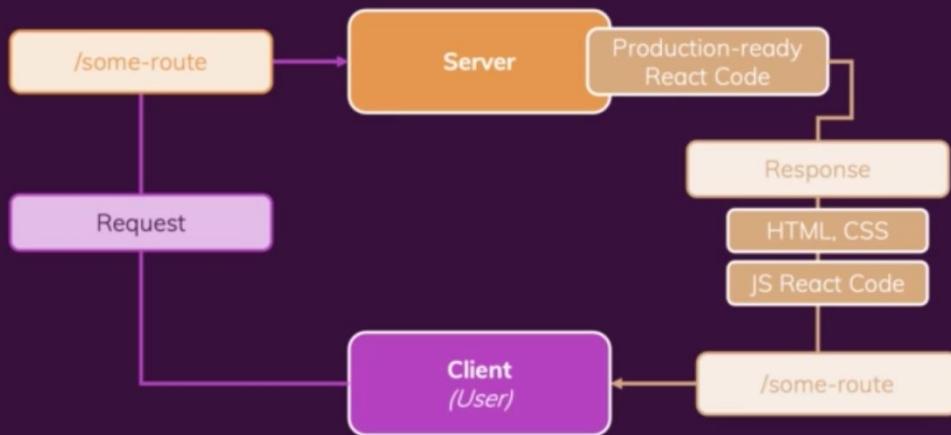
```
Ran `npm audit` for details.  
→ 20-react-router-demo-project git:(master) ✘ firebase login  
i Firebase optionally collects CLI usage and error reporting information to help improve our produc-  
acy policy (https://policies.google.com/privacy) and is not used to identify you.  
  
? Allow Firebase to collect CLI usage and error reporting information? Yes  
i To change your data collection preference at any time, run `firebase logout` and log in again.  
  
Visit this URL on this device to log in:  
https://accounts.google.com/o/oauth2/auth?client\_id=563584335869-fgrhgmd47bqnekij5i8b5pr03ho849e6.a  
3A%2F%2Fwww.googleapis.com%2Fauth%2Fcloudplatformprojects.readonly%20https%3A%2F%2Fwww.googleapis.c  
%2Fauth%2Fcloud-platform&response_type=code&state=370969519&redirect_uri=http%3A%2F%2Flocalhost%3A9  
  
Waiting for authentication...  
  
✓ Success! Logged in as naveen.kumar1@thoughtworks.com
```

```
→ 20-react-router-demo-project git:(master) ✘ firebase init  
#####  
## ## ## ## ## ##### ## ##### #####  
#### # # #### # # ##### # # ## ##  
## ## ## ## ## ##### ##### #####  
## ##### ## ## ## ##### ## ## ##  
## ##### ## ## ## ##### ## ## ##  
  
You're about to initialize a Firebase project in this directory:  
  
/Users/naveen.kumar1/Workspace/WEB/REACT/20-react-router-demo-project  
  
? Which Firebase features do you want to set up for this directory? Press Space to select features, the  
o Realtime Database: Configure a security rules file for Realtime Database and (optionally) provision  
o Firestore: Configure security rules and indexes files for Firestore  
o Functions: Configure a Cloud Functions directory and its files  
➤ o Hosting: Configure files for Firebase Hosting and (optionally) set up GitHub Action deploys  
o Hosting: Set up GitHub Action deploys  
o Storage: Configure a security rules file for Cloud Storage  
o Emulators: Set up local emulators for Firebase products  
(Move up and down to reveal more choices)
```

```
You're about to initialize a Firebase project in this directory:  
  
/Users/naveen.kumar1/Workspace/WEB/REACT/20-react-router-demo-project  
  
? Which Firebase features do you want to set up for this directory? Press Space to select fe-  
files for Firebase Hosting and (optionally) set up GitHub Action deploys  
  
== Project Setup  
  
First, let's associate this project directory with a Firebase project.  
You can create multiple project aliases by running firebase use --add,  
but for now we'll just set up a default project.  
  
? Please select an option: Use an existing project ←  
? Select a default Firebase project for this directory: food-order-app (food-order-app)  
➤ react-http-5c577 (react-http) ←  
  
But for now we'll just set up a default project.
```

```
? Please select an option: Use an existing project  
? Select a default Firebase project for this directory: react-http-5c577 (react-http)  
i Using project react-http-5c577 (react-http)  
  
== Hosting Setup  
  
Your public directory is the folder (relative to your project directory) that  
will contain Hosting assets to be uploaded with firebase deploy. If you  
have a build process for your assets, use your build's output directory.  
  
? What do you want to use as your public directory? build
```

## Server-side Routing vs Client-side Routing



Since we are having a single page application, we need to configure the server accordingly, otherwise server will consider the new url as separate request.

You're about to initialize a Firebase project in this directory:

```
/Users/naveen.kumar1/Workspace/WEB/REACT/20-react-router-demo-project
```

? Which Firebase features do you want to set up for this directory? Press Space to select features  
Configure files for Firebase Hosting and (optionally) set up GitHub Action deploys

### == Project Setup

First, let's associate this project directory with a Firebase project.  
You can create multiple project aliases by running `firebase use --add`,  
but for now we'll just set up a default project.

? Please select an option: Use an existing project  
? Select a default Firebase project for this directory: react-http-5c577 (react-http)  
i Using project react-http-5c577 (react-http)

### == Hosting Setup

Your `public` directory is the folder (relative to your project directory) that  
will contain Hosting assets to be uploaded with `firebase deploy`. If you  
have a build process for your assets, use your build's output directory.

? What do you want to use as your public directory? build  
? Configure as a single-page app (rewrite all urls to /index.html)? Yes  
? Set up automatic builds and deploys with GitHub? No  
? File `build/index.html` already exists. Overwrite? No  
i Skipping write of `build/index.html`  
  
i Writing configuration info to `firebase.json`...  
i Writing project information to `.firebaserc`...  
  
✓ Firebase initialization complete!

```
→ 20-react-router-demo-project git:(master) firebase deploy  
== Deploying to 'react-http-5c577'...  
  
i  deploying hosting  
i  hosting[react-http-5c577]: beginning deploy...  
i  hosting[react-http-5c577]: found 32 files in build  
✓  hosting[react-http-5c577]: file upload complete  
i  hosting[react-http-5c577]: finalizing version...  
✓  hosting[react-http-5c577]: version finalized  
i  hosting[react-http-5c577]: releasing new version...  
✓  hosting[react-http-5c577]: release complete  
  
✓  Deploy complete!  
  
Project Console: https://console.firebaseio.google.com/project/react-http-5c577/overview  
Hosting URL: https://react-http-5c577.web.app  
→ 20-react-router-demo-project git:(master)
```

## Adding Authentication to React App

CADE MIND

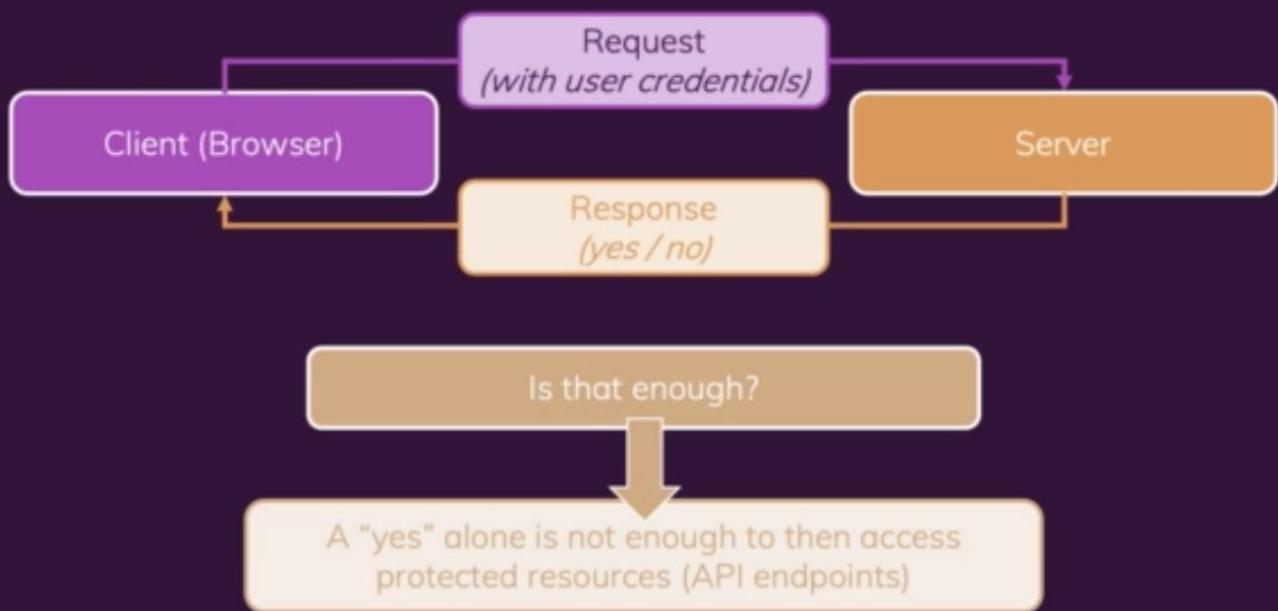
### Module Content

- How Authentication Works In React Apps
- Implementing User Authentication
- Adding Auth Persistence & Auto-Logout

## Two-step Process:

- 1) Get access / permission
- 2) Send request to protected resource

## Getting Permission



## How Does Authentication Work?

We can't just save and use the "yes"

We could send a fake "yes" to the server to request protected data

### Server-side Sessions

Store unique identifier on server, send same identifier to client

Client sends identifier along with requests to protected resources

### Authentication Tokens

Create (but not store) "permission" token on server, send token to client

Client sends token along with requests to protected resources

## Adding User Signup

<https://console.firebaseio.google.com/project/react-http-5c577/authentication/providers>

Choosing an authentication from firebase project.

The screenshot shows the 'Sign-in method' tab selected in the Firebase Authentication console. It displays a grid of providers:

Native providers	Additional providers
Email/Password	Google
Phone	Facebook
Anonymous	Play Games
	Game Center
	Apple
	Github
	Microsoft
	Twitter
	Yahoo

Below the grid, there is a section for 'Authorized domains'.

Saving the authentication type.

The screenshot shows the configuration for the 'Email/Password' sign-in provider. The 'Enable' switch is turned on. Below it, a description states: "Allow users to sign up using their email address and password. Our SDKs also provide email address verification, password recovery, and email address change primitives." A link to "Learn more" is provided. Another toggle switch for "Email link (passwordless sign-in)" is shown as off. At the bottom are "Cancel" and "Save" buttons.

## Sign-in provider.

The screenshot shows the Firebase Authentication console under the 'Sign-in method' tab. It lists a single provider, 'Email/Password', which is enabled. Below this, the 'Authorized domains' section shows three domains: 'localhost', 'react-http-5c577.firebaseio.com', and 'react-http-5c577.web.app', all of which are set to 'Default' type.

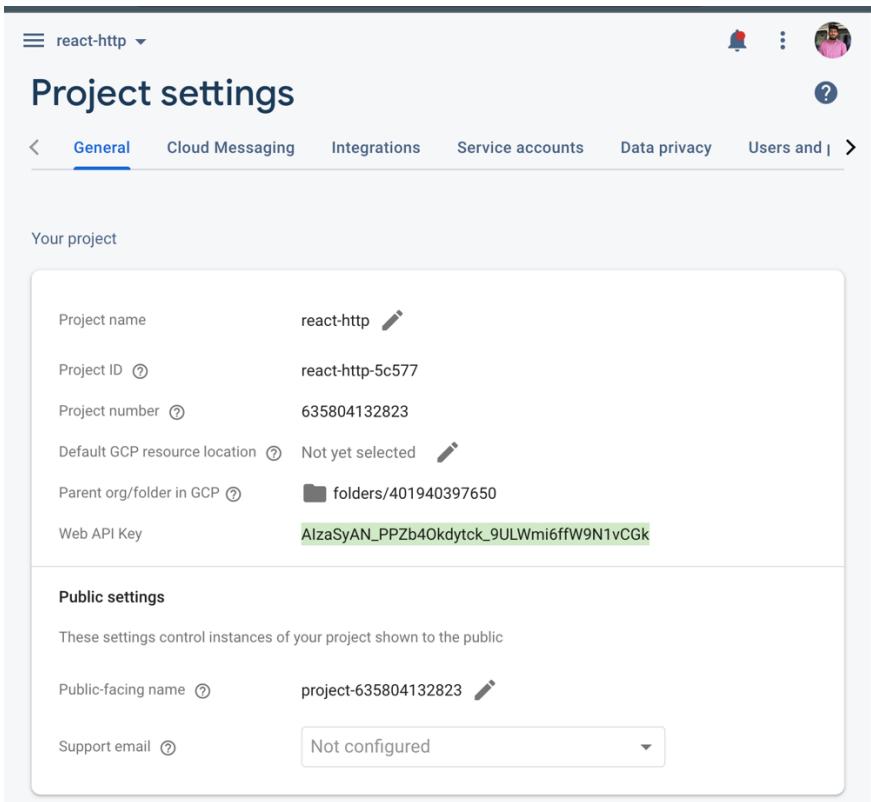
## Firebase Auth Rest API

The screenshot shows the Firebase Identity Toolkit documentation for signing up with email and password. It includes a note about missing refresh tokens, instructions for creating a user via POST to the `signupNewUser` endpoint, and details for the POST method, content type, and endpoint URL (`https://identitytoolkit.googleapis.com/v1/accounts:signUp?key=[API_KEY]`). It also provides a table for the request body payload properties.

### Request Body Payload

Property Name	Type	Description
email	string	The email for the user to create.
password	string	The password for the user to create.
returnSecureToken	boolean	Whether or not to return an ID and refresh token. Should always be true.

## Web API Key from firebase project setting for sign up



The screenshot shows the 'General' tab of the Firebase Project Settings. The 'Web API Key' field is populated with the value 'AIzaSyAN\_PPZb40kdytck\_9ULWmi6ffW9N1vCGK'. Other fields visible include 'Project name' (react-http), 'Project ID' (react-http-5c577), 'Project number' (635804132823), 'Default GCP resource location' (Not yet selected), 'Parent org/folder in GCP' (folders/401940397650), and 'Public settings' which show 'Support email' as 'Not configured'.

Sending a request to firebase on submitting the form, as we can see we are using the web api key we copied from firebase project setting.

```
Complexity is 7 It's time to do something...
const submitHandler = (event) => {
  event.preventDefault();
  const enteredEmail = emailInputRef.current.value;
  const enteredPassword = passwordInputRef.current.value;
  if (isLoggedIn){

} else {
  fetch(
    "https://identitytoolkit.googleapis.com/v1/accounts:signUp?key=AIzaSyAN_PPZb40kdytck_9"
  {
    method: "POST",
    body: JSON.stringify({
      email: enteredEmail,
      password: enteredPassword,
      returnSecureToken: true,
    }),
    headers: {
      "Content-Type": "application/json",
    },
  }
}

Complexity is 4 Everything is cool!
).then(res=> {
  if (res.ok) {
    //..
  } else {
    res.json().then(data => {
      console.log(data);
    });
  }
});
};

});
```

## Adding User Sign In

### Sign in with email / password

You can sign in a user with an email and password by issuing an HTTP `POST` request to the Auth `verifyPassword` endpoint.

**Method:** POST

**Content-Type:** application/json

#### Endpoint



#### Request Body Payload

Property Name	Type	Description
email	string	The email the user is signing in with.
password	string	The password for the account.
returnSecureToken	boolean	Whether or not to return an ID and refresh token. Should always be true.

A screenshot of a browser window showing a React Auth application. The page has a purple header with the title "React Auth" and navigation links for "Login", "Profile", and "Logout". Below the header is a "Login" form with fields for "Your Email" (containing "test@test.co") and "Your Password" (containing "\*\*\*\*\*"). A "Login" button is at the bottom of the form. To the right of the form is a link "Create new account". On the left side of the browser, the developer tools' "Console" tab is open, displaying the following log output:

```
[HMR] Waiting for update _log.js:24
signal from WDS...
POST https://identitytoolkit.googleapis.com/v1/accounts:signInWithPassword?key=AIzaSyAN_PPZb40kdytck9ULWm16fFW9N1vCGk 400
AuthForm.js:57
{kind: 'identitytoolkit#VerifyPasswordResponse', localId: 'gEyWx1G5Ezxe2nmNhNagu9Yia43', email: 'test@test.co', displayName: '', idToken: 'eyJhbGciOiJSUzI1NiIsImtpZCI6IjM1MDZmMzc1MjI0N2ZjZj...14puVkJ5zzPmxw6Xjx3L16mxofvZpla4qxN-F3TSYkC3iBQ', ...} 
  displayName: ""
  email: "test@test.co"
  expiresIn: "3600"
  idToken: "eyJhbGciOiJSUzI1NiIsI...
  kind: "identitytoolkit#VerifyPa...
  localId: "gEyWx1G5Ezxe2nmNhNag...
  refreshToken: "AFx04_q7_lg0yR-X...
  registered: true
> [[Prototype]]: Object
```

Complexity is 15 You must be kidding

```
const submitHandler = (event) => { █
  event.preventDefault();
  const enteredEmail = emailInputRef.current.value;
  const enteredPassword = passwordInputRef.current.value;

  setIsLoading(true);
  let url;
  if (isLoggedIn){
    url = 'https://identitytoolkit.googleapis.com/v1/accounts:signInWithPa
  } else {
    url = 'https://identitytoolkit.googleapis.com/v1/accounts:signUp?key=A
  }
  fetch(
    url,
    {
      method: "POST",
      body: JSON.stringify({
        email: enteredEmail,
        password: enteredPassword,
        returnSecureToken: true,
      }),
      headers: {
        "Content-Type": "application/json",
      },
    }
  )
```

Complexity is 10 It's time to do something...

```
).then(res=> { █
  setIsLoading(false);
  if (res.ok) {
    return res.json();
  } else {
    Complexity is 5 Everything is cool!
    return res.json().then(data => { █
      let errorMessage = 'Authentication failed';
      if ( data && data.error && data.error.message){
        errorMessage = data.error.message;
      }
      throw new Error(errorMessage);
    });
  }
}).then(data => {
  console.log(data);
}).catch(err => {
  alert(err.message);
});
};
```

## Redirecting the login page after the password change

```
Complexity is 10 It's time to do something...
const ProfileForm = () => { }

  const newPasswordInputRef = useRef();
  const authContext = useContext(AuthContext);
  const history = useHistory();

  const submitHandler = event => {
    event.preventDefault();
    const enteredNewPassword = newPasswordInputRef.current.value;
    console.log(authContext.token);
    let url = 'https://identitytoolkit.googleapis.com/v1/accounts:update?key';
    fetch(url, {
      method: "POST",
      body: JSON.stringify({
        idToken: authContext.token,
        password: enteredNewPassword,
        returnSecureToken: false,
      }),
      headers: {
        "Content-Type": "application/json",
      },
    }).then(res=> {
      history.replace("/auth");
      authContext.logout()
    });
  }
}
```

## Protecting Frontend Page

```
<Layout>
  <Switch>
    <Route path="/" exact>
      <HomePage />
    </Route>
    {!authContext.isLoggedIn && (
      <Route path="/auth">
        <AuthPage />
      </Route>
    )}
    <Route path="/profile">
      {authContext.isLoggedIn && <UserProfile />}
      {!authContext.isLoggedIn && <Redirect to="/auth" />}
    </Route>
    <Route path="*>
      <Redirect to="/" />
    </Route>
  </Switch>
</Layout>
```

## Storing login token in storage browser

```
Complexity is 5 Everything is cool!
export const AuthContextProvider = (props) => {
  const [token, setToken] = useState(null);
  const userIsLoggedIn = !token;

  const loginHandler = (token) => {
    setToken(token);
    localStorage.setItem('login_token', token);

  }

  const logoutHandler = () => {
    setToken(null);
    localStorage.removeItem('login_token');
  }

  const contextValue = {
    token,
    isLoggedIn: userIsLoggedIn,
    login: loginHandler,
    logout: logoutHandler
  };

  return (
    <AuthContext.Provider value={contextValue}>
      {props.children}
    </AuthContext.Provider>
  );
}
```

## Auto logout

```
const logoutHandler = () => {
  setToken(null);
  localStorage.removeItem("login_token");
}

const loginHandler = (token, expirationTime) => {
  setToken(token);
  localStorage.setItem("login_token", token);

  const remainingTime = calculateRemainingTime(expirationTime);

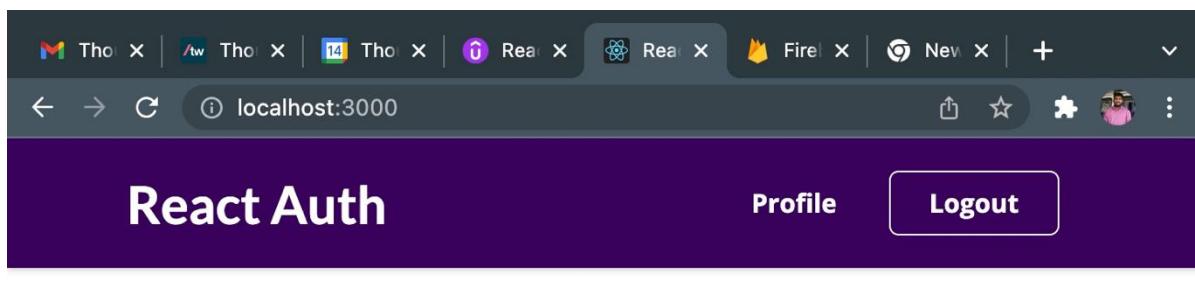
  setTimeout(logoutHandler, remainingTime);
}
```

Handling the timer while logout

```
const logoutHandler = () => {
  setToken(null);
  localStorage.removeItem("login_token");

  if (logoutHandler) {
    clearTimeout(logoutHandler);
  }
}

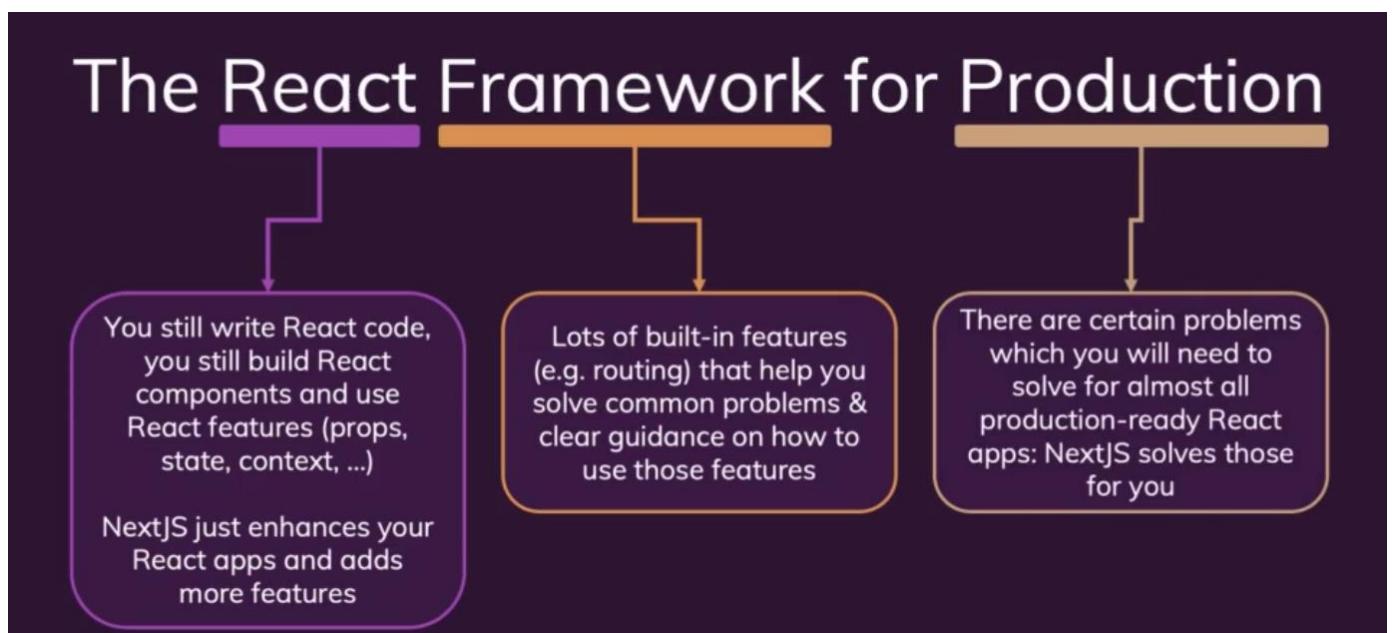
const loginHandler = (token, expirationTime) => {
  setToken(token);
  localStorage.setItem("login_token", token);
  const remainingTime = calculateRemainingTime(expirationTime);
  logoutTimer = setTimeout(logoutHandler, remainingTime);
}
```



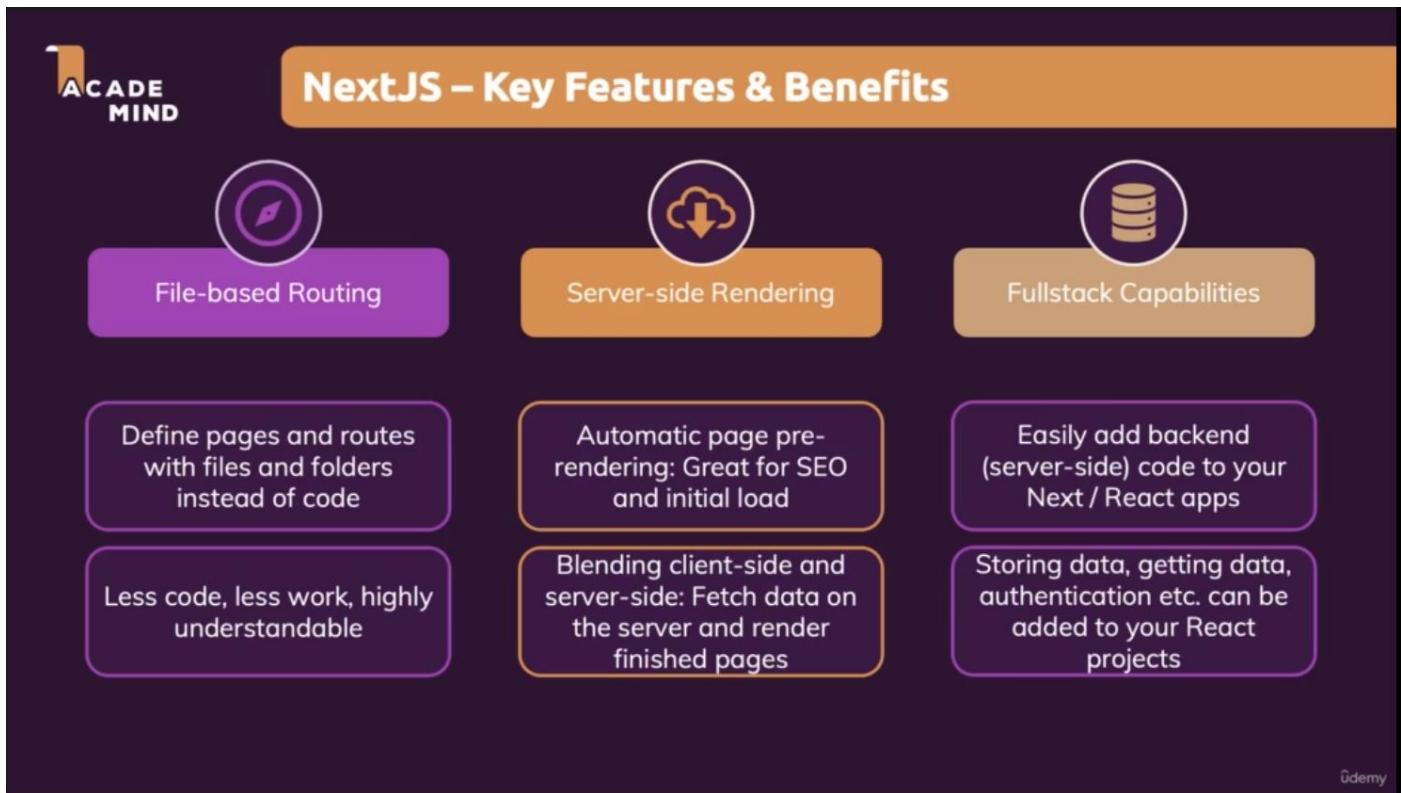
# Welcome on Board!



NextJS solves common problems and makes building React apps easier!



## Next JS Key Feature and Benefits



## Create Next JS Project

npx create-next-app

EXPLORER

- REACT
  - 01-starting-project
  - 02-react-styling
  - 03-react-errors
  - 04-user-project
  - 05-react-side-effect
  - 06-userreducer-starting-project
  - 07-food-order-app
  - 08-how-react-works
  - 09-class-based-component
  - 10-http-request
  - 11-http-request-post
  - 12-building-custom-hooks
  - 13-task-management-with-firebase
  - 14-user-forms-and-inputs
  - 15-food-order-app-with-http
  - 16-redux-demo
  - 17-redux-starter-project
  - 18-redux-advanced
  - 19-react-router
  - 20-react-router-demo-project
  - 21-react-auth-app
- 22-nextjs-app
  - .next
  - node\_modules
  - pages
    - \_app.js
    - index.js 1, M
    - news.js 1, U
  - public
  - styles
    - globals.css
    - .eslintrc.json
    - .gitignore

22-nextjs-app > pages > index.js > ...

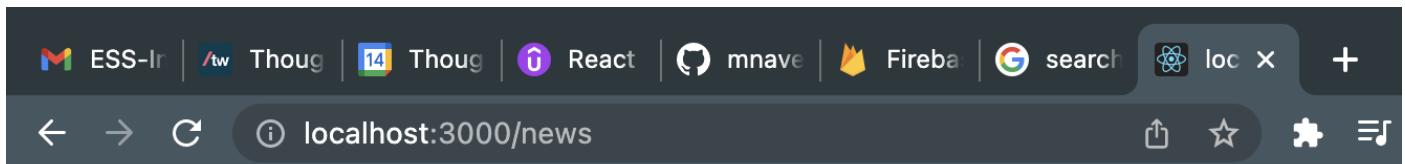
```
1 Complexity is 3 Everything is cool!
2 const HomePage = () => {
3   return <h1>Home Page</h1>
4 }
5
6 export default HomePage;
```

PROBLEMS 2 OUTPUT TERMINAL

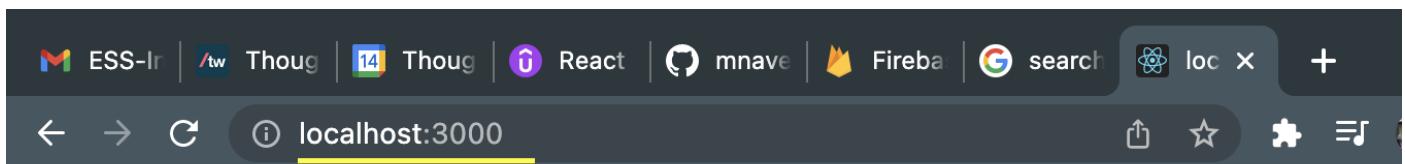
kumar1/Workspace/WEB/REACT/22-nextjs-app/node\_modules/next/dist/server/next-server.js:56:23)
at new Server (/Users/naveen.kumar1/Workspace/WEB/REACT/22-nextjs-app/node\_modules/next/dist/server/base-server.js:101:29)
at new NextNodeServer (/Users/naveen.kumar1/Workspace/WEB/REACT/22-nextjs-app/node\_modules/next/dist/server/next-server.js:146:9)
at NextServer.createServer (/Users/naveen.kumar1/Workspace/WEB/REACT/22-nextjs-app/node\_modules/next/dist/server/next-server.js:111:16)
at async /Users/naveen.kumar1/Workspace/WEB/REACT/22-nextjs-app/node\_modules/next/dist/server/next.js:123:31
-> 22-nextjs-app git:(master) x npm run dev

> dev
> next dev

ready - started server on 0.0.0.0:3000, url: http://localhost:3000
event - compiled client and server successfully in 3.1s (125 modules)
Attention: Next.js now collects completely anonymous telemetry regarding usage.
This information is used to shape Next.js' roadmap and prioritize features.
You can learn more, including how to opt-out if you'd like to participate in this anonymous program, by visiting the following URL:
https://nextjs.org/telemetry



# News Page

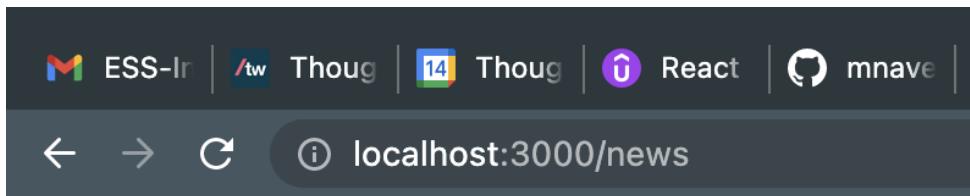


# Home Page

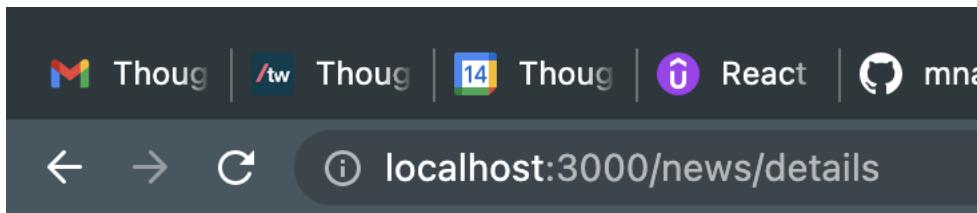
Nested Pages

A screenshot of a file explorer interface showing the directory structure of a Next.js application:

- > 21-react-auth-app
- ✓ 22-next-js-app
  - > .next
  - > node\_modules
  - ✓ pages
    - ✓ news
      - details.js
      - index.js
      - \_app.js
      - index.js
    - public
  - ✓ styles
    - globals.css



# News Page



# Detail Page

## Dynamic Page

Create the file name with square bracket. Here, newsId is a placeholder. Now this placeholder accept any values.

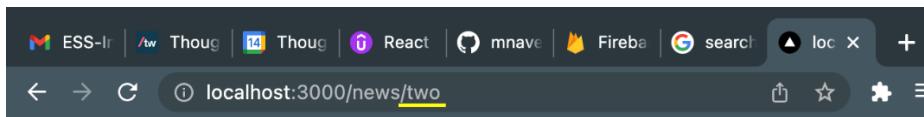
domainname/news/1  
domainname/news/2

A screenshot of a file explorer showing the directory structure of a Next.js application named "22-next-js-app".

- 20-react-router-demo-project
- 21-react-auth-app
- 22-next-js-app
  - .next
  - node\_modules
  - pages
    - news
      - [newsId].js
      - index.js
      - \_app.js
      - index.js
  - public
  - styles
    - globals.css
    - .eslintrc.json

# Detail Page

## Extracting a Dynamic Parameter Value



## Detail Page two

```
import { useRouter } from 'next/router'

Complexity is 3 Everything is cool!
const DetailPage = () => {
  const router = useRouter();

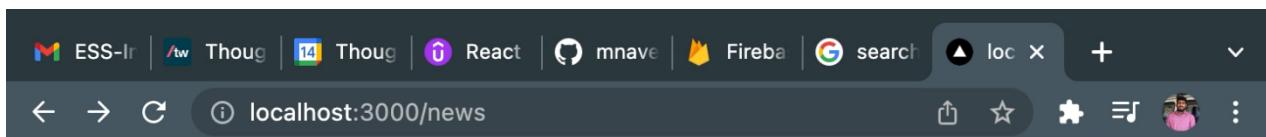
  const newsId = router.query.newsId;

  return <h1>Detail Page {newsId}</h1>
}

export default DetailPage;
```

## Linking between Pages

Normal way – using anchor element



## News Page

- Artical 1
- Artical 2
- Artical 3
- Artical 4

```

x~js-app / pages / news / index.js ...
import { Fragment } from "react";

Complexity is 13 You must be kidding
const NewsPage = () => {
  return (
    <Fragment>
      <h1>News Page</h1>
      <ul>
        <li>
          <a href="/news/artical1">Artical 1</a>
        </li>
        <li>
          <a href="/news/artical2">Artical 2</a>
        </li>
        <li>
          <a href="/news/artical3">Artical 3</a>
        </li>
        <li>
          <a href="/news/artical4">Artical 4</a>
        </li>
      </ul>
    </Fragment>
  );
}

export default NewsPage;

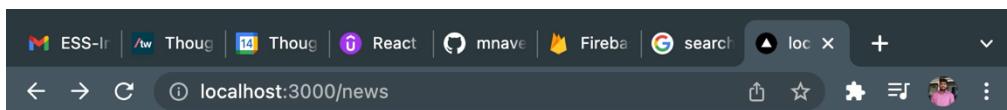
```

Disadvantage:

- Not a single page application
- Every time page loads when clicking the link.

Other way – using Link from next js

This approach gives a single page application.

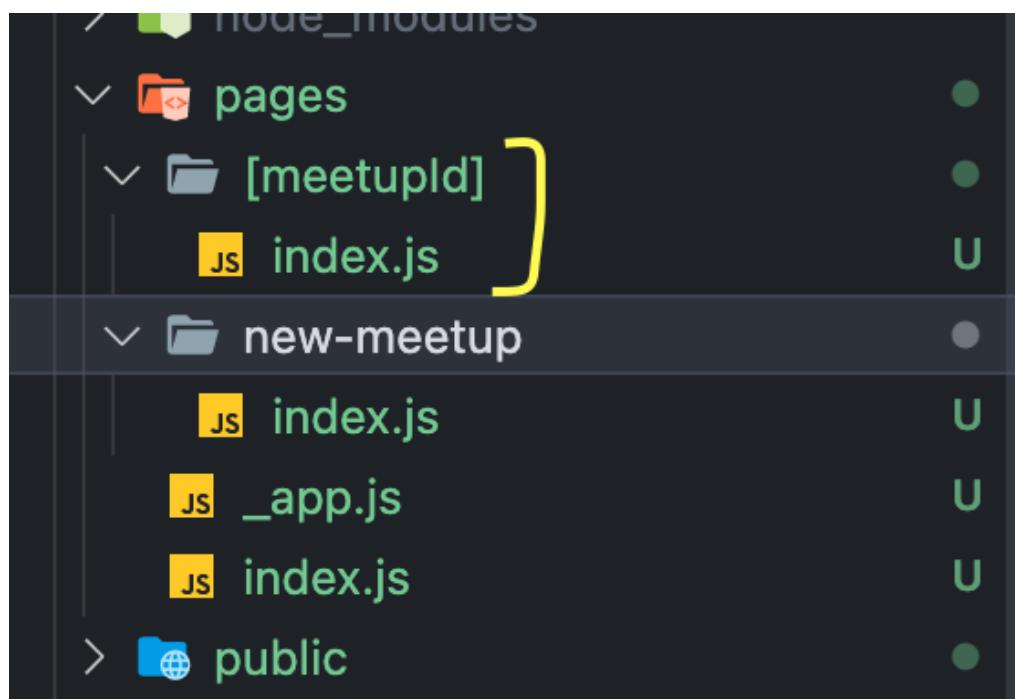


## News Page

- Artical 1
- Artical 2
- Artical 3
- Artical 4

```
const NewsPage = () => {  
  return (  
    <Fragment>  
      <h1>News Page</h1>  
      <ul>  
        <li>  
          <Link href="/news/artical1">Artical 1</Link>  
        </li>  
        <li>  
          <Link href="/news/artical2">Artical 2</Link>  
        </li>  
        <li>  
          <Link href="/news/artical3">Artical 3</Link>  
        </li>  
        <li>  
          <Link href="/news/artical4">Artical 4</Link>  
        </li>  
      </ul>  
    </Fragment>  
  );  
}  
  
export default NewsPage;
```

Preparing project pages



```
Setup Management > pages > index.js > [1] Homepage
import MeetupList from '../components/meetups/MeetupList';

const DUMMY_MEETUPS = [
  {
    id: 'm1',
    title: 'A First Meetup',
    image: 'https://officesnapshots.com/wp-content/uploads/2019/01/Chennai-Meetup-1.jpg',
    address: 'Chennai',
    description: 'This is a first meetup'
  },
  {
    id: 'm2',
    title: 'A Second Meetup',
    image: 'https://officesnapshots.com/wp-content/uploads/2019/01/Chennai-Meetup-2.jpg',
    address: 'Chennai',
    description: 'This is a second meetup'
  }
]

Complexity is 3 Everything is cool!
const HomePage = () => {
  return <MeetupList meetups={DUMMY_MEETUPS} />;
}

export default HomePage;
```



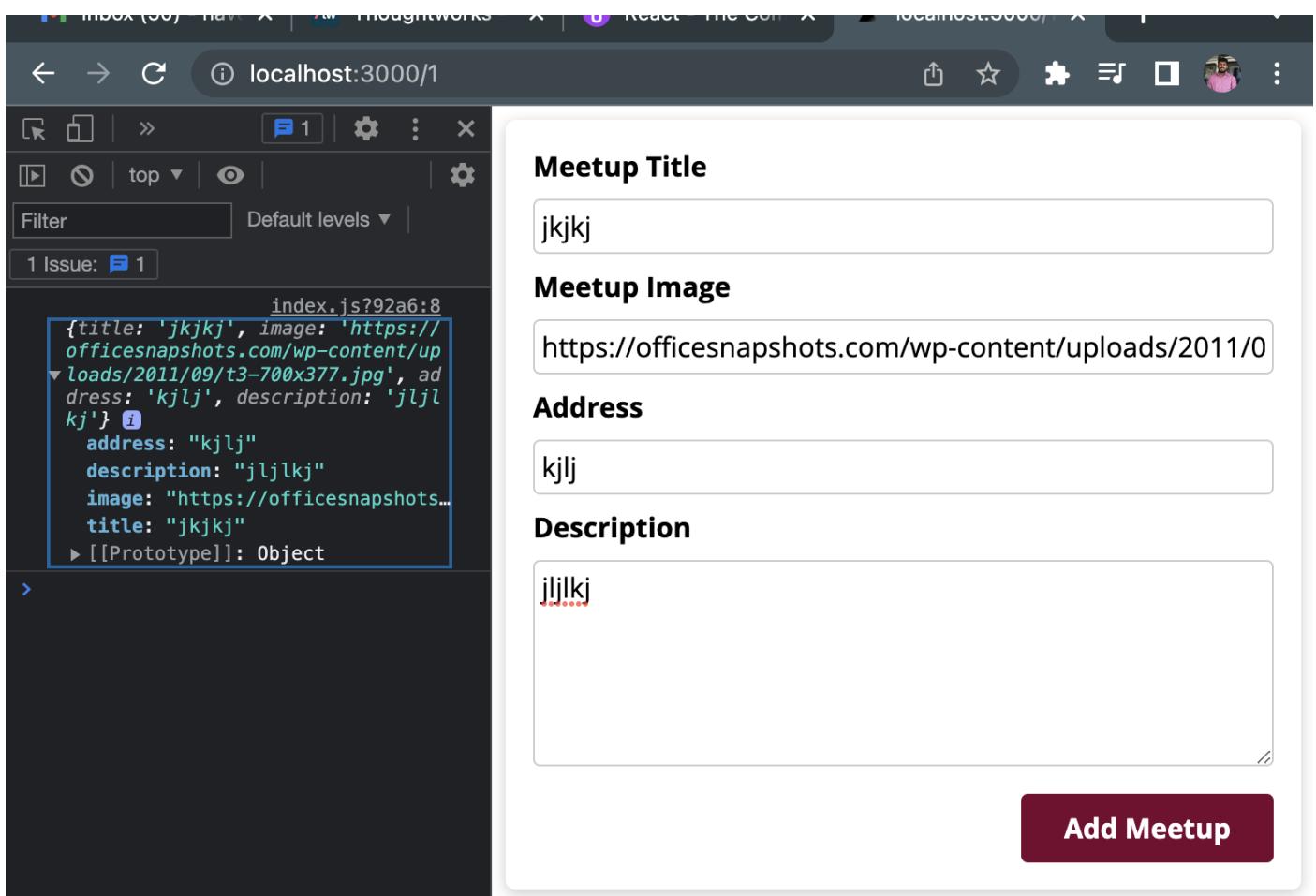
### A First Meetup

*Chennai*

[Show Details](#)



## Adding a new meetup form



A screenshot of a browser window showing a 'Meetup Title' input field containing 'jkjkj'. The browser interface includes a sidebar with a code editor showing a snippet of JavaScript code related to meetups.

Meetup Title  
jkjkj

Meetup Image  
<https://officesnapshots.com/wp-content/uploads/2011/09/t3-700x377.jpg>

Address  
kjlkj

Description  
jljlkj

Add Meetup

```
import { useRouter } from 'next/router';
import NewMeetupForm from '../../../../../components/meetups/NewMeetupForm';

Complexity is 4 Everything is cool!
const NewMeetupPage = () => {
  const router = useRouter();
  const newsId = router.query.newsId;
  const addMeetupHandler = (enteredMeetupData) => {
    console.log(enteredMeetupData);
  }
  return <NewMeetupForm onAddMeetup={addMeetupHandler} />;
}

export default NewMeetupPage;
```

## Adding a Navigation

```
import classes from './MainNavigation.module.css';
import Link from 'next/link';

Complexity is 10 It's time to do something...
function MainNavigation() {
  return (
    <header className={classes.header}>
      <div className={classes.logo}>React Meetups</div>
      <nav>
        <ul>
          <li>
            <Link href='/'>All Meetups</Link>
          </li> —
          <li>
            <Link href='/new-meetup'>Add New Meetup</Link>
          </li> —
        </ul>
      </nav>
    </header>
  );
}

export default MainNavigation;
```



## React Meetups

All  
Meetups

Add New  
Meetup



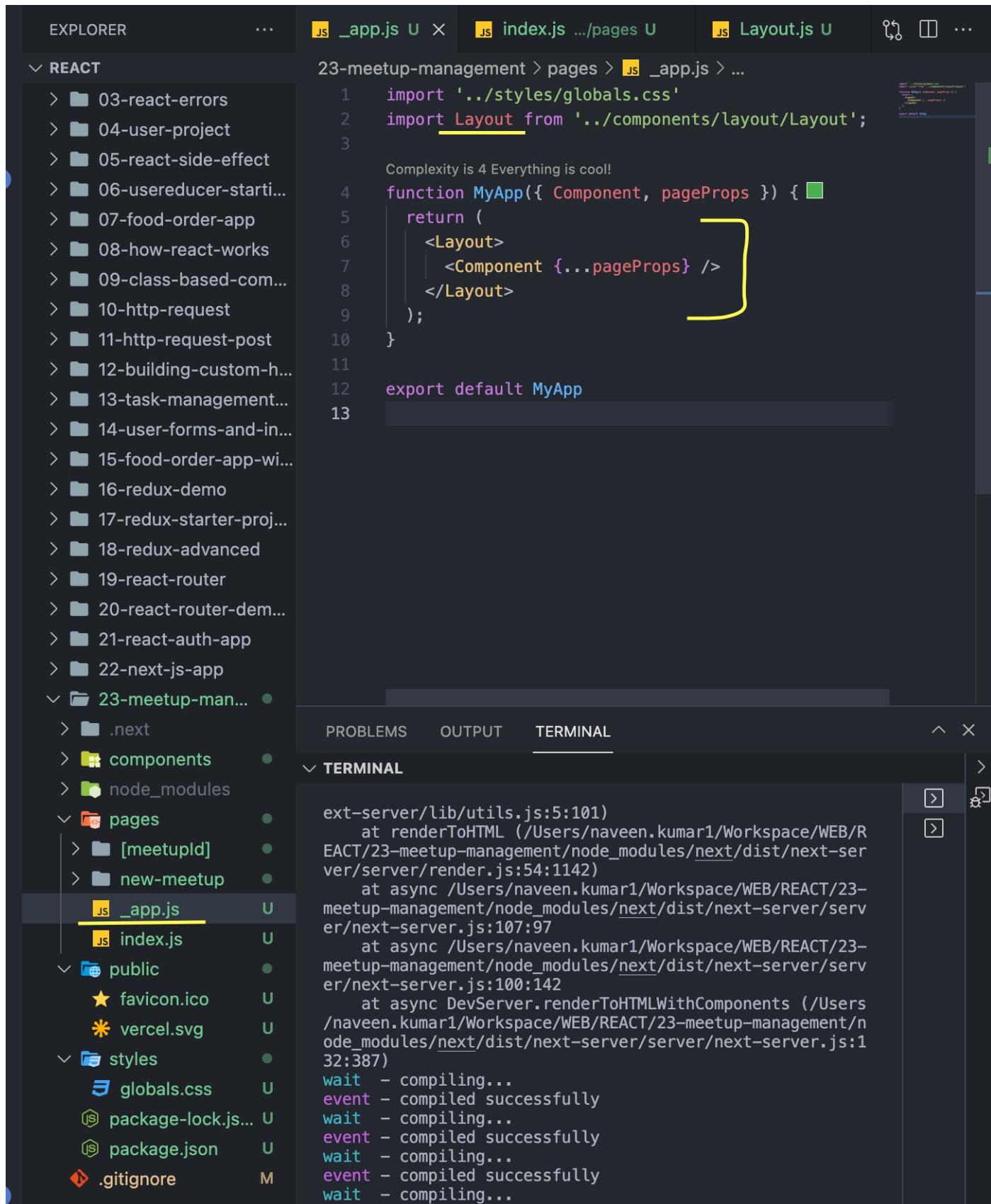
### A First Meetup

*Chennai*

Show Details

## Adding a navigation bar to entire pages

This is where `_app.js` file comes in handy in next js project. When we want change something which should through the app we can use the `_app.js` file.



The screenshot shows the VS Code interface with the following components:

- EXPLORER** sidebar: Shows a tree view of the project structure under the `REACT` category, including folders like `03-react-errors`, `04-user-project`, etc., and files like `.next`, `components`, `node_modules`, `pages`, `public`, `styles`, and `index.js`.
- CODE EDITOR**: The `_app.js` file is open. The code is as follows:

```
1 import '../styles/globals.css'
2 import Layout from '../components/layout/Layout';
3
4 Complexity is 4 Everything is cool!
5 function MyApp({ Component, pageProps }) {
6   return (
7     <Layout>
8       <Component {...pageProps} />
9     </Layout>
10  );
11
12 export default MyApp
13
```

- TERMINAL**: The terminal window shows the output of the application's build process, indicating the rendering of the `Layout` component across multiple files.



The screenshot shows a React application running on localhost:3000. The header bar includes a back arrow, forward arrow, refresh button, and a search icon. The URL 'localhost:3000' is displayed. The main navigation bar has three items: 'React Meetups' (highlighted in red), 'All Meetups', and 'Add New Meetup'. Below the navigation is a large image of a modern conference room with long tables, black office chairs, and large windows along the back wall. Overlaid on the image is a card containing the text 'A First Meetup' and 'Chennai', followed by a 'Show Details' button.

# React Meetups

All Meetups

Add New Meetup

## A First Meetup

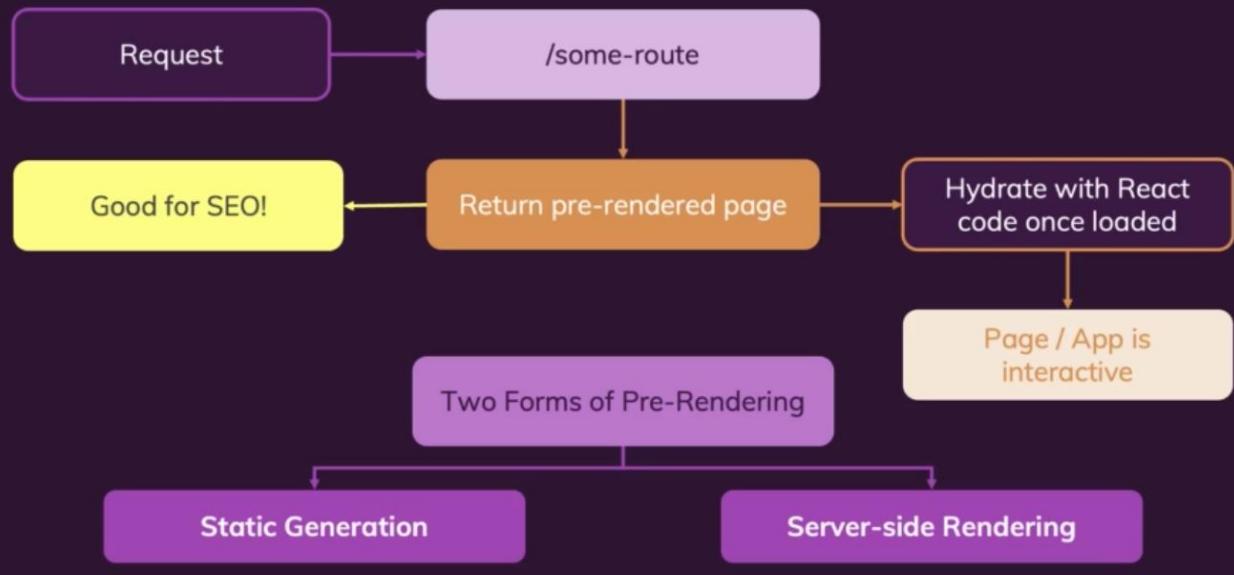
Chennai

Show Details

Next JS helps fetch the data for pre-rendering.

Data Fetching for static pages

## Page Pre-Rendering



### Static Site Generation (SSG)

```
];

Complexity is 3 Everything is cool!
const HomePage = (props) => {
  return <MeetupList meetups={props.meetups} />;
};

export async function getStaticProps() {
  return {
    props : {
      meetups: DUMMY_MEETUPS
    }
  };
}

export default HomePage;
```

`getStaticProps()` will be executed before the component function. It is used to provide the data which component function requires. Also, this special function will be executed during the build process.

Regenerate the code

revalidate attribute in return statement of getStaticProps function, helps to regenerate the props value every once in server side for given duration.

```
Complexity is 3 Everything is cool!
const HomePage = (props) => { 
  return <MeetupList meetups={props.meetups} />;
};

export async function getStaticProps() {
  return {
    props : {
      meetups: DUMMY_MEETUPS
    },
    revalidate: 10
  };
}
```

```
→ 23-meetup-management git:(master) ✘ npm run build

> nextjs-course@0.1.0 build
> next build

info  - Creating an optimized production build
info  - Compiled successfully
info  - Collecting page data
info  - Generating static pages (4/4)
info  - Finalizing page optimization

Page
  ● /
    └ css/a328f70f1500a47f59a4.css
    /_app
    ○ /[meetupId]
      └ css/f13f2474f39af1e03147.css
    ○ /404
    ○ /new-meetup
      └ css/60f9e9f292a28db77853.css
+ First Load JS shared by all
  |- chunks/94268f14e8a7805d20be5bd304f076c04a2cb3ee.49b41a.js
  |- chunks/framework.d886aa.js
  |- chunks/main.cfc182.js
  |- chunks/pages/_app.c98eca.js
  |- chunks/webpack.50bee0.js
  |- css/8b9a233bedcad23e246e.css

Size      First Load JS
691 B    65.1 kB
409 B
0 B       64.4 kB
498 B    64.9 kB
81 B
3.46 kB   67.8 kB
753 B    65.1 kB
360 B

λ (Server)  server-side renders at runtime (uses getInitialProps or getServerSideProps)
○ (Static)  automatically rendered as static HTML (uses no initial props)
● (SSG)     automatically generated as static HTML + JSON (uses getStaticProps)
  (ISR)     incremental static regeneration (uses revalidate in getStaticProps)

→ 23-meetup-management git:(master) ✘
```

## Server-Side Rendering (SSR)

```
Complexity is 3 Everything is cool!
const HomePage = (props) => {
  return <MeetupList meetups={props.meetups} />;
};

export async const getServerSideProps = (context) => {
  const req = context.req;
  const res = context.res;

  //fetch data from an API

  return {
    props: {
      meetups: DUMMY_MEETUPS
    }
  };
};
```

Server-Side Props will be regenerated for every incoming request.

## getStaticPaths()

```
export const getStaticPaths = async () => {
  return {
    fallback: false,
    paths: [
      {
        params: {
          meetupId: "m1",
        },
      },
      {
        params: {
          meetupId: "m2",
        },
      },
    ],
  };
};
```

`getStaticPaths()` tells, for which path, static side generation should happen.

```
export const getStaticProps = async (context) => {

  const meetupId = context.params.meetupId;
  console.log(meetupId);

  return {
    props: {
      meetupData: {
        image:
          "https://officesnapshots.com/wp-content/uploads/2011/09/t3-700x377",
        id: "m1",
        title: "Fresh Meetup",
        address: "Chennai",
        description: "This is a first meetup",
      },
    },
  };
};

export default MeetupDetailPage;
```

## Working with MongoDB

The screenshot shows the MongoDB Atlas interface. On the left, there's a sidebar with 'Project 0' and sections for 'DEPLOYMENT', 'DATA SERVICES', and 'SECURITY'. In the center, a modal window titled 'Connect to Cluster0' is open. It has three tabs: 'Setup connection security' (selected), 'Choose a connection method', and 'Connect'. Step 1, 'Select your driver and version', shows 'DRIVER' set to 'Node.js' and 'VERSION' set to '4.0 or later'. Step 2, 'Add your connection string into your application code', contains a code input field with the following content:

```
mongodb+srv://mnaveensmn:<password>@cluster0.q9ej8.mongodb.net/myFirstDatabase?  
retryWrites=true&w=majority
```

Below the code, it says: 'Replace <password> with the password for the mnaveensmn user. Replace myFirstDatabase with the name of the database that connections will use by default. Ensure any option params are URL encoded.'

At the bottom of the modal, there are 'Go Back' and 'Close' buttons. To the right of the modal, there's a sidebar with 'All Clusters', 'Get Help', and 'Naveen'. A green 'Create' button is also visible. At the bottom of the page, there's a 'Get Started' button and some footer text.

```
^C
→ 23-meetup-management git:(master) ✘ npm install mongodb
added 1 package, and audited 320 packages in 6s
28 packages are looking for funding
  run `npm fund` for details
7 vulnerabilities (6 moderate, 1 high)

To address all issues (including breaking changes), run:
  npm audit fix --force

Run `npm audit` for details.
→ 23-meetup-management git:(master) ✘
```

Collection – Tables

Documents – Entries in Tables – Row

Inserting data to MongoDB

```
import { MongoClient } from "mongodb";

const hanlder = async (req, res) => {
  if (req.method === "POST") {
    const data = req.body;

    const client = await MongoClient.connect(
      "mongodb+srv://mnaveensmn:Y05X9Mfva6hiZ0FV@cluster0.q9ej8.mongodb.net/meetups"
    );

    const db = client.db();

    const meetupsCollections = db.collection("meetups");
    const result = await meetupsCollections.insertOne(data);
    console.log(result);
    client.close();
    res.status(201).json({ message: "Meeting Inserted!" });
  }
};

export default hanlder;
```

DATABASES: 1 COLLECTIONS: 1

**meetups.meetups**

COLLECTION SIZE: 430B TOTAL DOCUMENTS: 2 INDEXES TOTAL SIZE: 20KB

Find Indexes Schema Anti-Patterns Aggregation Search Indexes

INSERT DOCUMENT

QUERY RESULTS 1-2 OF 2

```

_id: ObjectId("61e2d146a304dc036ef7e44b")
title: "React Lessons"
image: "https://officesnapshots.com/wp-content/uploads/2011/09/t3-700x377.jpg"
address: "Thoughtworks,Chennai."
description: "Meetup on React Advanced Concepts"

_id: ObjectId("61e2d19ea304dc036ef7e44c")
title: "React Lessons"
image: "https://officesnapshots.com/wp-content/uploads/2011/09/t3-700x377.jpg"
address: "Thoughtworks,Chennai."
description: "Meetup on React Advanced Concepts"

```

Get Started

System Status: All Good

Code Editor:

```

const data = req.body;
const client = await MongoClient.connect(`mongodb+srv://mnaveen${process.env.MONGO_DB}`);
const db = client.db();
const meetupsCollections = db.collection('meetups');
const result = await meetupsCollections.insertOne(data);
console.log(result);
client.close();
res.status(201).json({ message: 'Meetup created' });
}
export default handler;

```

TERMINAL

```

at async DevServer.run (/Users/naveen.kumar1/Workspace/WEB/REACT/23-meetup-management/node_modules/next/dist/next-server/server/next-server.js:66:1042)
  to: 'next-dev-server', handleRequest (/Users/naveen.kumar1/Workspace/WEB/REACT/23-meetup-management/no
de_modules/next/dist/next-server/server/next-server.js:34:504)
  wait - compiling...
  event - compiled successfully
  {
    acknowledged: true,
    insertedId: new ObjectId("61e2d19ea304dc036ef7e44c")
  }

```

localhost:3000/new-meetup

# React Meetups

All Meetups Add New Meetup

Meetup Title

Meetup Image

Address

Description

Add Meetup

## Loading Data from MongoDB

```
Complexity is 3 Everything is cool!
export const getStaticProps = async () => {
  const client = await MongoClient.connect(
    "mongodb+srv://mnaveensmn:Y05X9Mfva6hiZ0FV@cluster0.q9ej8.mongodb.net/meetup"
  );
  const db = client.db();
  const meetupsCollections = db.collection("meetups");
  const meetups = await meetupsCollections.find().toArray();
  client.close();

  return {
    props: {
      meetups: meetups.map((meetup) => ({
        title: meetup.title,
        address: meetup.address,
        image: meetup.image,
        id: meetup._id.toString()
      })),
    },
    revalidate: 10,
  };
};
```

## Loading a Dynamic Data in MeetupDetails page

```
export const getStaticProps = async (context) => {

  const meetupId = context.params.meetupId;
  console.log(meetupId);
  const client = await MongoClient.connect(
    "mongodb+srv://mnaveensmn:Y05X9Mfva6hiZ0FV@cluster0.q9ej8.mongodb.net/meetup"
  );

  const db = client.db();

  const meetupsCollections = db.collection("meetups");

  const selectedMeetup = await meetupsCollections.findOne({ _id: ObjectId(meetupId) });
  console.log(selectedMeetup);
  client.close();

  return {
    props: {
      meetupData: {
        id: selectedMeetup._id.toString(),
        title: selectedMeetup.title,
        address: selectedMeetup.address,
        image: selectedMeetup.image,
        description: selectedMeetup.description,
      },
    },
  };
};
```

```

Complexity is 3 Everything is cool!
const MeetupDetailPage = (props) => { █
  return (
    <MeetupDetail
      image={props.meetupData.image}
      title={props.meetupData.title}
      address={props.meetupData.address}
      description={props.meetupData.description}
    />
  );
}

Complexity is 3 Everything is cool!
export const getStaticPaths = async () => { █
  const client = await MongoClient.connect(
    "mongodb+srv://mnaveensmn:Y05X9Mfva6hiZ0FV@cluster0.q9ej8.mongodb.net/meetu█
  );

  const db = client.db();

  const meetupsCollections = db.collection("meetups");

  const meetups = await meetupsCollections.find({}, { _id: 1 }).toArray();

  client.close();

  return {
    fallback: false,
    paths: meetups.map((meetup) => ({
      params: { meetupId: meetup._id.toString() },
    })),
  };
};

```

## Adding a Meta Data to Page before Deployment

```

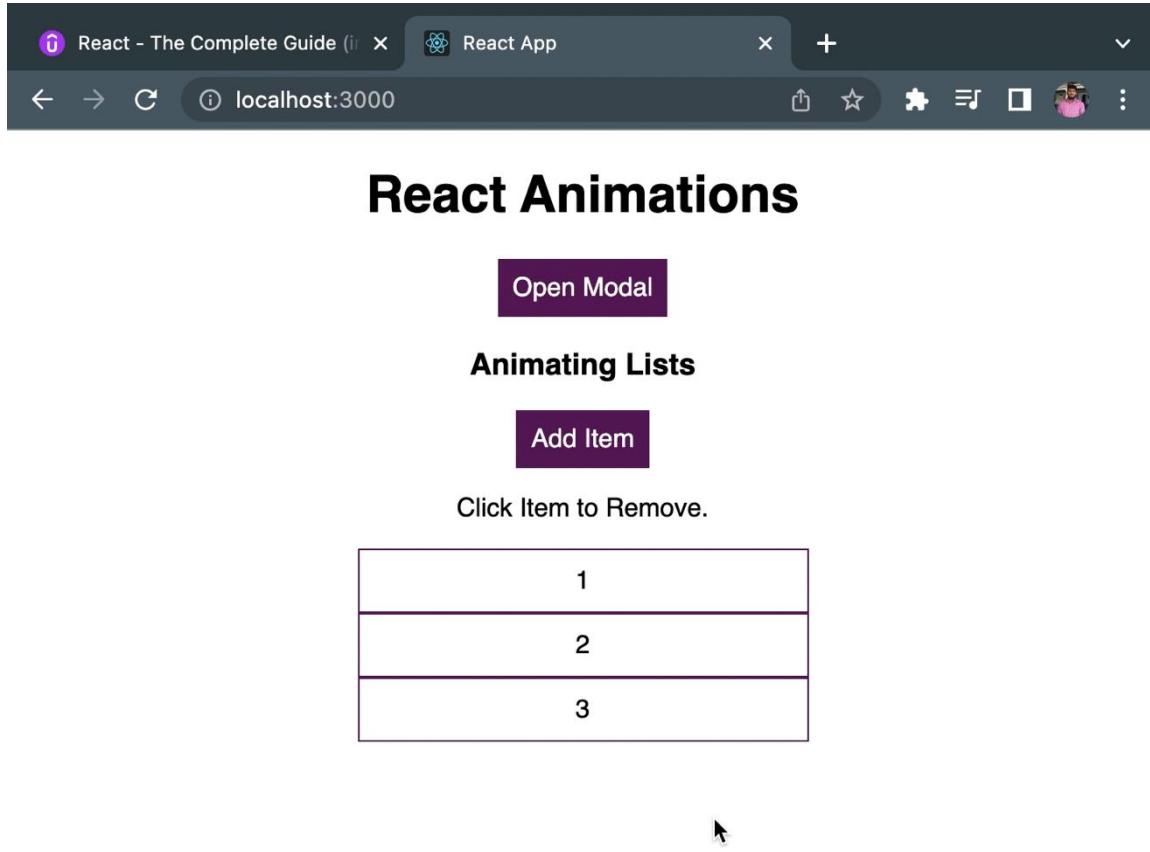
Complexity is 7 It's time to do something...
const HomePage = (props) => { █
  return (
    <Fragment>
      <Head>
        <title>React Meetups</title>
        <meta
          name="description"
          content="Browse a huge list of highly active React meetups!"
        />
      </Head>
      <MeetupList meetups={props.meetups} />
    </Fragment>
  );
};

```

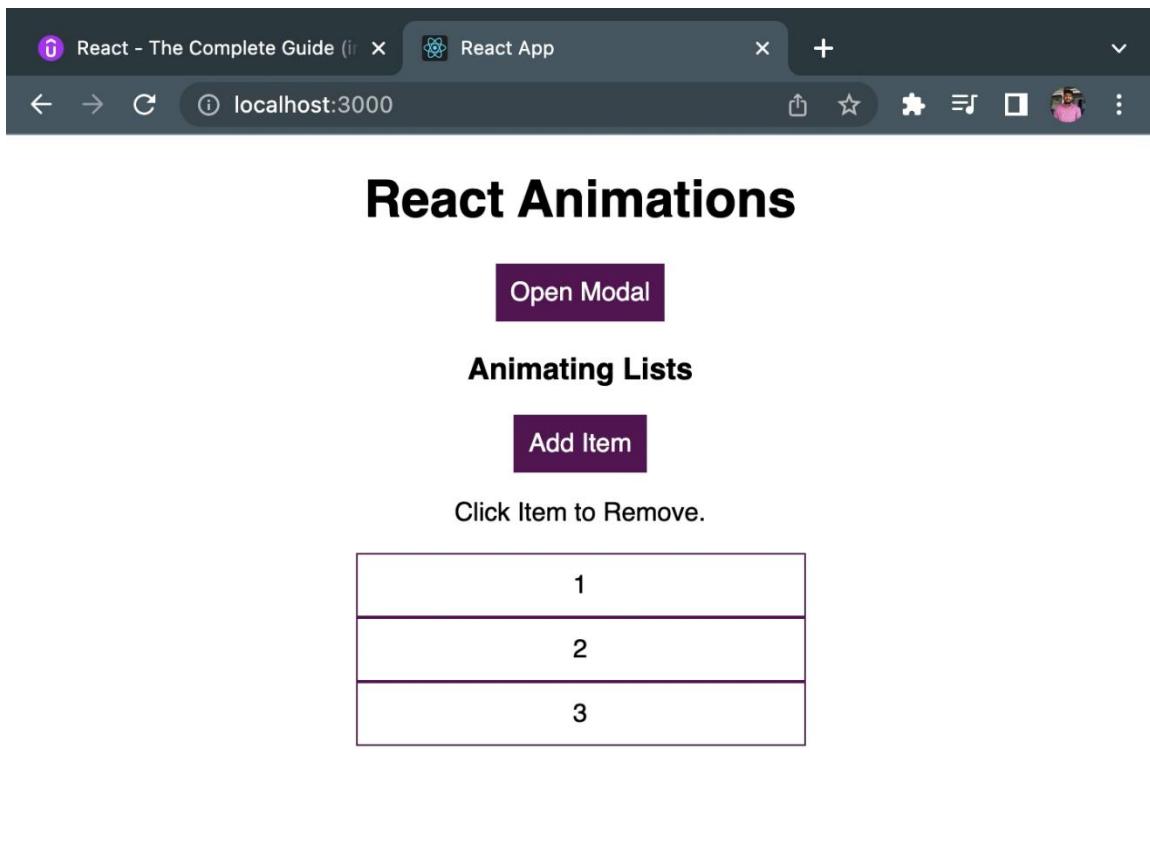


## Animation

Preparing the project for adding animation



Animating the model pop up



```
.Modal {  
  position: fixed;  
  z-index: 200;  
  border: 1px solid #eee;  
  box-shadow: 0 2px 2px #ccc;  
  background-color: white;  
  padding: 10px;  
  text-align: center;  
  box-sizing: border-box;  
  top: 30%;  
  left: 25%;  
  width: 50%;  
  transition: all 0.3s ease-in;  
  
}  
  
.ModalOpen {  
  opacity: 1;  
  transform: translateY(0);  
  
}  
  
.ModalClose {  
  opacity: 0;  
  transform: translateY(-100%);  
}
```

```
import React from 'react';  
  
import './Modal.css';  
  
Complexity is 6 It's time to do something...  
const Modal = (props) =>{  
  
  const cssClasses = ["Modal", props.show ? "ModalOpen" : "ModalClose"];  
  
  return (  
    <div className={cssClasses.join(" ")}>  
      <h1>A Modal</h1>  
      <button className="Button" onClick={props.closed}>  
        | Dismiss  
      </button>  
    </div>  
  );  
};  
  
export default Modal;
```

## CSS Animation Property

```
.ModalOpen {  
    animation: openModel 0.4s ease-out forwards;  
}  
  
.ModalClose {  
    animation: closeModel 0.4s ease-out forwards;  
}  
  
@keyframes openModel {  
    0%{  
        opacity: 0;  
        transform: translateY(-100%);  
    }  
  
    50%{  
        opacity: 1;  
        transform: translateY(90%);  
    }  
  
    100%{  
        opacity: 1;  
        transform: translateY(0);  
    }  
}  
  
@keyframes closeModel {  
    0%{  
        opacity: 1;  
        transform: translateY(0);  
    }  
  
    50%{  
        opacity: 0.8;  
        transform: translateY(60%);  
    }  
  
    100%{  
        opacity: 0;  
        transform: translateY(-100%);  
    }  
}
```

# React Animations

Open Modal

## Animating Lists

Add Item

Click Item to Remove.

1
2
3

Using ReactTransition group

```
→ 24-react-animations git:(master) ✘ npm install react-transition-group --save
npm WARN EBADENGINE Unsupported engine {
npm WARN   package: 'got@5.7.1',
npm WARN   required: { node: '>=0.10.0 <7' },
npm WARN   current: { node: 'v14.17.5', npm: '7.24.0' }
npm WARN EBADENGINE }
npm WARN deprecated fsevents@1.1.2: fsevents 1 will break on node v14+ and could
be using insecure binaries. Upgrade to fsevents 2.
(||) :: reify:dom-helpers: http fetch GET 200 https://registry.n
```

```
import React, { Component } from "react";
import Transition from 'react-transition-group/Transition';
import "./App.css";
import Modal from "./components/Modal/Modal";
import Backdrop from "./components/Backdrop/Backdrop";
import List from "./components/List>List";
```

```
    >
      Complexity is 3 Everything is cool!
      {({state}) => (
        <div
          style={{
            backgroundColor: "red",
            width: 100,
            height: 100,
            margin: "auto",
            transition: "opacity 1s ease-out",
            opacity: state === "exiting" ? 0 : 1,
          }}
        ></div>
      )}
    </Transition>
```

The screenshot shows a browser window with the title "React - The Complete Guide" and the URL "localhost:3000". The page content is titled "React Animations". It features a purple button labeled "Open Modal" which has a "Toogle" label above it. Below this is a section titled "Animating Lists" with a purple button labeled "Add Item". A text instruction "Click Item to Remove." is followed by a list of three items: "1", "2", and "3", each enclosed in a separate rectangular box.

## Adding Transition to Model Popup

```
<Transition
  mountOnEnter
  unmountOnExit
  in={this.state.modalIsOpen}
  timeout={300}>
  {(state) => <Modal show={state} closed={this.closeModal} />}
</Transition>
<br />
{this.state.modalIsOpen ? <Backdrop show /> : null}
<button className="Button" onClick={this.showModal}>
  Open Modal
</button>
<h3>Animating Lists</h3>
<List />
```

The screenshot shows a browser window with the title "React - The Complete Guide". The address bar indicates the URL is "localhost:3000". The main content area displays the text "React Animations" in large bold letters. Below it is a purple button labeled "Open Modal". A smaller purple button labeled "Toggle" is positioned above the "Open Modal" button. The text "Animating Lists" is displayed in bold. A purple button labeled "Add Item" is present. Below this, the text "Click Item to Remove." is shown. A list of three items, "1", "2", and "3", is displayed in separate boxes, each with a small delete icon in the top right corner.

## Wrapping the Transition Component

It will work the same way as the previous demo video.

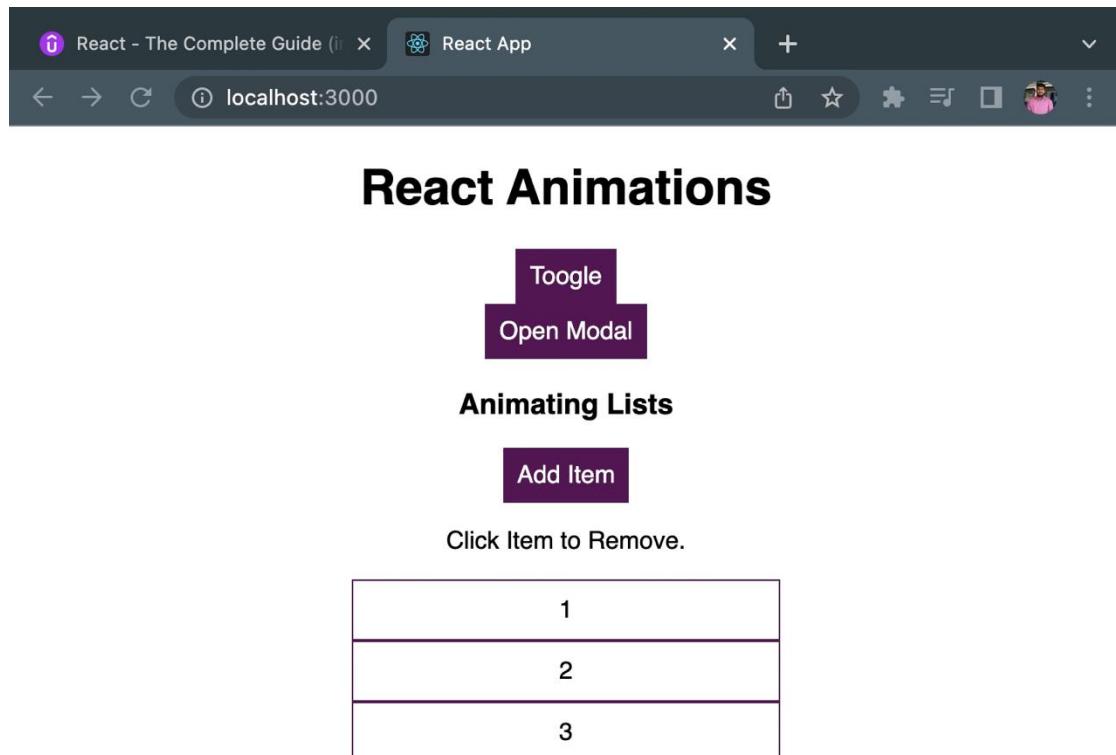
```
</button>
<Transition
  mountOnEnter
  unmountOnExit
  in={this.state.showBlock}
  timeout={300}>
  Complexity is 3 Everything is cool!
  {({state}) => (
    <div
      style={{
        backgroundColor: "red",
        width: 100,
        height: 100,
        margin: "auto",
        transition: "opacity 1s ease-out",
        opacity: state === "exiting" ? 0 : 1,
      }}
    ></div>
  )}
</Transition>
<Modal show={this.state.modalIsOpen} closed={this.closeModal} />
<br />
{this.state.modalIsOpen ? <Backdrop show /> : null}
<button className="Button" onClick={this.showModal}>
  Open Modal
</button>
<h2>Animating Lists</h2>
```

```
Complexity is 10 It's time to do something...
const Modal = (props) => {
  return (
    <Transition mountOnEnter unmountOnExit in={props.show} timeout={300}>
      Complexity is 7 It's time to do something...
      {({state}) => {
        const cssClasses = [
          "Modal",
          state === "entering"
            ? "ModalOpen"
            : state === "exiting"
            ? "ModalClosed"
            : null,
        ];
        return (
          <div className={cssClasses.join(" ")}>
            <h1>A Modal</h1>
            <button className="Button" onClick={props.closed}>
              Dismiss
            </button>
          </div>
        );
      }}
    </Transition>
  );
};

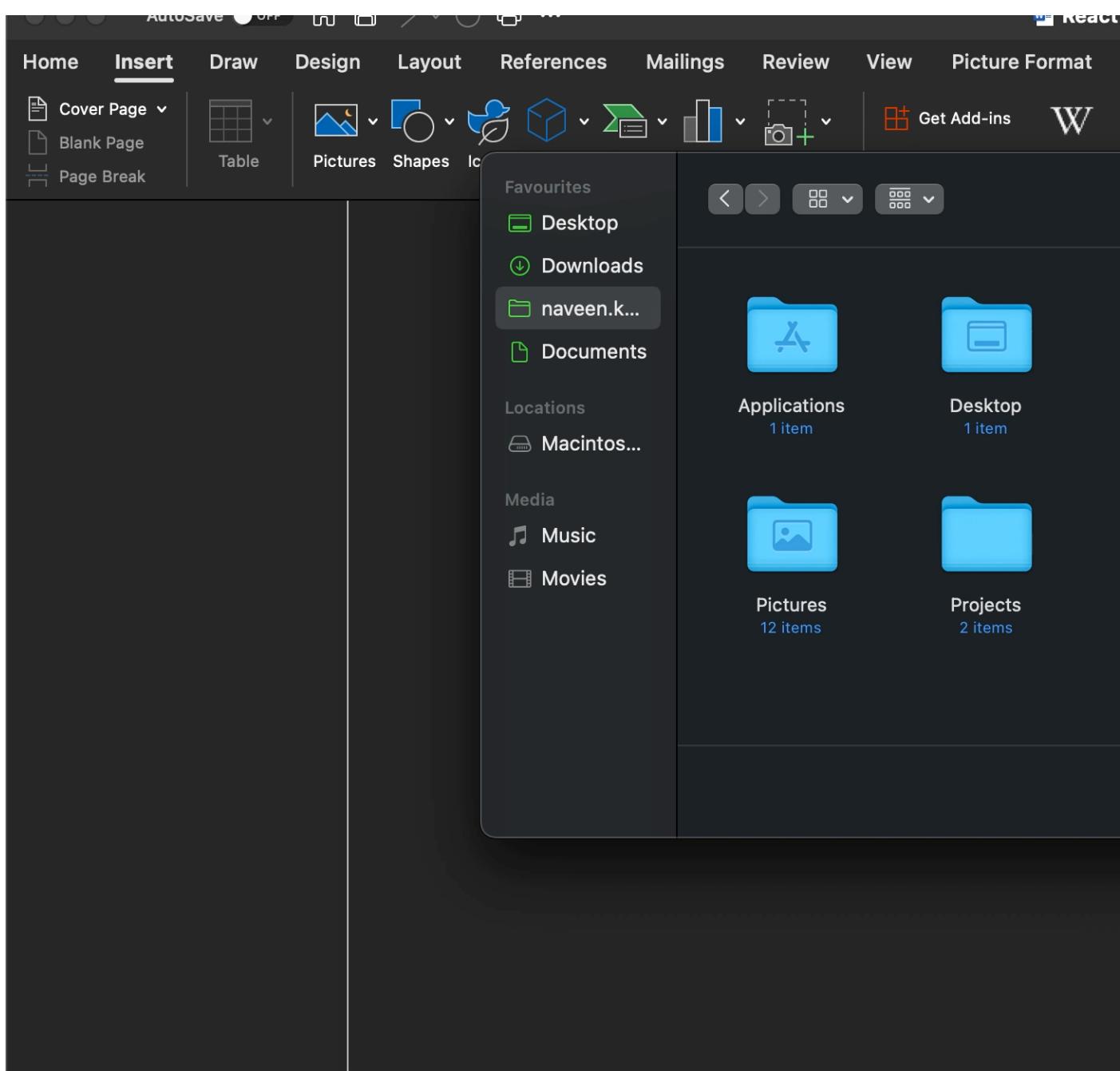
export default Modal;
```

## Animation Timings

```
Complexity is 10 It's time to do something...
const Modal = (props) => {
  const animationTiming = {
    enter: 400,
    exit: 1000
  }
  return (
    <Transition
      mountOnEnter
      unmountOnExit
      in={props.show}
      timeout={animationTiming}
    >
      Complexity is 7 It's time to do something...
      {(state) => {
        const cssClasses = [
          "Modal",
          state === "entering"
            ? "ModalOpen"
            : state === "exiting"
            ? "ModalClosed"
            : null,
        ]
      }}
    
```



## Transition Events

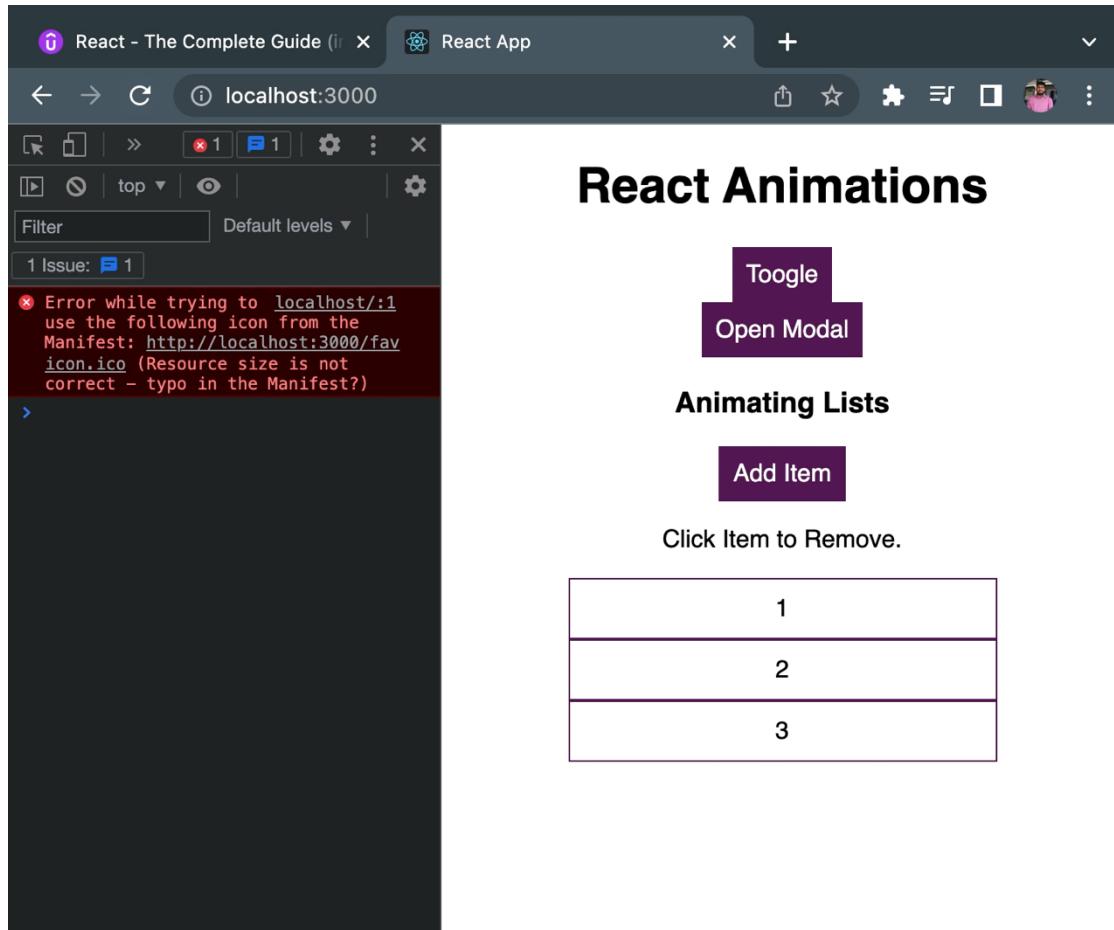


```

<Transition
  mountOnEnter
  unmountOnExit
  in={this.state.showBlock}
  timeout={300}
  onEnter={() => console.log("onEnter")}
  onEntering={() => console.log("onEntering")}
  onEntered={() => console.log("onEntered")}
  onExit={() => console.log("onExit")}
  onExiting={() => console.log("onExiting")}
  onExited={() => console.log("onExited")}

>
  Complexity is 3 Everything is cool!
  {({state}) => (
    <div
      style={{
        backgroundColor: "red",
        width: 100,
        height: 100,
        margin: "auto",
        transition: "opacity 1s ease-out",
        opacity: state === "exiting" ? 0 : 1,
      }}
    ></div>
  )}
</Transition>

```



## CSSTransition component

Handling the animation through different CSS class names which will be applied on different events.

```
import CSSTransition from 'react-transition-group/CSSTransition';
import './Modal.css';

Complexity is 6 It's time to do something...
const Modal = (props) => {
  const animationTiming = {
    enter: 400,
    exit: 1000
  }
  return (
    <CSSTransition
      mountOnEnter
      unmountOnExit
      in={props.show}
      timeout={animationTiming}
      classNames="fade-slide"
    >
      <div className="Modal">
        <h1>A Modal</h1>
        <button className="Button" onClick={props.closed}>
          | Dismiss
        </button>
      </div>
    </CSSTransition>
  );
};


```

```
.ModalOpen {
  animation: openModal 0.4s ease-out forwards;
}

.ModalClose {
  animation: closeModal 0.4s ease-out forwards;
}

.fade-slide-enter {
}

.fade-slide-enter-active {
  animation: openModal 0.4s ease-out forwards;
}

.fade-slide-exit {
}

.fade-slide-exit-active {
  animation: closeModal 0.4s ease-out forwards;
}
```

The code block shows five CSS classes: .ModalOpen, .ModalClose, .fade-slide-enter, .fade-slide-enter-active, and .fade-slide-exit. Three curly braces are drawn to group them: one brace groups .ModalOpen and .ModalClose together; another brace groups .fade-slide-enter and .fade-slide-enter-active together; and a third brace groups .fade-slide-exit and .fade-slide-exit-active together.

## Using a classnames in CSSTransition component

```
        return (
      <CSSTransition
        mountOnEnter
        unmountOnExit
        in={props.show}
        timeout={animationTiming}
        classNames={{
          enter: '',
          enterActive: "ModalOpen",
          exit: '',
          exitActive: "ModalClose",
        }}
      >
        <div className="Modal">
          <h1>A Modal</h1>
          <button className="Button" onClick={props.closed}>
            Dismiss
          </button>
        </div>
      </CSSTransition>
    );
  );
};
```

## Using TransitionGroup to manage multiple transitions

The screenshot shows a browser window with the URL `localhost:3000`. The page title is **React Animations**. On the left, there is a button labeled **Toogle** above a larger button labeled **Open Modal**. Below the modal button, the text **Animating Lists** is displayed. A button labeled **Add Item** is present. The text **Click Item to Remove.** is followed by a list of three items: **1**, **2**, and **3**, each enclosed in a separate box.

```
Complexity is 10 It's time to do something...
render () { 
  Complexity is 4 Everything is cool!
  const listItems = this.state.items.map((item, index) => ( 
    <CSSTransition key={index} classNames="fade" timeout={300}>
      <li
        className="ListItem"
        onClick={() => this.removeItemHandler(index)}
      >
        {item}
      </li>
    </CSSTransition>
  )));
}

return (
  <div>
    <button className="Button" onClick={this.addItemHandler}>
      Add Item
    </button>
    <p>Click Item to Remove.</p>
    <TransitionGroup component="ul" className="List">
      {listItems}
    </TransitionGroup>
  </div>
);
}
```

## Other React Animation Library

React Move

React Route Transition

## Replacing Redux with React Hooks

**Replacing Redux with Context +  
Hooks**

A Totally Optional Of Reducing Your  
Dependencies

```

import { useState, useEffect } from 'react';

let globalState = {};
let listeners = [];
let actions = {};

Complexity is 10 It's time to do something...
export const useStore = (shouldListen = true) => {
  const setState = useState(globalState)[1];

  const dispatch = (actionIdentifier, payload) => {
    const newState = actions[actionIdentifier](globalState, payload);
    globalState = { ...globalState, ...newState };

    for (const listener of listeners) {
      listener(globalState);
    }
  };
}

Complexity is 6 It's time to do something...
useEffect(() => {
  if (shouldListen) {
    listeners.push(setState);
  }
})

Complexity is 3 Everything is cool!
return () => {
  if (shouldListen) {
    listeners = listeners.filter((li) => li !== setState);
  }
}, [setState, shouldListen]);

return [globalState, dispatch];
};

export const initStore = (userActions, initialState) => {
  if (initialState) {
    globalState = { ...globalState, ...initialState };
  }
  actions = { ...actions, ...userActions };
};

```

```
import { initStore } from './store';

Complexity is 4 Everything is cool!
const configureStore = () => {
  const actions = {
    Complexity is 3 Everything is cool!
    TOGGLE_FAV: (curState, productId) => {
      const prodIndex = curState.products.findIndex(p => p.id === productId);
      const newFavStatus = !curState.products[prodIndex].isFavorite;
      const updatedProducts = [...curState.products];
      updatedProducts[prodIndex] = {
        ...curState.products[prodIndex],
        isFavorite: newFavStatus
      };
      return { products: updatedProducts };
    }
  };
  initStore(actions, {
    products: [
      {
        id: 'p1',
        title: 'Red Scarf',
        description: 'A pretty red scarf.',
        isFavorite: false
      },
      {
        id: 'p2',
        title: 'Blue T-Shirt',
        description: 'A pretty blue t-shirt.',
        isFavorite: false
      },
      {
        id: 'p3',
        title: 'Green Trousers',
        description: 'A pair of lightly green trousers.',
        isFavorite: false
      },
      {
        id: 'p4',
        title: 'Orange Hat',
        description: 'Street style! An orange hat.',
        isFavorite: false
      }
    ]
  });
}

export default configureStore;
```

# Testing React App



## Testing React Apps

Automated Testing

### Module Content

- What is “Testing”? And Why?
- Understanding Unit Tests
- Testing React Components & Building Blocks

What is Testing?

### What is “Testing”?

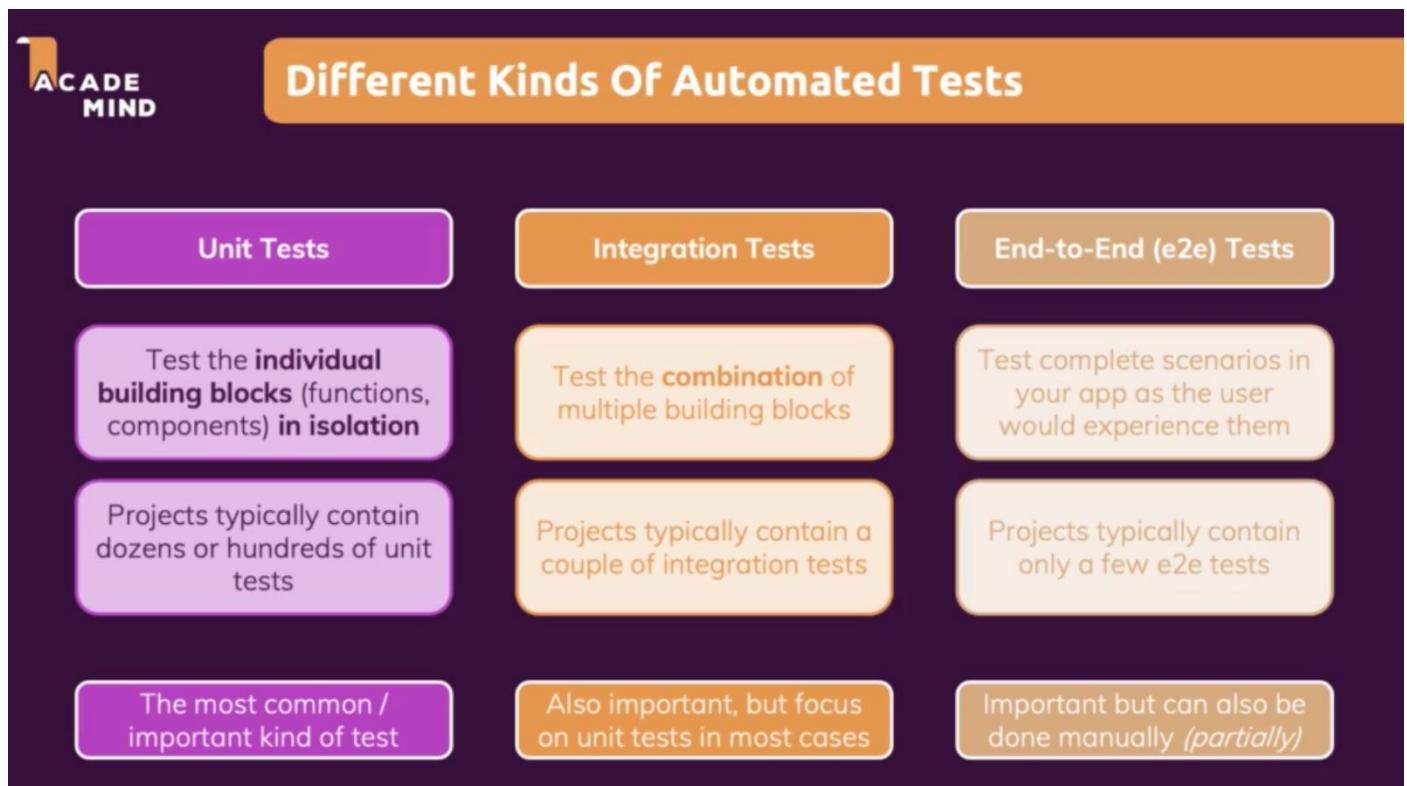
<b>Manual Testing</b>	<b>Automated Testing</b>
Write Code <> Preview & Test in Browser	Code that tests your code
Very important: You see what your users will see	You test the individual building blocks of your app

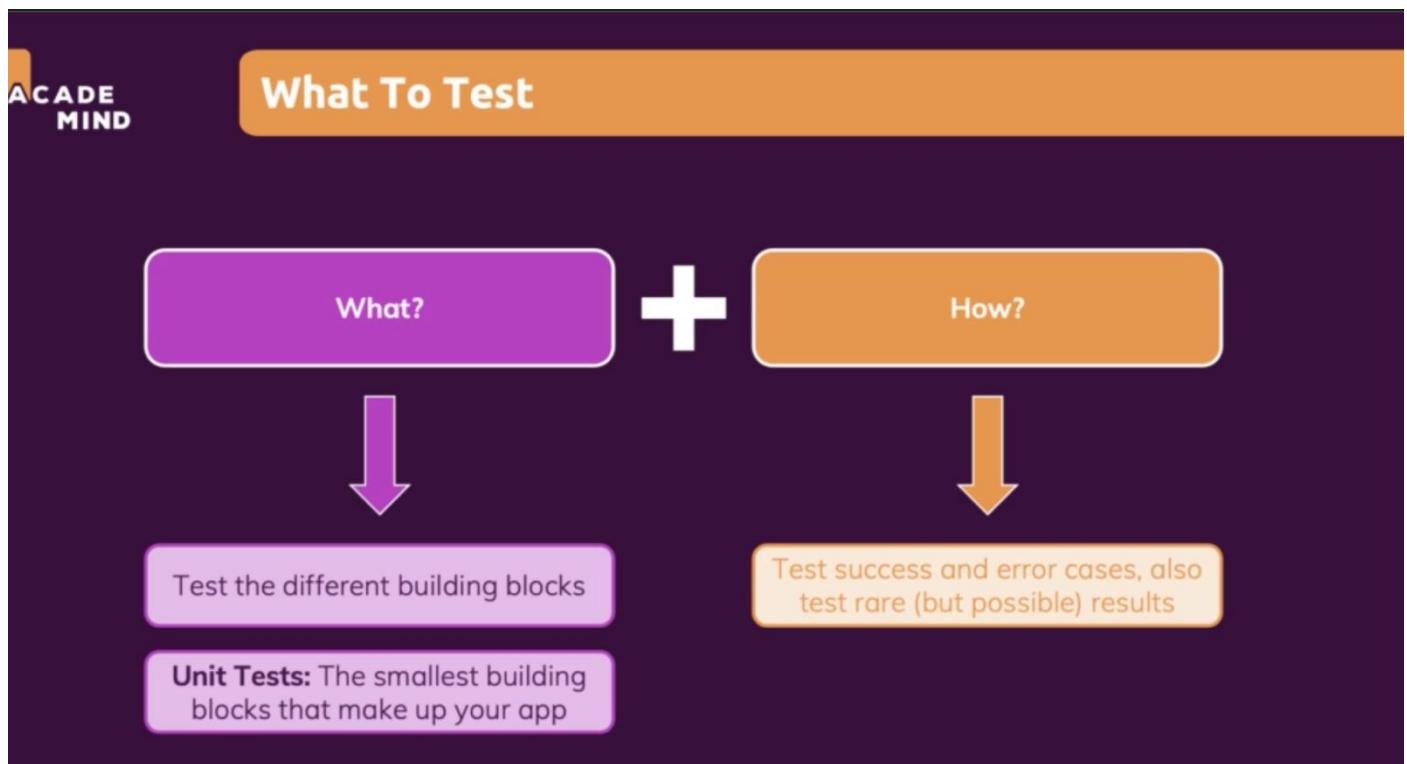
Error-prone: It's hard to test all possible combinations and scenarios

Very technical but allows you to test ALL building blocks at once

## Different Kind of Test



## What to test ?



Running the test

```
import { render, screen } from '@testing-library/react';
import App from './App';

test('renders learn react link', () => {
  render(<App />);
  const linkElement = screen.getByText(/learn react/i);
  expect(linkElement).toBeInTheDocument();
});
```

The screenshot shows a code editor with a package.json file open and a terminal window below it.

**package.json**

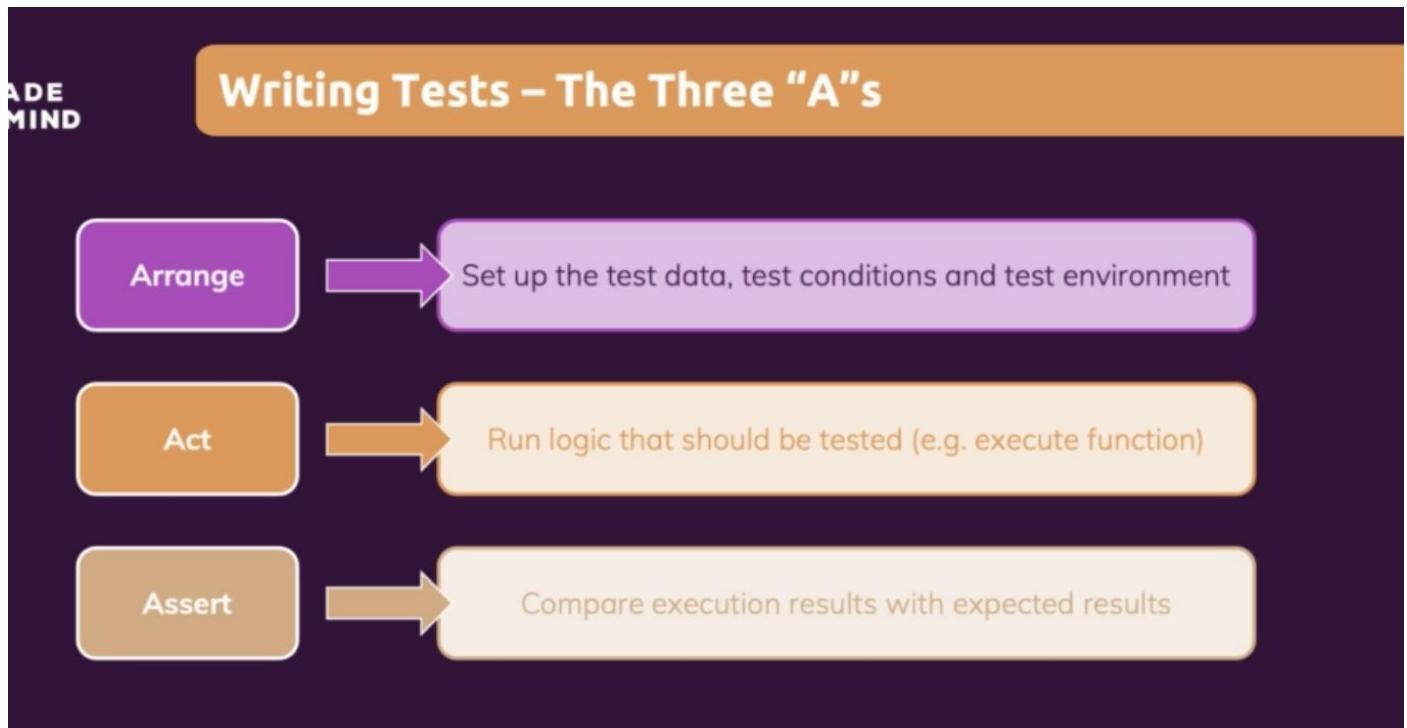
```
26-react-test > package.json > ...
  ...
  "scripts": {
    "start": "react-scripts start",
    "build": "react-scripts build",
    "test": "react-scripts test",
    "eject": "react-scripts eject"
  },
  "eslintConfig": {
    "extends": [
      "react-app"
    ]
  }
},
```

**TERMINAL**

```
PASS src/App.test.js
  ✓ renders learn react link (32 ms)

Test Suites: 1 passed, 1 total
Tests:       1 passed, 1 total
Snapshots:   0 total
Time:        7.451 s
Ran all test suites.

Watch Usage: Press w to show more.
→ 26-react-test git:(master) npm test[]
```



```
import { render, screen } from "@testing-library/react";
import Greeting from "./Greetings";

test("renders Hello World as a test", () => {
  //Arrange
  render(<Greeting />);

  //Act
  //... nothing

  //Assert
  const helloWorldElement = screen.getByText("Hello World!", { exact: false });
  expect(helloWorldElement).toBeInTheDocument();
});
```

Here exact is true out of the box.

Grouping test together with Test Suites

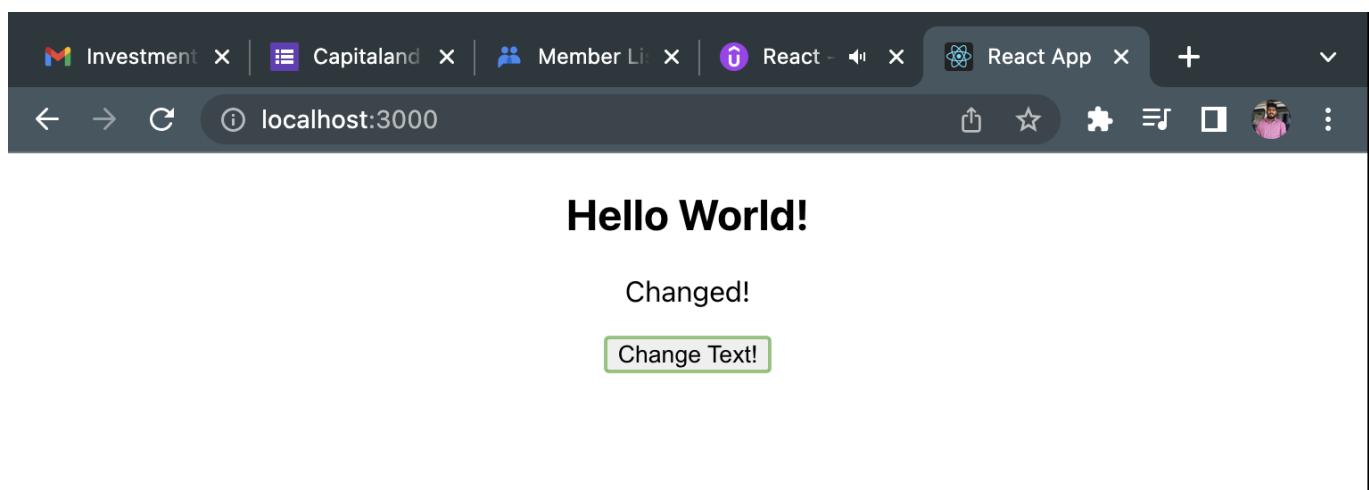
Describe function use to combine the test together.

```
Complexity is 3 Everything is cool!
describe('Greeting component', () => {
  test("renders Hello World as a test", () => {
    //Arrange
    render(<Greeting />);

    //Act
    //... nothing

    //Assert
    const holloWorldElement = screen.getByText("Hello World!", {
      exact: false,
    });
    expect(holloWorldElement).toBeInTheDocument();
  });
});
```

## Testing User Interaction and state



```
import { useState } from "react";

Complexity is 10 It's time to do something...
const Greeting = () => {
  const [changeText, setChangeText] = useState(false);

  const changeTextHandler = () => {
    setChangeText(true);
  };

  return (
    <div>
      <h2>Hello World!</h2>
      {!changeText && <p>It's good to see you</p>}
      {changeText && <p>Changed!</p>}
      <button onClick={changeTextHandler}>Change Text!</button>
    </div>
  );
}

export default Greeting;
```

```
test('renders "good to see" you if the button was not clicked', ()=>{
  render(<Greeting/>);
  const outputElement = screen.getByText("good to see you", {
    exact: false,
  });
  expect(outputElement).toBeInTheDocument();
});

test('renders "Changed!" if the button was clicked', ()=>{
  //Arrange
  render(<Greeting/>);

  //Act
  const buttonElement = screen.getByRole('button');
  userEvent.click(buttonElement)

  //Assert
  const outputElement = screen.getByText("Changed!");
  expect(outputElement).toBeInTheDocument();
});

test('should not renders "good to see" when "Changed!" had been rendered',
  //Arrange
  render(<Greeting/>);

  //Act
  const buttonElement = screen.getByRole('button');
  userEvent.click(buttonElement)

  //Assert
  const outputElement = screen.queryByText("good to see you", {
    exact: false,
  });
  expect(outputElement).toBeNull();
});

});
```

## Testing connected component

Here “Output” component will be rendered part of the Greeting component.

The screenshot shows a code editor with a dark theme. At the top, there is a status bar with icons for file operations. Below the status bar is a code editor window containing a file named 'Greeting.js'. The code defines a functional component 'Greeting' that uses the useState hook to manage a boolean state 'changeText'. It contains an H2 element and a button that triggers the 'changeTextHandler' function. The code is annotated with several yellow highlights: one on the first line, one on the useState hook, one on the condition in the return statement, and one on the 'Output' component tag. Line numbers 3 through 21 are visible on the left side of the code editor.

```
3 Complexity is 10 It's time to do something...
4 const Greeting = () => { █
5   const [changeText, setChangeText] = useState(false);
6
7   const changeTextHandler = () => {
8     setChangeText(true);
9   };
10
11   return (
12     <div>
13       <h2>Hello World!</h2>
14       {!changeText && <Output>It's good to see you</Output>}
15       {changeText && <Output>Changed!</Output>}
16       <button onClick={changeTextHandler}>Change Text!</button>
17     </div>
18   );
19 }
20
21 export default Greeting;
```

Below the code editor is a terminal window titled 'TERMINAL'. It shows the output of a Jest test run. The test passed, with 1 suite and 4 tests passed. The terminal also displays the execution environment as 'node 26-react...' and the shell as 'zsh'. The right side of the terminal window has a sidebar with a list of recent sessions.

```
PROBLEMS 1 OUTPUT TERMINAL ^ X
▽ TERMINAL >
PASS src/components/Greeting.test.js
  Greeting component
    ✓ renders Hello World as a test (24 ms)
    ✓ renders "good to see" you if the button was not clicked (3 ms)
    ✓ renders "Changed!" if the button was clicked (53 ms)
    ✓ should not renders "good to see" when "Changed!" had been rendered
      (22 ms)

Test Suites: 1 passed, 1 total
Tests:       4 passed, 4 total
Snapshots:   0 total
Time:        2.609 s
Ran all test suites related to changed files.

Watch Usage: Press w to show more. █
```

node 26-react...
zsh
node 26-react...
zsh
zsh

## Testing Asynchronous Code

```
import { useEffect, useState } from 'react';

Complexity is 9 It's time to do something...
const Async = () => {
  const [posts, setPosts] = useState([]);

Complexity is 3 Everything is cool!
useEffect(() => {
  fetch('https://jsonplaceholder.typicode.com/posts')
    .then((response) => response.json())
    .then((data) => {
      setPosts(data);
    });
}, []);

return (
  <div>
    <ul>
      {posts.map((post) => (
        <li key={post.id}>{post.title}</li>
      ))}
    </ul>
  </div>
);
};

export default Async;
```

```
import { render, screen } from "@testing-library/react";
import userEvent from '@testing-library/user-event';
import Async from "./Async";

Complexity is 3 Everything is cool!
describe('Async component', ()=> {
  test('renders posts if request succeeds', async ()=>{
    render(<Async />);
    const listItemElements = await screen.findAllByRole("listitem");
    expect(listItemElements).toHaveLength(0);
  });
});
```

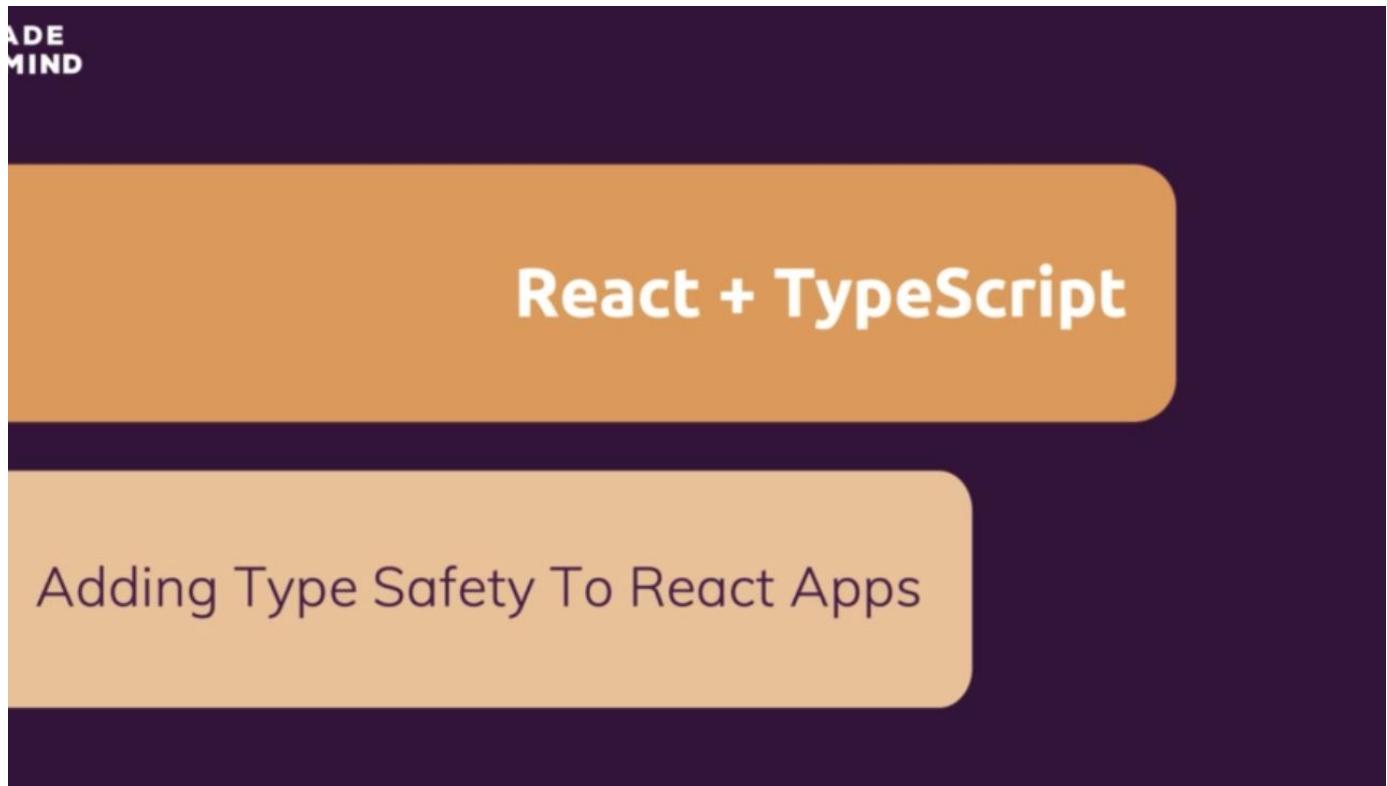
## Working with Mock

```
import { render, screen } from "@testing-library/react";
import userEvent from '@testing-library/user-event';
import Async from "./Async";

Complexity is 4 Everything is cool!
describe('Async component', ()=> { [REDACTED]
  Complexity is 3 Everything is cool!
  test('renders posts if request succeeds', async ()=>{ [REDACTED]
    window.fetch = jest.fn();
    window.fetch.mockResolvedValueOnce({
      json: async () => [{ id: "p1", title: "First post" }],
    });
    render(<Async />);
    const listItemElements = await screen.findAllByRole("listitem");
    expect(listItemElements).toHaveLength(0);
  });
});
```

<https://www.w3.org/TR/html-aria/#docconformance>

## React + Typescript



## Module Content

What & Why?

TypeScript Basics

Combining React & TypeScript

What and Why?

TypeScript is a “superset” to JavaScript

Extends the core JavaScript

JavaScript is dynamically typed. No need to specify a variable type ahead of time. JavaScript will recognize the type and act accordingly.

```
function add(a, b) {  
  return a + b;  
}  
  
const result = add(2, 5);  
  
console.log(result);
```

TypeScript is statically typed. We need to specify the type of a variable ahead of time. Since the TypeScript is statically typed, the application becomes less error prone.

As we can see in below JavaScript (dynamically typed language) example, variable "a" and "b" accepts both numbers and string.

```
function add(a, b) {
  return a + b;
}

const result = add('2', '5');

console.log(result);
```

As we can see in below TypeScript (dynamically typed language) example, variable "a" and "b" accepts only numbers. That's the reason we are getting an error in line number 5 because of string is being passed in place of number.

```
s with-typescript.ts ● index.html
1  function add(a: number, b: number) {
2    return a + b;
3  }
4
5  const result = add('2', '5');
6
7  console.log(result);
```

## Installing TypeScript

npm init -y

Create a package.json file.

```
→ 27-typescript-example git:(master) ✘ npm init -y
Wrote to /Users/naveen.kumar1/Workspace/WEB/REACT/27-typescript-example
/package.json:

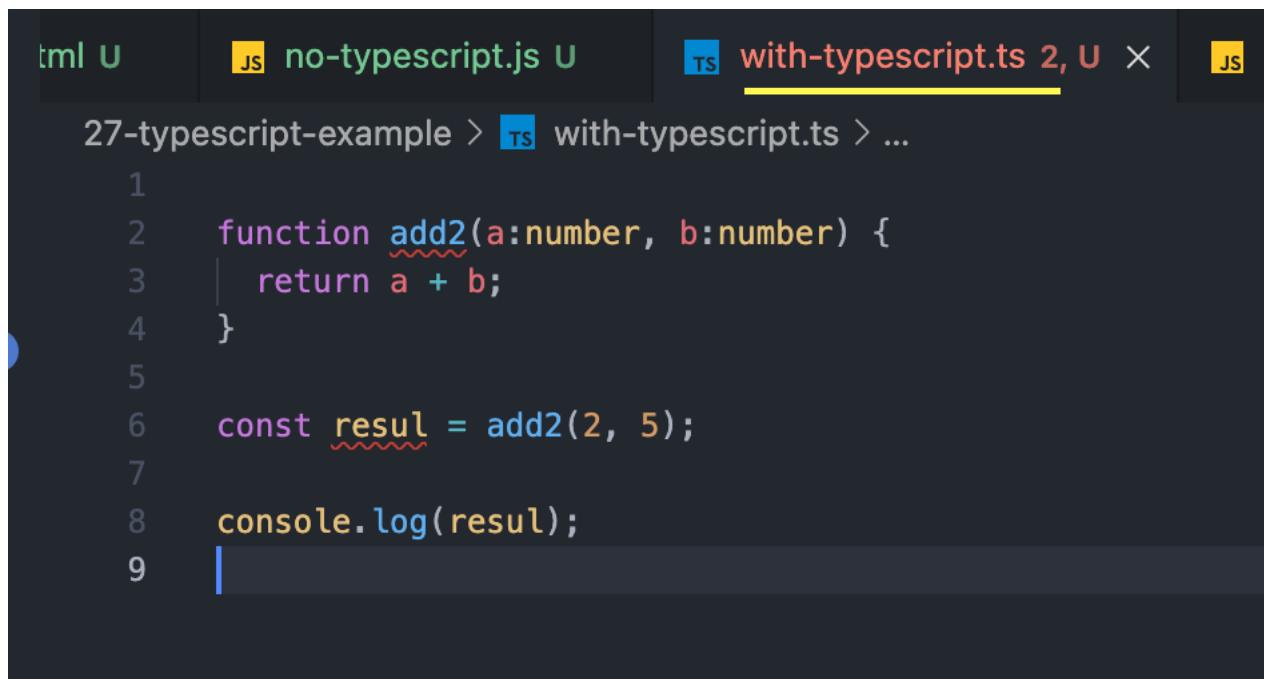
{
  "name": "27-typescript-example",
  "version": "1.0.0",
  "description": "",
  "main": "no-typescript.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [],
  "author": "",
  "license": "ISC"
}
```

```
npm install typescript
```

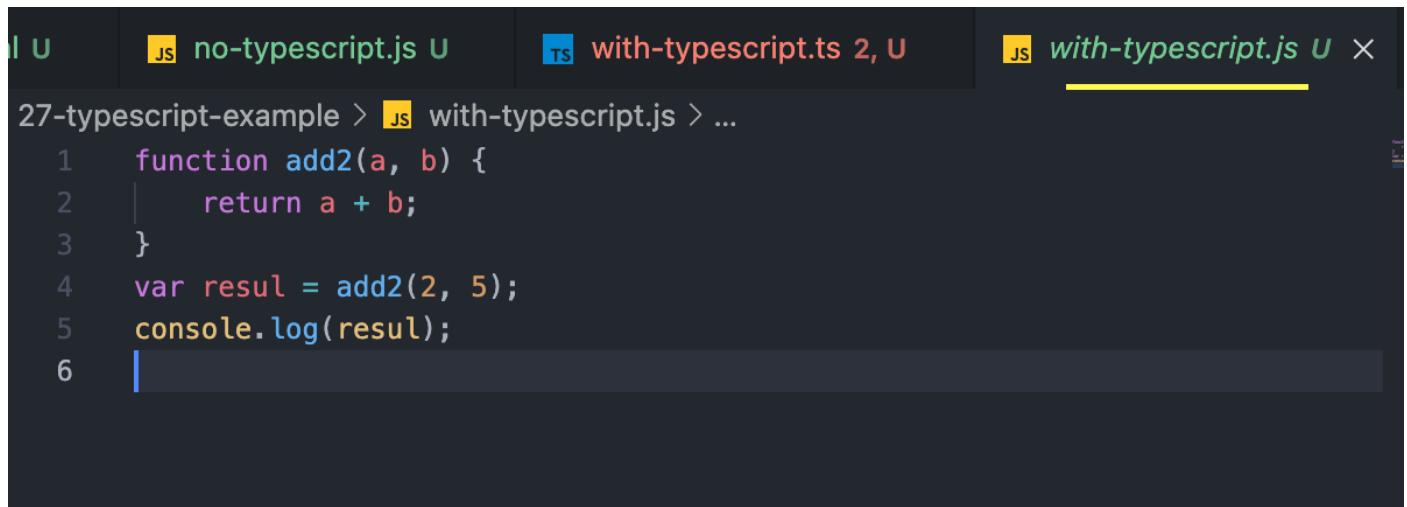
```
→ 27-typescript-example git:(master) ✘ npm install typescript
added 1 package, and audited 2 packages in 6s
found 0 vulnerabilities
→ 27-typescript-example git:(master) ✘
```

Compiling type script to JavaScript

```
→ 27-typescript-example git:(master) ✘ npx tsc with-typescript.ts
→ 27-typescript-example git:(master) ✘
```



```
27-typescript-example > ts with-typescript.ts > ...
1
2     function add2(a:number, b:number) {
3         return a + b;
4     }
5
6     const resul = add2(2, 5);
7
8     console.log(resul);
9
```



```
27-typescript-example > js with-typescript.js > ...
1     function add2(a, b) {
2         return a + b;
3     }
4     var resul = add2(2, 5);
5     console.log(resul);
6
```

## Base Types

```
//Primitives

let age: number = 23;

age = 12

let userName: string = 'naveen';

userName = 'naveen.kumar'

let isReactAwsome = true;
```

## Array and Objects

```
//Array
let courses: string[];

courses= ['React','Datastructure','Dimension Modeling']

//Object
let person: {
  name: string;
  age: number;
};

person = {
  name : 'naveen',
  age : 26
}

//Object Array
let people: {
  name: string;
  age: number;
}[];

people = [
  (person = {
    name: "naveen",
    age: 26,
  }),
  (person = {
    name: "kumar",
    age: 26,
  }),
];
```

## Type Inference

```
//Type inference

let course = 'ss';
//Here course is initialized without explicitly specifying a type.
//so course variable is inferred as string type.
//It will give an error if we are type to assign value which is
//other than a string.
//course = 22;
```

## Union Type

```
//Union Type
|
let unionType: string | number | boolean;
unionType = "a";
unionType = 1;
unionType = true;
```

## Type alias

```
type Person = {
  name: string;
  age: number;
}

//Object
let person: Person;

person = {
  name : 'naveen',
  age : 26
}

//Object Array
let people: Person[];

people = [
  (person = {
    name: "naveen",
    age: 26,
  }),
  (person = {
    name: "kumar",
    age: 26,
  }),
];
```

## Function and Function Types

```
//Function and Types

const add = (a: number, b: number): number => {
  return a + b; _____
};

const printOutput = (value: any): void => {
  console.log(value); _____
};
```

## Generics

```
//Generics

Complexity is 4 Everything is cool!
const insertAtBeginnings = (array: any[], value: any) => { _____
  const newArray = [value,...array];
  return newArray;
};

const demoArray = [1,2,3];

const updatedArray = insertAtBeginnings(demoArray, -1);
```

```
//Generics

const insertAtBeginnings= <T>(array: T[], value: T) => {
  const newArray = [value,...array];
  return newArray;
};

const demoArray = [1,2,3];

const updatedArray = insertAtBeginnings(demoArray, -1);
```

```
//Generics

const insertAtBeginnings = <T>(array: T[], value: T) => {
  const newArray = [value, ...array];
  return newArray;
};

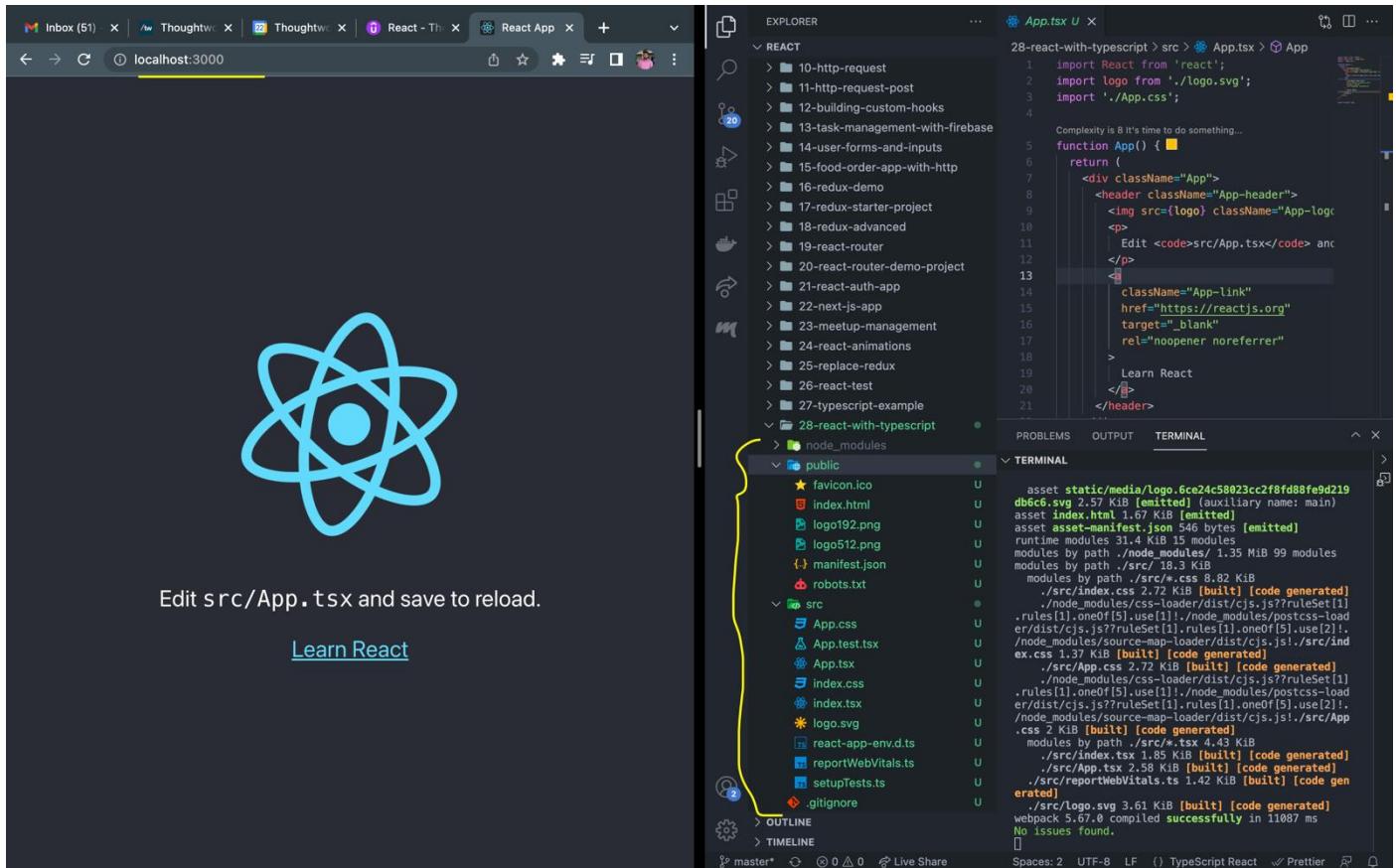
const updatedArray = insertAtBeginnings([1, 2, 3], -1);
const stringArray = insertAtBeginnings(["a", "b", "c"], "d");
```

## Working with React TypeScript Project

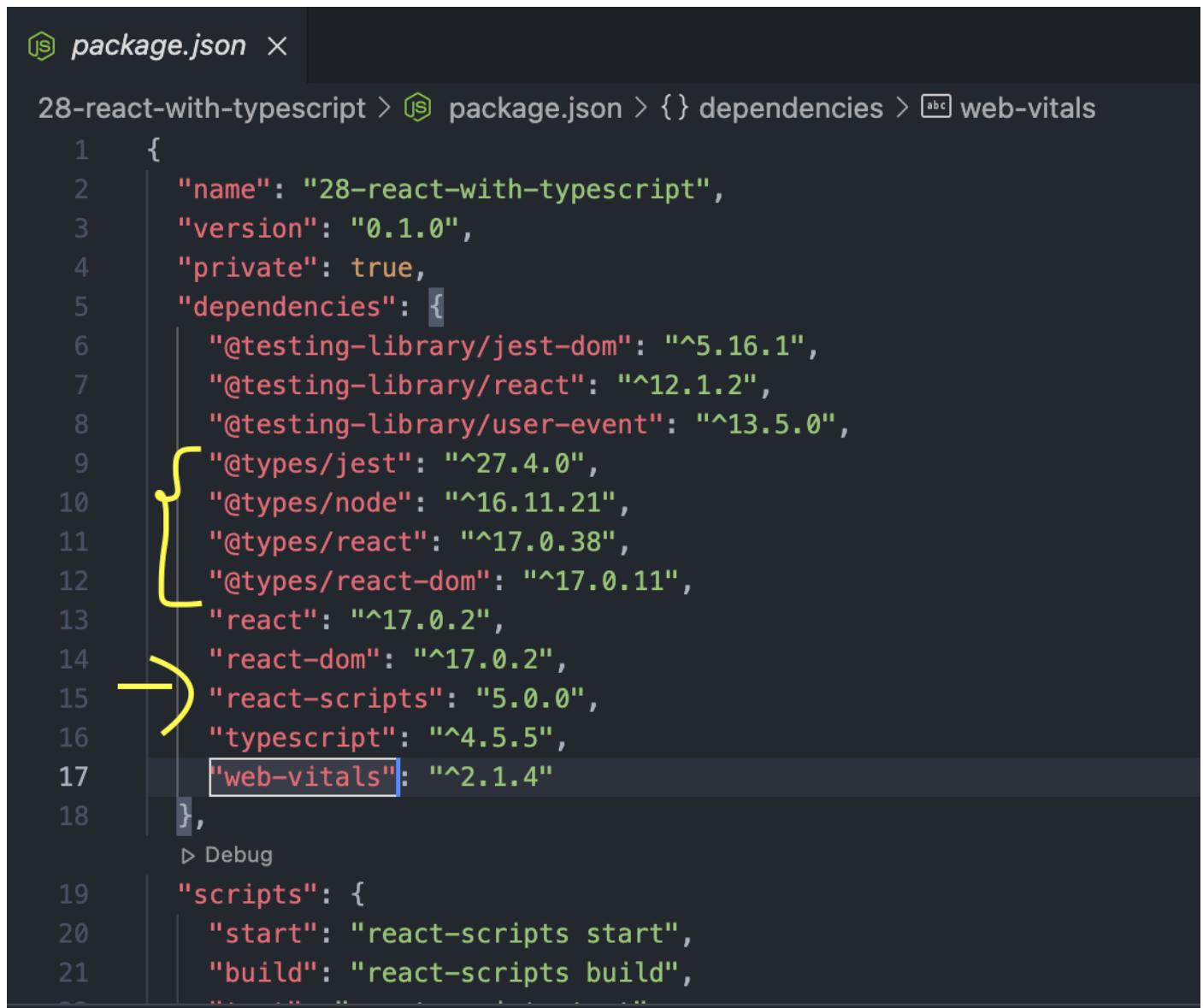
### Creating a project

```
→ REACT git:(master) npx create-react-app 28-react-with-typescript --template typescript
Creating a new React app in /Users/naveen.kumar1/Workspace/WEB/REACT/28-react-with-typescript
.

Installing packages. This might take a couple of minutes.
Installing react, react-dom, and react-scripts with cra-template-typescript...
(( )) :: reify:tempy: http fetch GET 200 https://registry.npmjs.org/tempy/-/te
```



To run the typescript code, compiler internally convert the typescript code to JavaScript code. Below libraries from package.json file helps to do the conversion.



```
package.json ×

28-react-with-typescript > package.json > {} dependencies > web-vitals

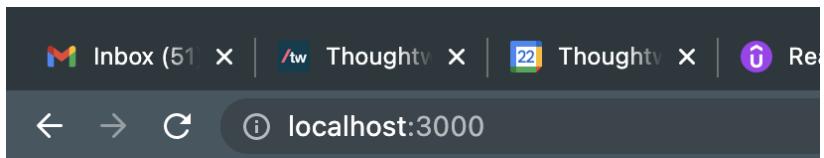
1  {
2    "name": "28-react-with-typescript",
3    "version": "0.1.0",
4    "private": true,
5    "dependencies": {
6      "@testing-library/jest-dom": "^5.16.1",
7      "@testing-library/react": "^12.1.2",
8      "@testing-library/user-event": "^13.5.0",
9      "@types/jest": "^27.4.0",
10     "@types/node": "^16.11.21",
11     "@types/react": "^17.0.38",
12     "@types/react-dom": "^17.0.11",
13     "react": "^17.0.2",
14     "react-dom": "^17.0.2",
15     "react-scripts": "5.0.0",
16     "typescript": "^4.5.5",
17     "web-vitals": "^2.1.4"
18   },
19   "scripts": {
20     "start": "react-scripts start",
21     "build": "react-scripts build",
22     "eject": "react-scripts eject"
23   }
}
```

## Working with component and TypeScript



```
Complexity is 5 Everything is cool!
const Todos = () => {
  return (
    <ul>
      <li>Learn React</li>
      <li>Learn Typescript</li>
    </ul>
  );
}

export default Todos;
```



- Learn React
- Learn Typescript

## Working with props and Typescript

```
Complexity is 3 Everything is cool!
const Todos: React.FC = (props) => {
  return (
    <ul>
      {props.children}
    </ul>
  );
};

export default Todos;
```

Here, `React.FC` is type annotation which tells that `Todos` is functional component. Because of this, `props` contain `children` attribute by default.

```
type TodoData = {
  items: string[]
}

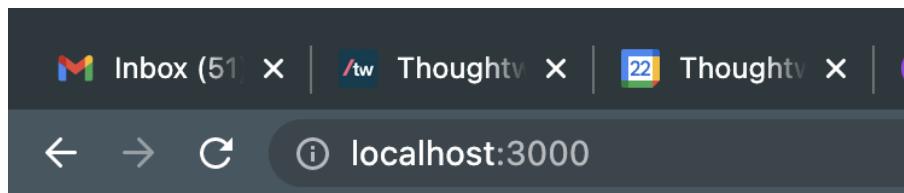
Complexity is 5 Everything is cool!
const Todos: React.FC<TodoData> = (props) => {
  return (
    <ul>
      {props.items.map((item) => (
        <li key={item}>{item}</li>
      ))}
    </ul>
  );
};

export default Todos;
```

If we specify the React.FC alone, it will be more generic type. It contains only default attributes of props. Like for example children props. To specify the props attribute that we want for functional component we specifying it as React.FC<TodoData>. By specifying this, TodoData attribute will be merged with default attributes for props.

The screenshot shows a code editor with several tabs at the top: 'M' (marked), App.tsx (active), Todos.tsx, basics.ts, and App.css. The code in App.tsx is as follows:

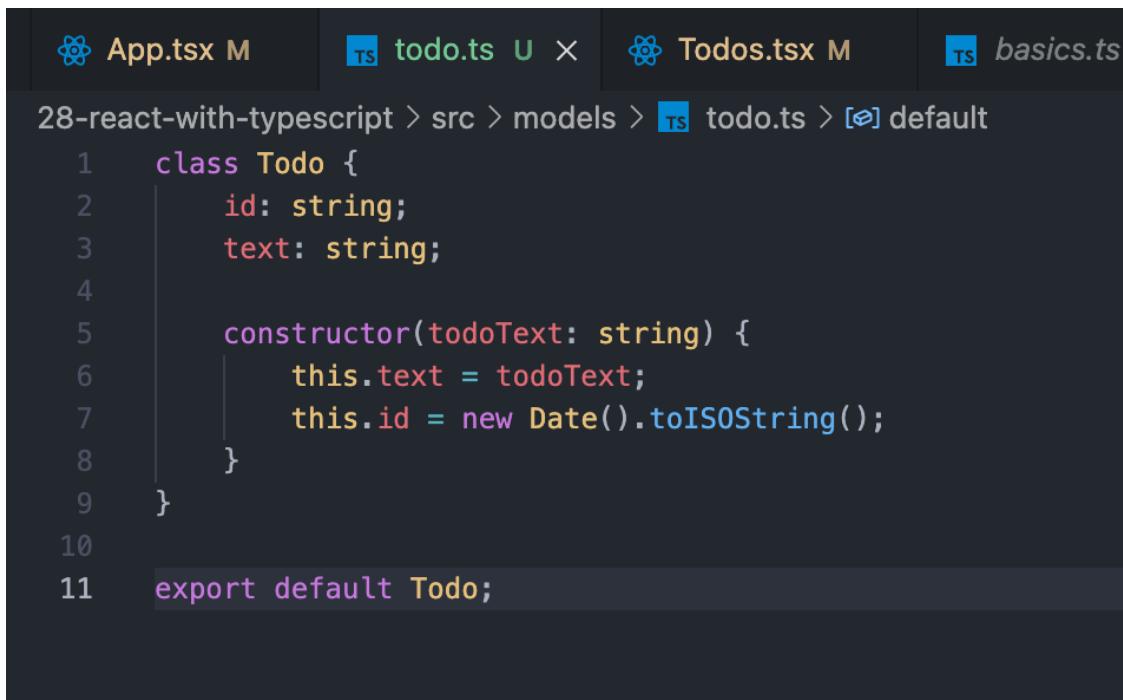
```
1 import React from 'react';
2 import './App.css';
3 import Todos from './components/Todos';
4
5 const TODO_DATA = ['Learn React', 'Learn Typescript'];
6
7 Complexity is 4 Everything is cool!
8 const App = () => {
9   return (
10     <div className="App">
11       <Todos items={TODO_DATA} />
12     </div>
13   );
14
15 export default App;
16
```



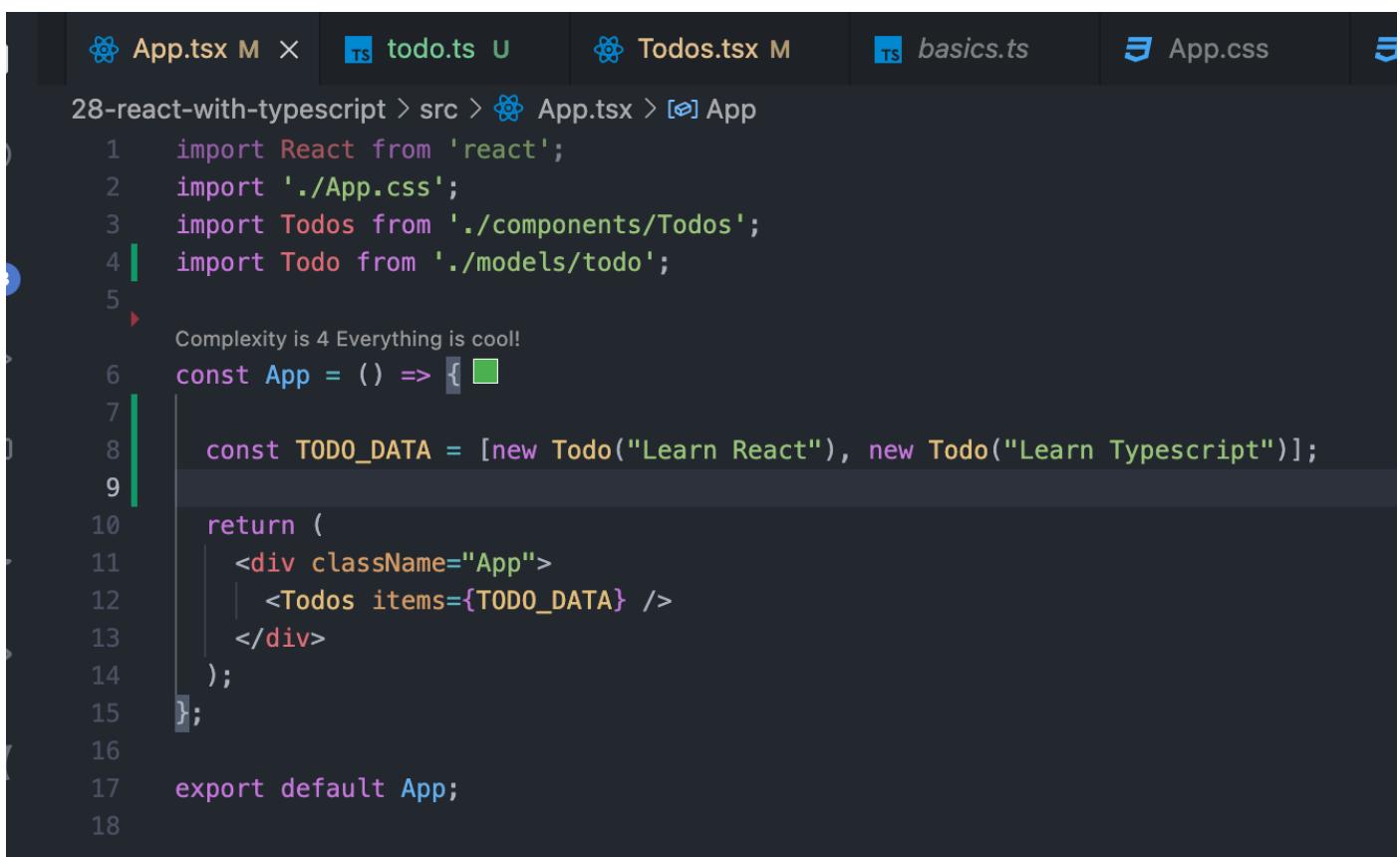
- Learn React
- Learn Typescript

## Adding a Data Model

Adding a data model to represent a Todo data.



```
App.tsx M todo.ts U Todos.tsx M basics.ts
28-react-with-typescript > src > models > todo.ts > [?] default
1  class Todo {
2      id: string;
3      text: string;
4
5      constructor(todoText: string) {
6          this.text = todoText;
7          this.id = new Date().toISOString();
8      }
9  }
10
11 export default Todo;
```



```
App.tsx M X todo.ts U Todos.tsx M basics.ts App.css
28-react-with-typescript > src > App.tsx > [?] App
1 import React from 'react';
2 import './App.css';
3 import Todos from './components/Todos';
4 import Todo from './models/todo';
5
6 Complexity is 4 Everything is cool!
7 const App = () => {
8
9     const TODO_DATA = [new Todo("Learn React"), new Todo("Learn Typescript")];
10
11     return (
12         <div className="App">
13             <Todos items={TODO_DATA} />
14         </div>
15     );
16
17     export default App;
18 }
```

```
28-react-with-typescript > src > components > Todos.tsx > [o] Todos > [?] props.ite
1 | import Todo from "../models/todo";
2 |
3 | type TodoData = {
4 |   items: Todo[]
5 | }
6 | Complexity is 5 Everything is cool!
7 | const Todos: React.FC<TodoData> = (props) => {
8 |   return (
9 |     <ul>
10 |       {props.items.map((item) => (
11 |         <li key={item.id}>{item.text}</li>
12 |       ))}
13 |     </ul>
14 |   );
15 |
16 | export default Todos;
17 |
```

Sperate out by Adding a TodoItem component

```
import React from 'react';

type Todo = {
  todo: string;
}

Complexity is 4 Everything is cool!
const TodoItem: React.FC<Todo> = (props) => {
  return (
    <div>
      <p>{props.todo}</p>
    </div>
  );
}

export default TodoItem;
```

```

import Todo from "../models/todo";
import TodoItem from './TodoItem';

type TodoData = {
  items: Todo[]
}

Complexity is 5 Everything is cool!
const Todos: React.FC<TodoData> = (props) => {
  return (
    <ul>
      {props.items.map((item) => (
        <TodoItem todo={item.text} key={item.id} />
      ))}
    </ul>
  );
};

export default Todos;

```

As we can in TodoItem props, key can be added even though it not specified in Todo type. “key” props is default to React.FC type annotation. So, it is merging it.

## Working with ref, useRef and Function Props

Below example also explains the way for including the props in props.

```
act-with-typescript / src / components / ↴ NewTodo.js ↴ default
import React, { useRef } from 'react';

type TodoAddition = {
  onAddTodo: (text: string) => void;
};

Complexity is 9 It's time to do something...
const NewTodo: React.FC<TodoAddition> = (props) => {
  const todoTextInputRef = useRef<HTMLInputElement>(null);

  Complexity is 3 Everything is cool!
  const submitHanlder = (event: React.FormEvent) => {
    event.preventDefault();

    const enteredText = todoTextInputRef.current!.value;

    if (enteredText.trim().length === 0) {
      return;
    }

    props.onAddTodo(enteredText);
  };

  return (
    <form onSubmit={submitHanlder}>
      <label htmlFor="text">Todo text</label>
      <input type="text" id="text" ref={todoTextInputRef} />
      <button>Add Todo</button>
    </form>
  );
};

export default NewTodo;
```

## Managing State

```
import React, { useEffect, useState } from 'react';
import './App.css';
import NewTodo from './components/NewTodo';
import Todos from './components/Todos';
import Todo from './models/todo';

const TODO_DATA = [new Todo("Learn React"), new Todo("Learn Typescript")];

Complexity is 9 It's time to do something...
const App = () => {

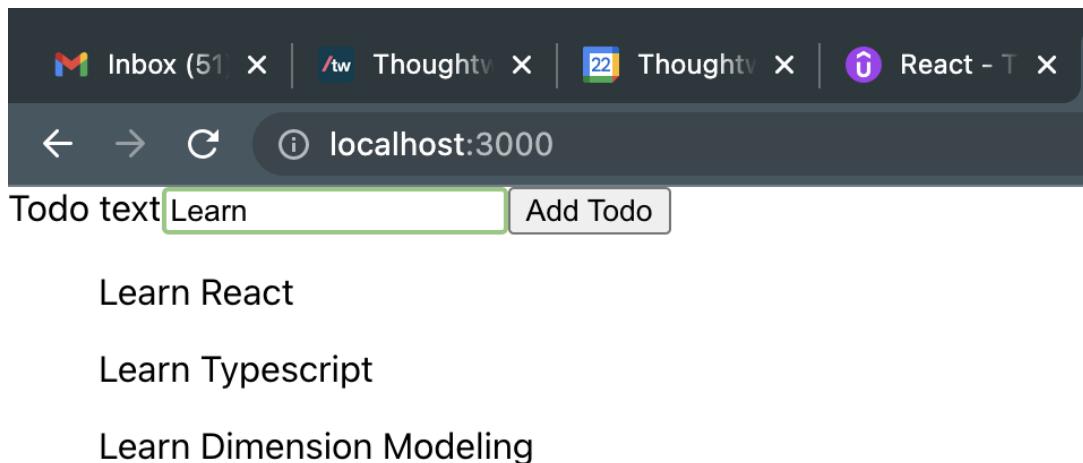
  const [todos, setTodos] = useState<Todo[]>(TODO_DATA);

  useEffect(() => {}, []);

  Complexity is 3 Everything is cool!
  const addTodoHandler = (todoText: string) => {
    const newTodo = new Todo(todoText);
    setTodos((prevTodos) => {
      return prevTodos.concat(newTodo);
    });
  };

  return (
    <div className="App">
      <NewTodo onAddTodo={addTodoHandler} />
      <Todos items={todos} />
    </div>
  );
};

export default App;
```



## Removing Todo Item

```
import React, { useEffect, useState } from 'react';
import './App.css';
import NewTodo from './components/NewTodo';
import Todos from './components/Todos';
import Todo from './models/todo';

Complexity is 13 You must be kidding
const App = () => { █

  const [todos, setTodos] = useState<Todo[]>([]);

  useEffect(() => {}, []);

Complexity is 4 Everything is cool!
const onRemoveTodoHandler = (todoId: string) => { █
  Complexity is 3 Everything is cool!
  setTodos((prevTodos) => { █
    return prevTodos.filter((todo) => todo.id !== todoId);
  });
};

Complexity is 3 Everything is cool!
const addTodoHandler = (todoText: string) => { █
  const newTodo = new Todo(todoText);
  setTodos((prevTodos) => {
    return prevTodos.concat(newTodo);
  });
};

return (
  <div className="App">
    <NewTodo onAddTodo={addTodoHandler} />
    <Todos items={todos} onRemoveTodo={onRemoveTodoHandler} />
  </div>
);
};

export default App;
```



- Here we are doing the bind on “onRemoveTodo” function props.
- Here, “item.id” will be passed as “todoid” parameter which is required for onRemoveTodo function props.
- So, this help us to provide the future parameter reference to a function.

```
import Todo from "../models/todo";
import TodoItem from './TodoItem';
import classes from './Todos.module.css';

type TodoData = {
  items: Todo[];
  onRemoveTodo: (todoId: string) => void
}
```

Complexity is 5 Everything is cool!

```
const Todos: React.FC<TodoData> = (props) => {
```

```
  return (
    <ul className={classes.todos}>
      {props.items.map((item) => (
        <TodoItem
          todo={item.text}
          key={item.id}
          onRemoveTodo={props.onRemoveTodo.bind(null, item.id)}
        />
      ))}
    </ul>
  );
}
```

```
export default Todos;
```

- Since we are using the bind, TodoItem does not required to pass a todo id which is required for performing the delete operation. (Beside we don't have todo id from TodoItem component. That's why we are following this approach)
- As we can see it the below screenshot, actual on click operation is happening in TodoItem component. But the reference of, which TodoItem has been clicked, is taken from Todos component using bind approach.

```
react-with-typescript > src > components > TodoItem.tsx > [d] default
```

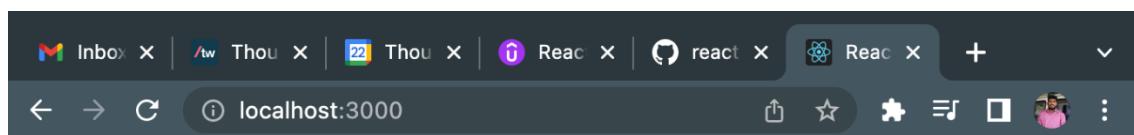
```
import React from 'react';
import classes from './TodoItem.module.css';

type Todo = {
  todo: string;
  onRemoveTodo: () => void;
};
```

Complexity is 3 Everything is cool!

```
const TodoItem: React.FC<Todo> = (props) => {
  return (
    <li className={classes.item} onClick={props.onRemoveTodo}>
      {props.todo}
    </li>
  );
}

export default TodoItem;
```



**Todo text**

Learn Dimension Modeling

Add Todo

Learn React

Learn Dimension Modeling

Learn Dimension Modeling

Learn Dimension Modeling

## Using Context API

```
TodoItem.tsx todo-context.tsx M X App.tsx M Todos.tsx M NewTodo.tsx M ⌂ ⌂ ...  
28-react-with-typescript > src > store > todo-context.tsx > [o] TodosContextProvider  
1 import React, { useState } from 'react';  
2 import Todo from '../models/todo';  
3  
4 type TodoContextType = {  
5   items: Todo[];  
6   addTodo: (text: string) => void;  
7   removeTodo: (id: string) => void;  
8 }  
9  
10 export const TodoContext = React.createContext<TodoContextType>({  
11   items: [],  
12   addTodo: () => {},  
13   removeTodo: (id: string) => {},  
14 });  
15  
Complexity is 10 It's time to do something...  
16 const TodosContextProvider: React.FC = (props) => {  
17   const [todos, setTodos] = useState<Todo[]>([]);  
18  
Complexity is 4 Everything is cool!  
19   const removeTodoHandler = (todoId: string) => {  
20     Complexity is 3 Everything is cool!  
21     setTodos((prevTodos) => {  
22       return prevTodos.filter((todo) => todo.id !== todoId);  
23     });  
24  
Complexity is 3 Everything is cool!  
25   const addTodoHandler = (todoText: string) => {  
26     const newTodo = new Todo(todoText);  
27     setTodos((prevTodos) => {  
28       return prevTodos.concat(newTodo);  
29     });  
30   };  
31  
32   const contextValue: TodoContextType = {  
33     items: todos,  
34     addTodo: addTodoHandler,  
35     removeTodo: removeTodoHandler,  
36   };  
37  
38   return (  
39     <TodoContext.Provider value={contextValue}>  
40       {props.children}  
41     </TodoContext.Provider>  
42   );  
43  
44  
45 export default TodosContextProvider;
```

```

src-with-typescript > src > App.tsx > ...
import React from "react";
import "./App.css";
import NewTodo from "./components/NewTodo";
import Todos from "./components/Todos";
import TodosContextProvider from "./store/todo-context";

Complexity is 6 It's time to do something...
const App = () => { █
  return (
    <TodosContextProvider>
      <div className="App">
        <NewTodo />
        <Todos />
      </div>
    </TodosContextProvider>
  );
};

export default App;

```

```

import React, { useContext, useRef } from 'react';
import classes from './NewTodo.module.css';
import { TodoContext } from "../store/todo-context";

Complexity is 9 It's time to do something...
const NewTodo: React.FC = () => { █
  const todoTextInputRef = useRef<HTMLInputElement>(null);
  const todoCtx = useContext(TodoContext);

Complexity is 3 Everything is cool!
  const submitHanlder = (event: React.FormEvent) => { █
    event.preventDefault();

    const enteredText = todoTextInputRef.current!.value;

    if (enteredText.trim().length === 0) {
      return;
    }

    todoCtx.addTodo(enteredText);
  };

  return (
    <form onSubmit={submitHanlder} className={classes.form}>
      <label htmlFor="text">Todo text</label>
      <input type="text" id="text" ref={todoTextInputRef} />
      <button>Add Todo</button>
    </form>
  );
};

export default NewTodo;

```

```
import TodoItem from "./TodoItem";
import classes from "./Todos.module.css";
import { TodoContext } from "../store/todo-context";
import { useContext } from "react";

Complexity is 5 Everything is cool!
const Todos: React.FC = () => {
  const todoCtx = useContext(TodoContext);

  return (
    <ul className={classes.todos}>
      {todoCtx.items.map((item) => (
        <TodoItem
          todo={item.text}
          key={item.id}
          onRemoveTodo={todoCtx.removeTodo.bind(null, item.id)}
        />
      ))}
    </ul>
  );
};

export default Todos;
```

tsconfig.json

This file take care of the configuration for typescript. Like libraries to manage etc.

```
{  
  "compilerOptions": {  
    "target": "es5",  
    "lib": [  
      "dom",  
      "dom.iterable",  
      "esnext"  
    ],  
    "allowJs": true,  
    "skipLibCheck": true,  
    "esModuleInterop": true,  
    "allowSyntheticDefaultImports": true,  
    "strict": true,  
    "forceConsistentCasingInFileNames": true,  
    "noFallthroughCasesInSwitch": true,  
    "module": "esnext",  
    "moduleResolution": "node",  
    "resolveJsonModule": true,  
    "isolatedModules": true,  
    "noEmit": true,  
    "jsx": "react-jsx"  
},  
  "include": [  
    "src"  
  ]  
}
```

## Module Content

What are “React Hooks”?

Using React Hooks With Functional Components

Rules of Hooks & Custom Hooks

What are React Hooks ?

## What are “React Hooks”?

React Hooks are **special functions** that can **only be used in functional components** (and custom hooks)



useState(), useEffect(), useXYZ()

These functions add extra capabilities to functional components

For example, useState() adds functionality that allows functional components to manage internal state

For class-based component, state is always an object.

## useState Hook

```
Complexity is 10 You must be kidding
const IngredientForm = React.memo(props)  Invitation from Tanmay Pereira Naik
                                            This Thursday at 17:00
                                            
    const [input, setInput] = useState({ title: "", amount: "" });

    console.log(input);

    const titleChangeHandler = (event) => {
        setInput({
            title: event.target.value,
            amount: input.amount
        });
    }

    const amountChangeHandler = (event) => {
        setInput({
            title: input.title,
            amount: event.target.value,
        });
    }

    const submitHandler = event => {
        event.preventDefault();
        // ...
    };

    return (
        <section className="ingredient-form">
            <Card>
                <form onSubmit={submitHandler}>
                    <div className="form-control">
                        <label htmlFor="title">Name</label>
                        <input type="text" id="title" value={input.title} onChange={titleChangeHandler}>
                    </div>
                    <div className="form-control">
                        <label htmlFor="amount">Amount</label>
                        <input type="number" id="amount" value={input.amount} onChange={amountChangeHandler}>
                    </div>
                    <div className="ingredient-form__actions">
                        <button type="submit">Add Ingredient</button>
                    </div>
                </form>
            </Card>
        </section>
    );
}

export default IngredientForm;
```

## SetStateFunction with Previous state

Here, we are getting the error because we are using the event object inside the anonymous function. Because of this reason, event object will be locked inside the set state function. The subsequent keystroke followed by first keystroke, try to reuse the first event object which has been locked inside. This in turn cause the error.

The screenshot shows a browser developer tools console on the left and a code editor on the right. The browser console lists several errors:

- A warning about synthetic events being reused for performance reasons.
- An uncaught TypeError: Cannot read properties of null (reading 'value').

The code editor shows the `IngredientForm.js` file:

```
1 import React, { useState } from 'react';
2 import Card from '../UI/Card';
3 import './IngredientForm.css';
4
5 Complexity is 17 You must be kidding
6 const IngredientForm = React.memo(props => {
7   const [input, setInput] = useState({ title: "", amount: "" });
8
9   console.log(input);
10
11   const titleChangeHandler = (event) => {
12     setInput((prevInputState) => ({
13       title: event.target.value,
14       amount: prevInputState.amount,
15     }));
16   }
17
18   const amountChangeHandler = (event) => {
19     setInput({
20       title: input.title,
21       amount: event.target.value,
22     });
23   }
24
25   const submitHandler = event => {
26     event.preventDefault();
27     // ...
28   };
29
30});
```

The terminal at the bottom of the code editor shows "Compiled successfully!"

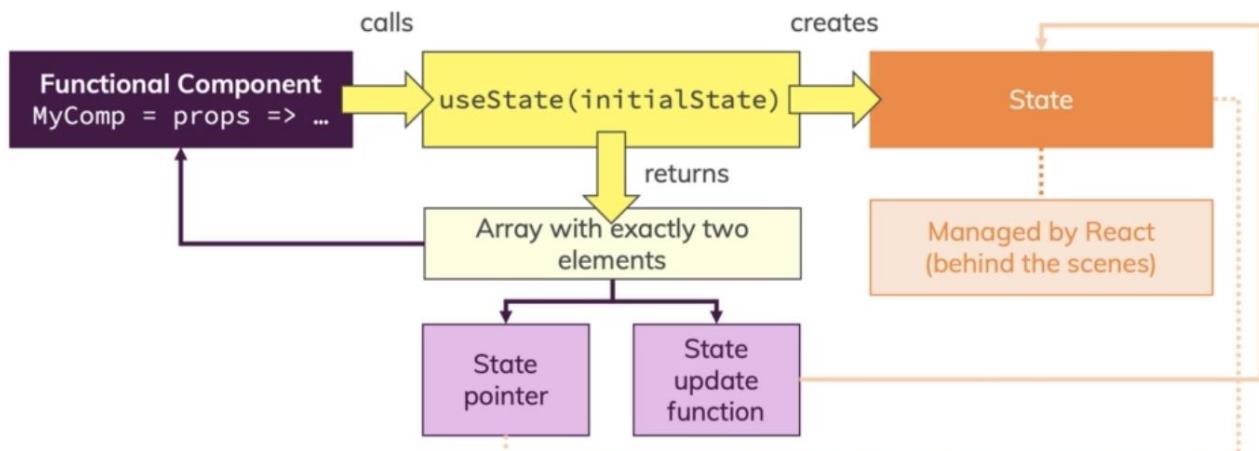
Solution is to refer to the new event object value.

```
const titleChangeHandler = (event) => {
  const newTitle = event.target.value;
  setInput((prevInputState) => ({
    title: newTitle,
    amount: prevInputState.amount,
  }));
}

const amountChangeHandler = (event) => {
  const newAmount = event.target.value;
  setInput((prevInputState) => ({
    title: prevInputState.title,
    amount: newAmount,
  }));
}
```



## Understanding “useState”



## setState & State Update Batching

setNewSet(oldState + 1)

vs

setNewSet(prevState => prevState + 1)

React batches state updates!

All state updates from one and the same synchronous event handler are batched together!

After setNewState(), you can't immediately use the new state when NOT using the function form!

# More on State Batching & State Updates

React batches state updates -

see: <https://github.com/facebook/react/issues/10231#issuecomment-316644950>

That simply means that calling

```
1 | setName('Max');
2 | setAge(30);
```

in the same synchronous (!) execution cycle (e.g. in the same function) will **NOT trigger two component re-render cycles**.

Instead, the component will **only re-render once** and both state updates will be **applied simultaneously**.

Not directly related, but also sometimes misunderstood, is when the new state value is available.

Consider this code:

```
1 | console.log(name); // prints name state, e.g. 'Manu'
2 | setName('Max');
3 | console.log(name); // ??? what gets printed? 'Max'?
```

You could think that accessing the name state after `setName('Max');` should yield the new value (e.g. `'Max'`) but this is **NOT the case**. Keep in mind, that the **new state value is only available in the next component render cycle** (which gets scheduled by calling `setName()`).

**Both concepts (batching and when new state is available) behave in the same way for both functional components with hooks as well as class-based components with `this.setState()`!**

```
const throwError = useCallback(() => {
  dispatchHttp({ type: "CLEAR" });
}, []);

Complexity is 3 Everything is cool!
const ingredientList = useMemo(()=>{ 
  return (
    <IngredientList
      ingredients={userIngredients}
      onRemoveItem={removeIngredientHandler}
    />
  );
}, [userIngredients, removeIngredientHandler]);

return (
  <div className="App">
    {httpState.error && (
      <ErrorModel onClose={clearError}>{httpState.error}</ErrorModel>
    )}
    <IngredientForm
      onAddIngredient={addIngredientHandler}
      loading={httpState.loading}
    />

    <section>
      <Search onLoadIngredients={filteredIngredientsHandler} />
      {ingredientList}
    </section>
  </div>
);
};
```

## React Summary

**ACADE MIND**

### Module Content

- What & Why?
- Core Concepts: Components, Props, State, React Hooks
- Full Demo Project & Advanced Concepts (Effects, Context)

Udemy

## What is React ?

Java script library for building user interface.

**ACADE MIND**

### What is React.js?

- React.js**
- A client-side JavaScript library
- All about building modern, reactive user interfaces for the web
- Declarative, component-focused approach

Udemy

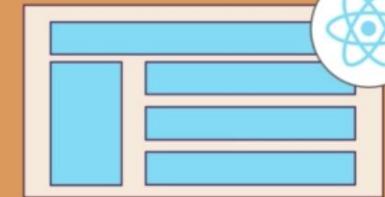
## Building Single-Page-Applications (SPAs)

React can be used to **control parts** of HTML pages or entire pages.



"Widget" approach on a multi-page-application.  
(Some) pages are still **rendered on** and served by a backend server.

React can also be used to **control** the **entire frontend** of a web application



"Single-Page-Application" (SPA) approach. Server **only sends one HTML page**, thereafter, React takes over and controls the UI.

Udemy

### Reactive Alternative

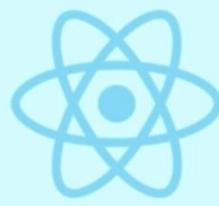
## React.js Alternatives

Angular



Complete component-based UI framework, packed with features. Uses TypeScript. Can be overkill for smaller projects.

React.js



Lean and focused component-based UI library. Certain features (e.g. routing) are added via community packages.

Vue.js



Complete component-based UI framework, includes most core features. A bit less popular than React & Angular.

Udemy

## Creating a React project

```
npx create-react-app my-app
```

The screenshot shows the VS Code interface with the terminal tab active. The terminal window displays the command `npx create-react-app 30-react-app` being run, followed by the output of the application creation process. The sidebar on the left shows a file tree with several React-related projects listed, and the bottom left shows navigation links for 'OUTLINE' and 'TIMELINE'.

```
PROBLEMS OUTPUT TERMINAL  
TERMINAL  
/Users/naveen.kumar1/.zshrc:68: no matches found: load?  
→ REACT git:(master) npx create-react-app 30-react-app  
Creating a new React app in /Users/naveen.kumar1/Workspace/WEB/REACT/30-react-app.  
Installing packages. This might take a couple of minutes.  
Installing react, react-dom, and react-scripts with cra-template...  
(( )) :: idealTree:30-re  
(( )) :: idealTree:@pmmmwh/react
```

## React Eco System



