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

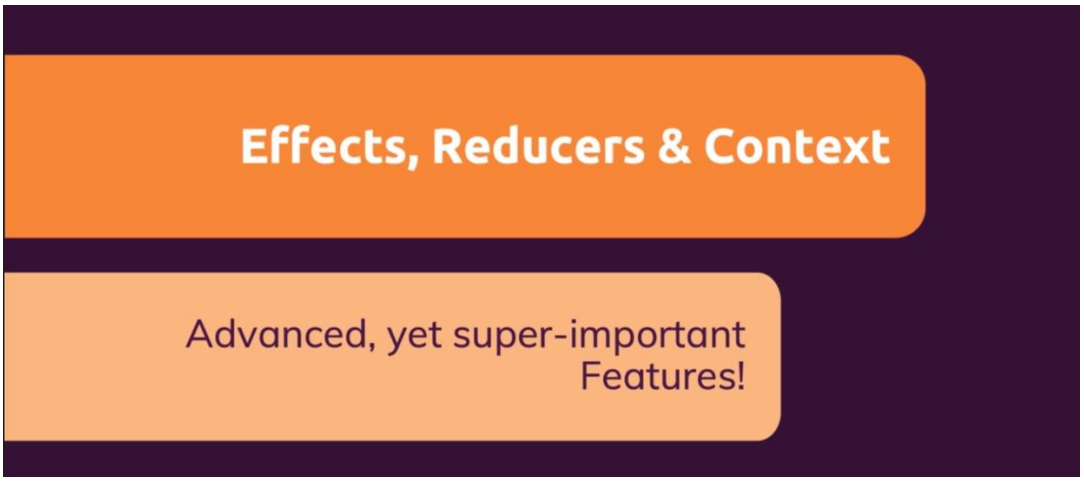
https://www.linkedin.com/in/naveen-kumar-m-5a6960119/

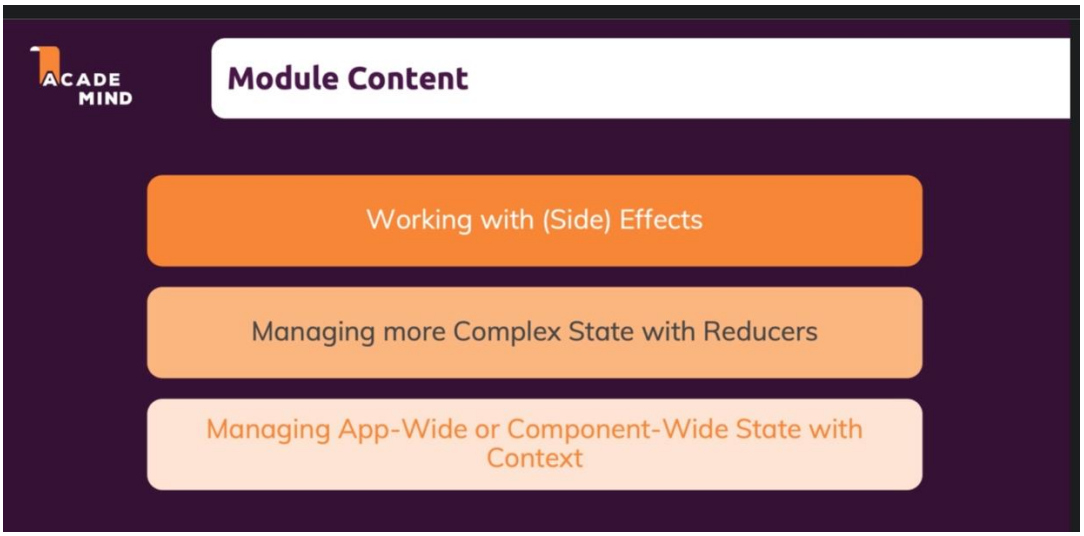
Source Code Link

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

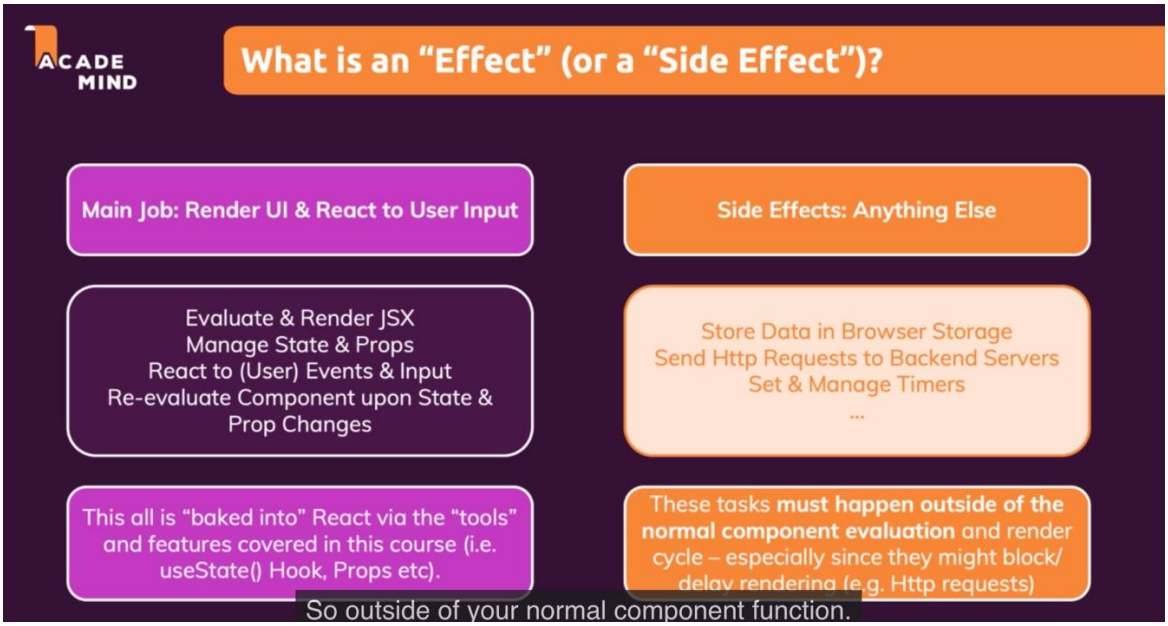
Table of Contents

Effects, Reducers & Context

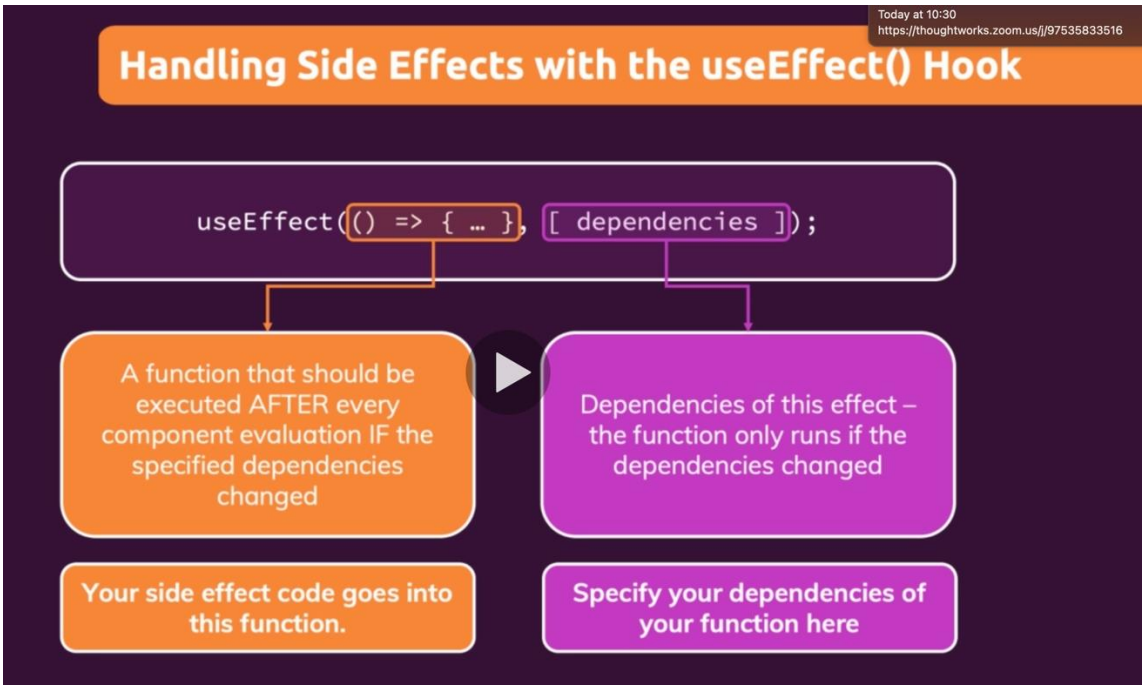


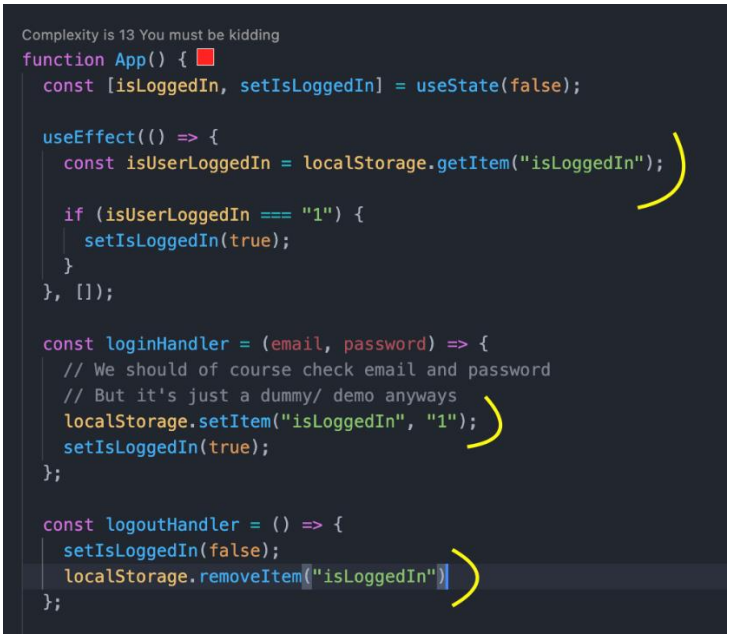


What is Side Effect ?



useEffect() Hook





localStorege is built-in storage in browser.

Use Effect Example

This useEffect() function run only once on starting the app, since we don’t specified 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!

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

let myTimer;

const MyComponent = (props) => {
  const [timerIsActive, setTimerIsActive] = useState(false);

  const { timerDuration } = props; // using destructuring to

  useEffect(() => {
    if (!timerIsActive) {
      setTimerIsActive(true);
      myTimer = setTimeout(() => {
        setTimerIsActive(false);
      }, timerDuration);
    }
  }, [timerIsActive, timerDuration]);
};
```

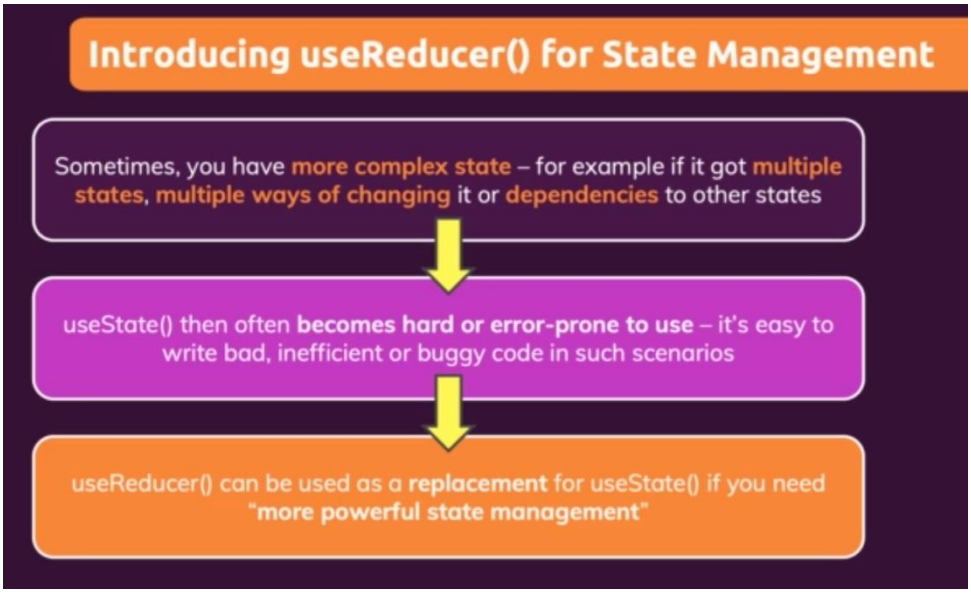
- `timerIsActive` is **added as a dependency** because it's component state that may change when the component changes (e.g. because the state was updated)
- `timerDuration` is **added as a dependency** because it's a prop value of that component - so it may change if a parent component changes that 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()

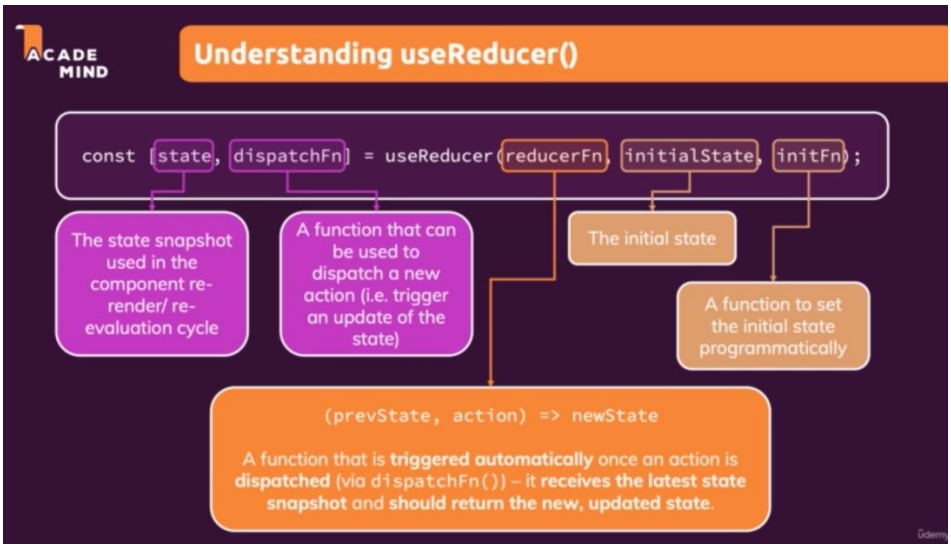
- setTimeout() – Based on the dependency, useEffect function will execute. Every time dependencies value changes, useEffect function trigger immediately after the component function render. So, every time key typed useEffect will get triggered. This will end make the multiple call to this function. To 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.



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.val, isValid: action.val.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**.

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

This works just fine as well!

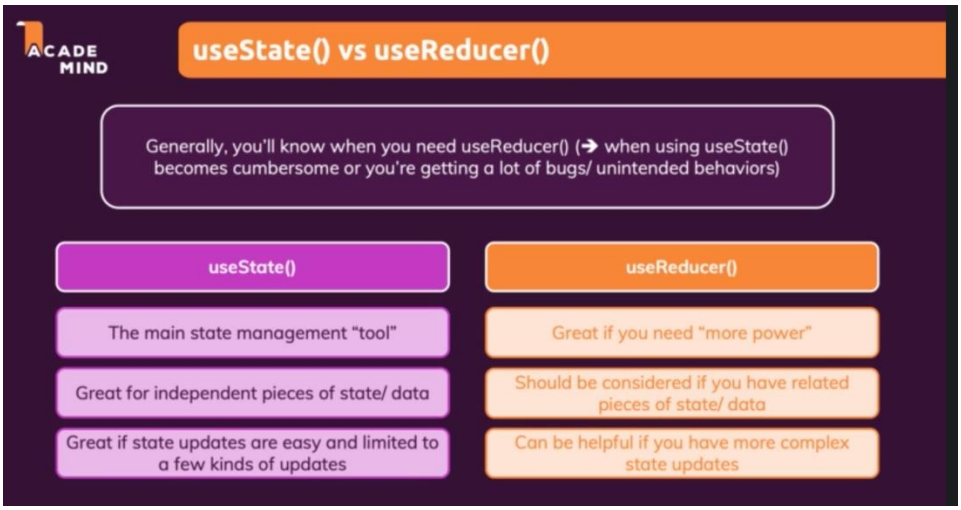
But you should **avoid** this code:

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

Why?

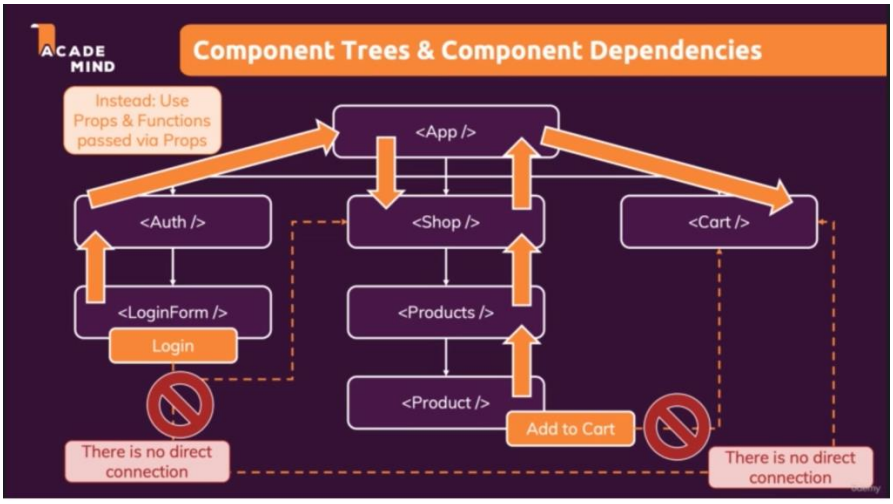
Because now the **effect function would re-run whenever ANY property** of `someObject` changes - not just the one property (`someProperty` in the above example) our effect might depend on.

useReducer() vs useState - State Management

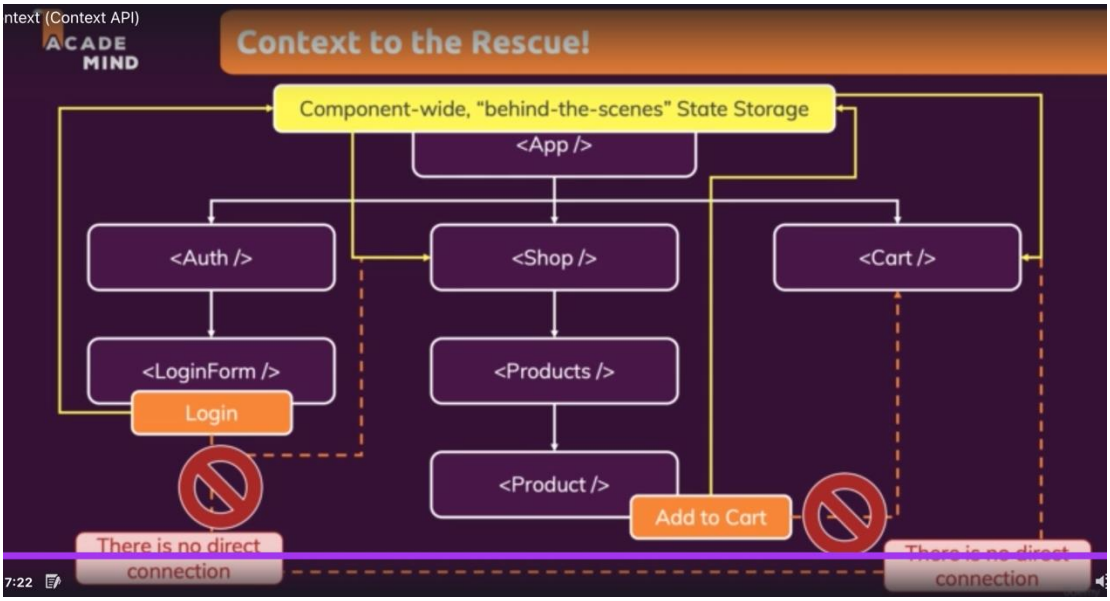


Context

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
              </li>
            )}
          </ul>
        </nav>)
      }}
    </AuthContext.Consumer>
```

Context consumer using hook

```
import React, {useContext} from 'react';
import AuthContext from '../store/auth-conext';

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</
          </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</bu
          </li>
        )}
      </ul>
    </nav>
```

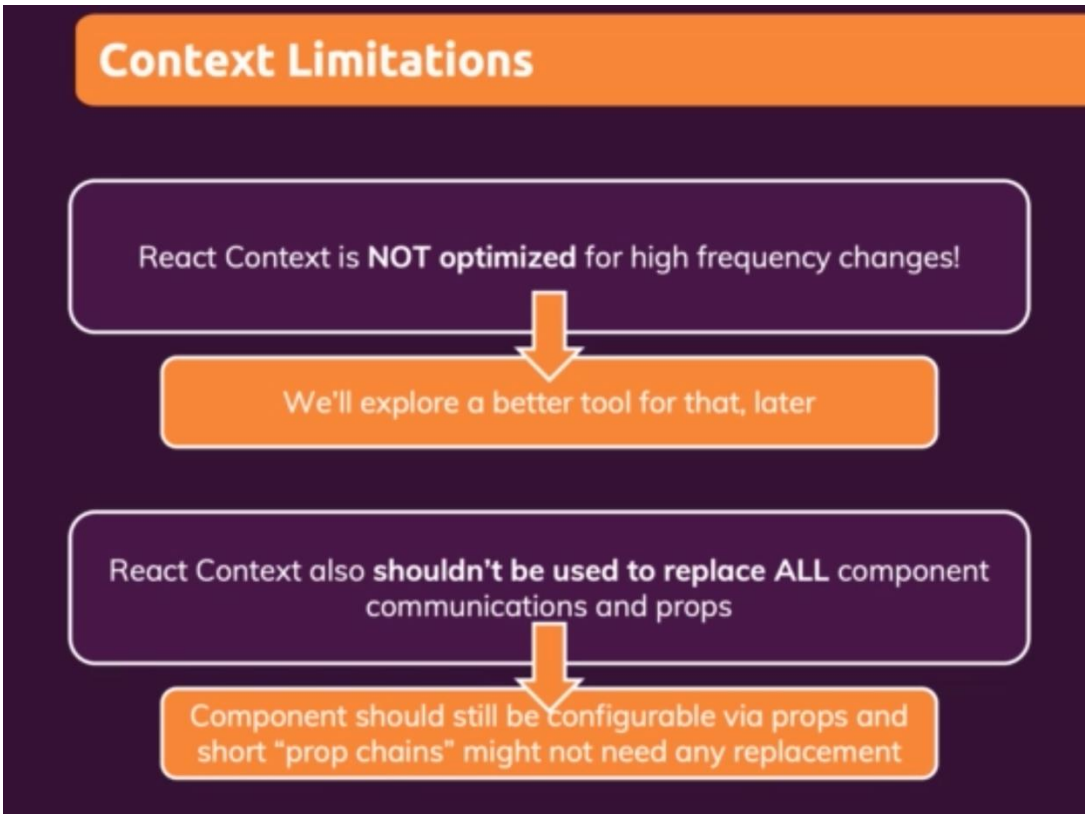
Using centralized state management using Context

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

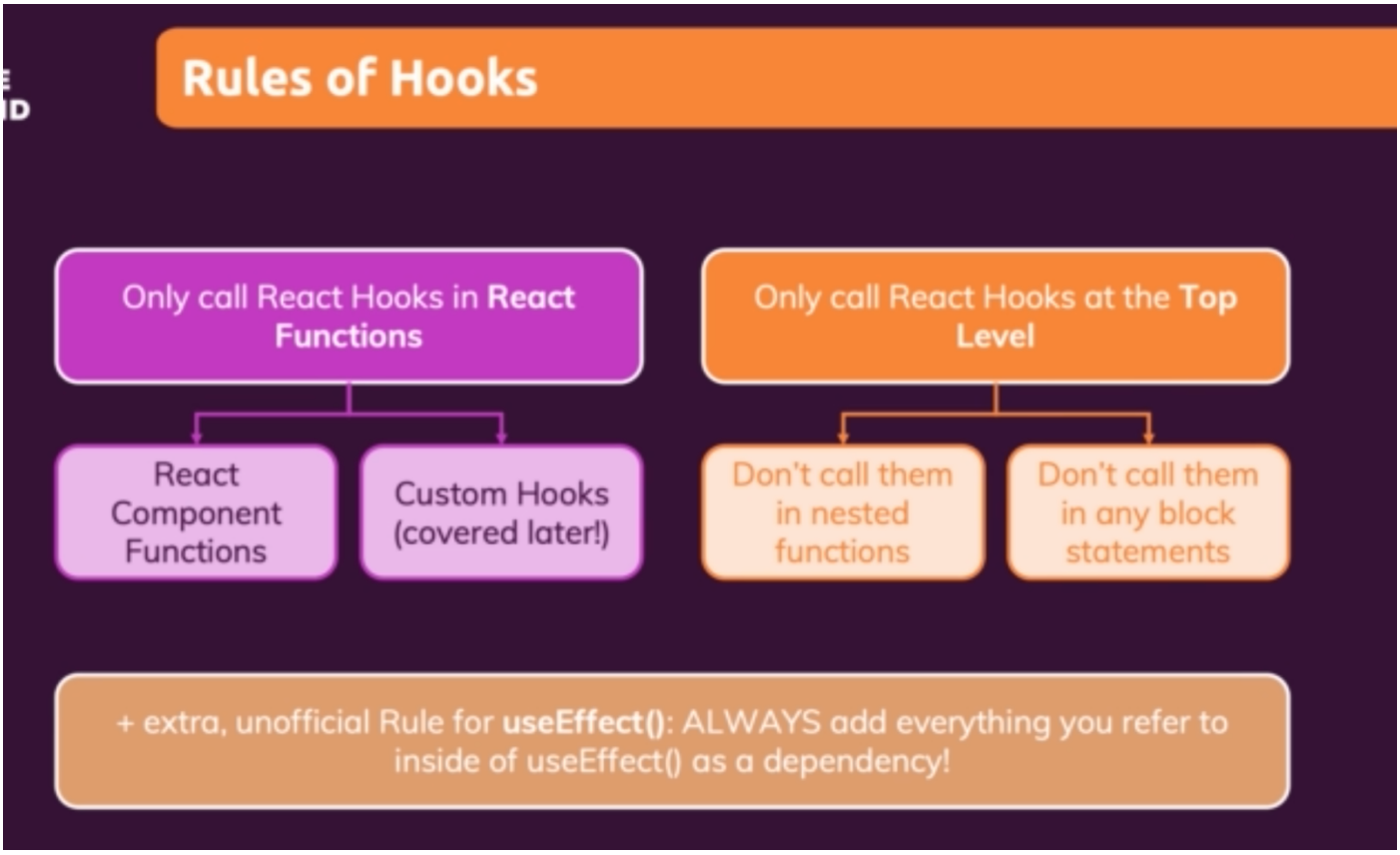
```
import React, { useState, useEffect } from "react";

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 Limitation



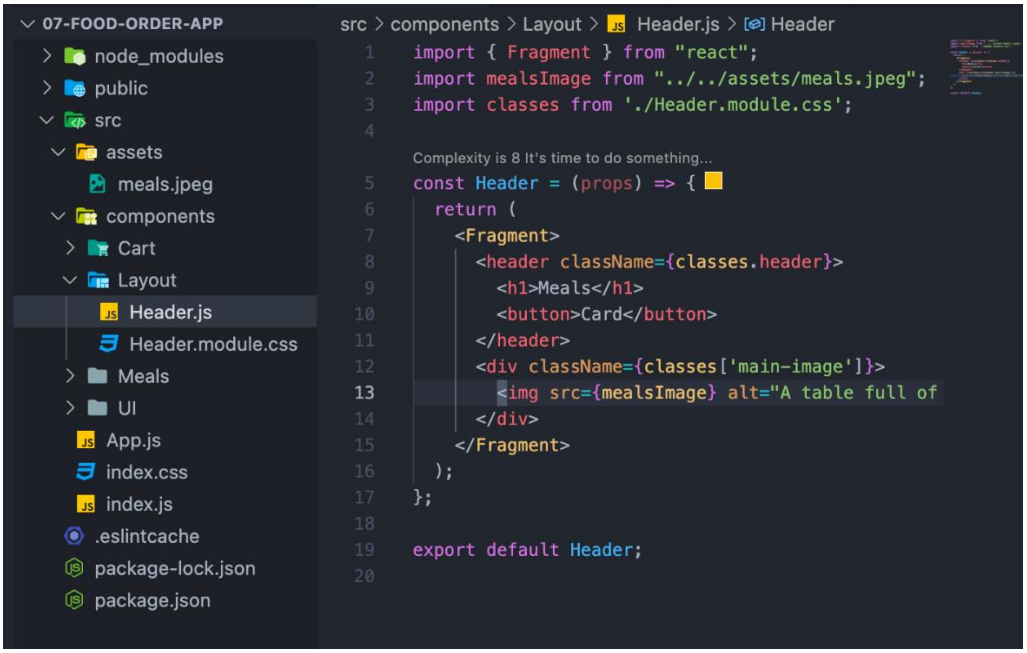
Rules of Hooks

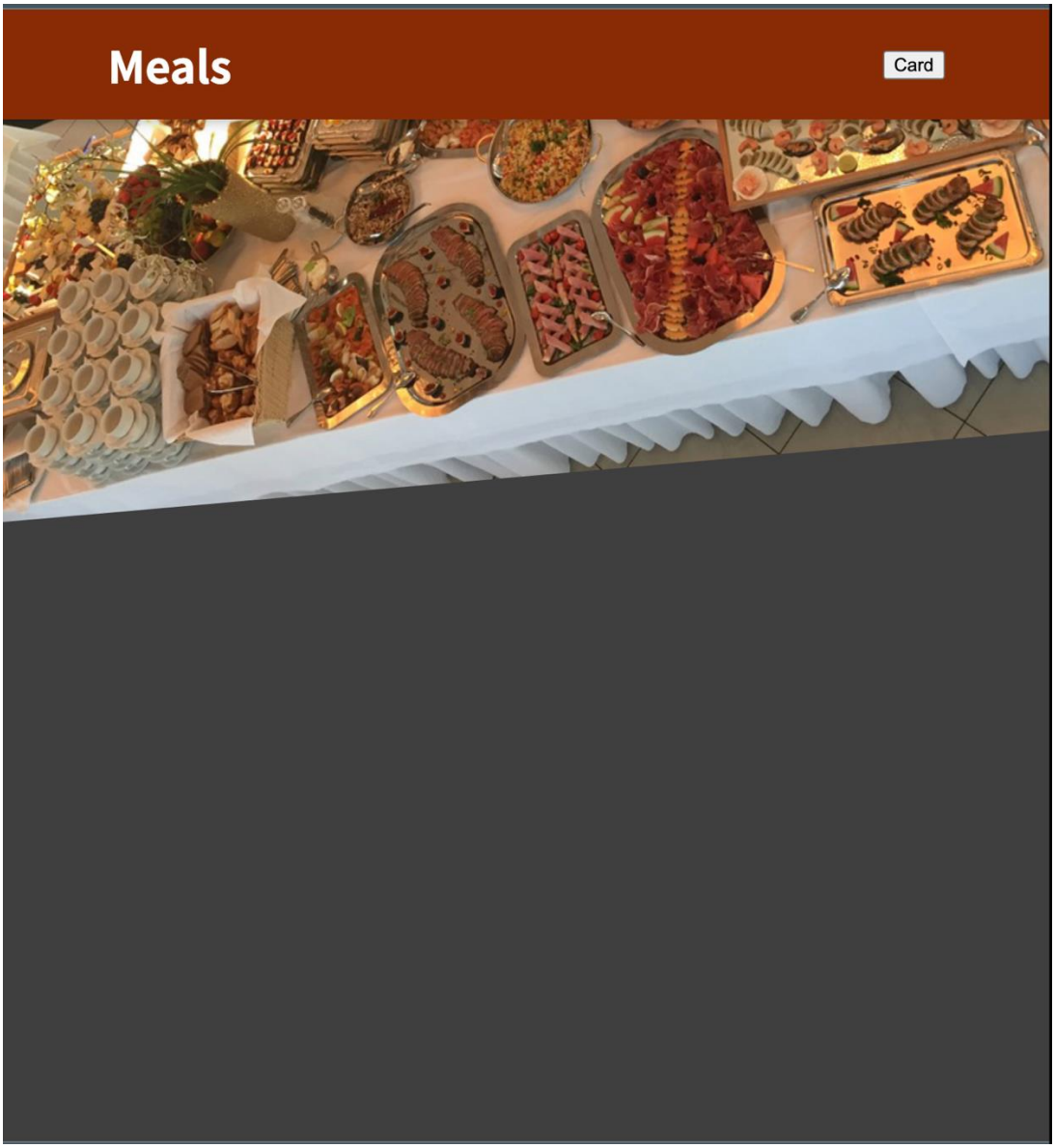


```
Complexity is 5 Everything is cool!
const submitHandler = (event) => {
  event.preventDefault();
  if (formIsValid) {
    authCtx.onLogin(email.value, password.value);
  } else if (!emailIsValid) {
    emailInputRef.current.focus();
  } else {
    passwordInputRef.current.focus();
  }

};

return (
  <Card className={classes.login}>
    <form onSubmit={submitHandler}>
      <Input
        ref={emailInputRef}
        id="email"
        label="E-Mail"
        type="email"
        isValid={emailIsValid}
        value={email.value}
        onChange={emailChangeHandler}
        onBlur={validateEmailHandler}
      />
      <Input
        ref={passwordInputRef}
        id="password"
        label="Password"
        type="password"
        isValid={passwordIsValid}
        value={password.value}
        onChange={passwordChangeHandler}
        onBlur={validatePasswordHandler}
      />
```

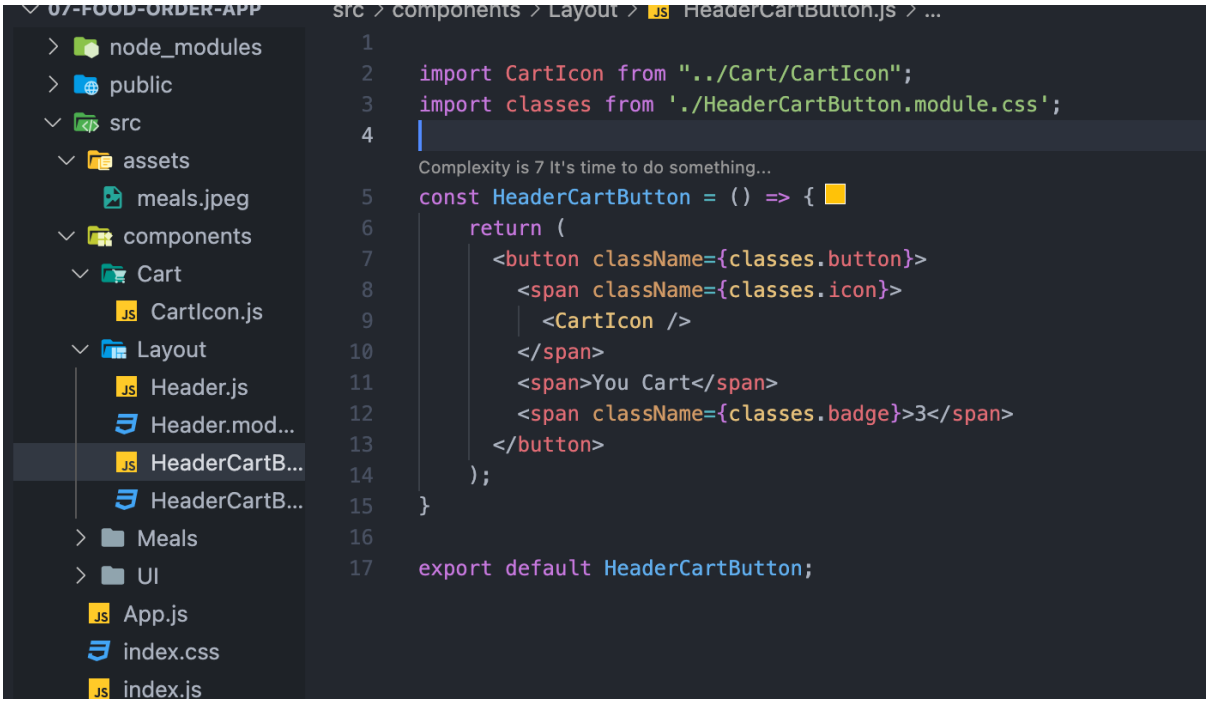
```
Complexity is 9 It's time to do something...
const Input = React.forwardRef((props, ref) => {

  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;
```

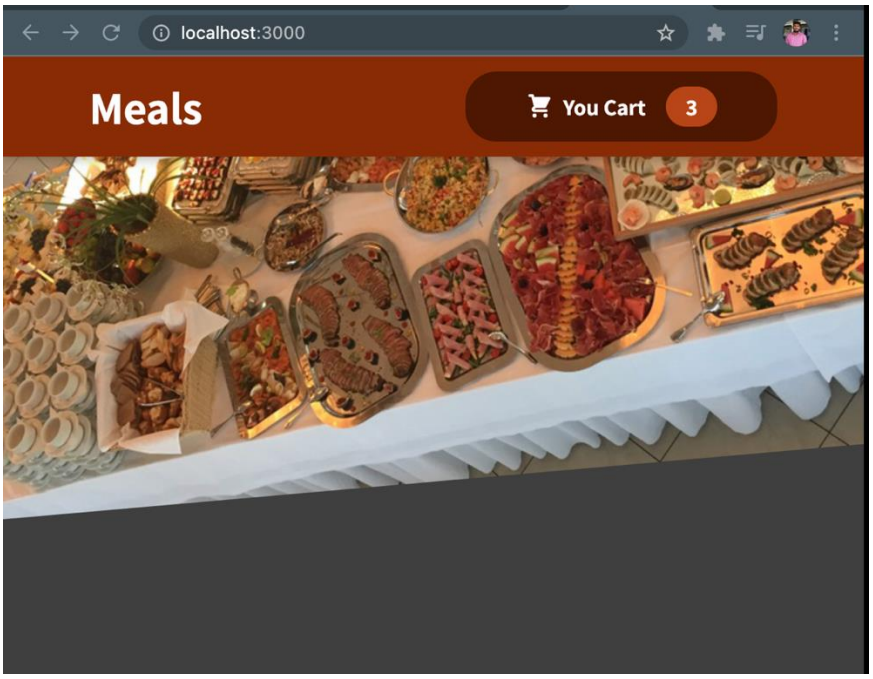
Practice Food Order App



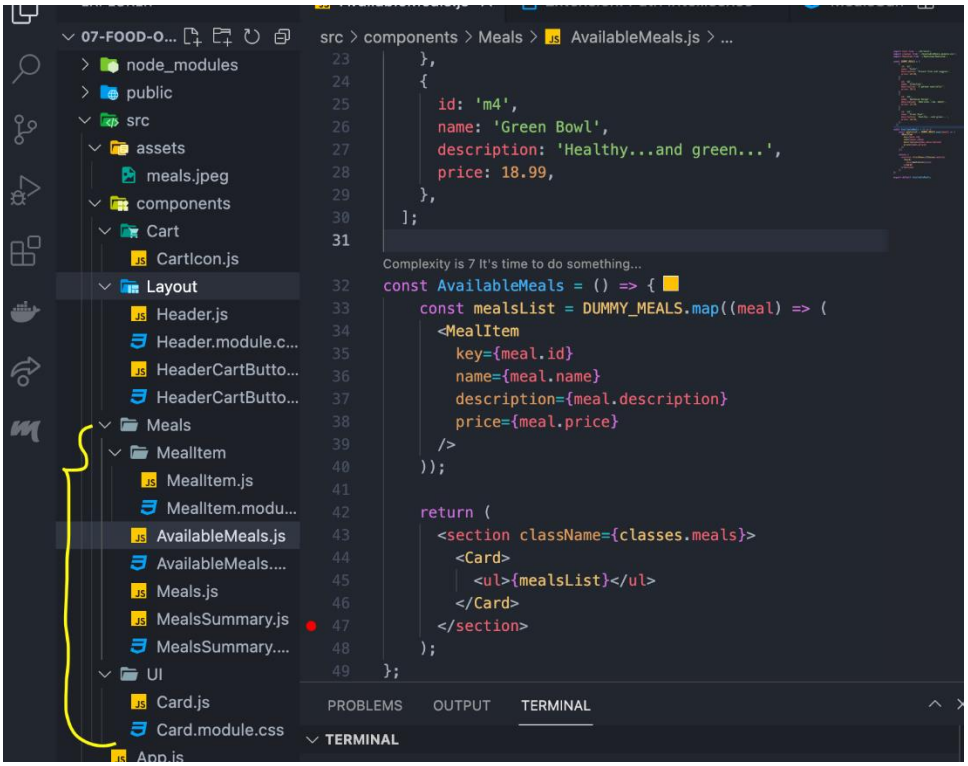


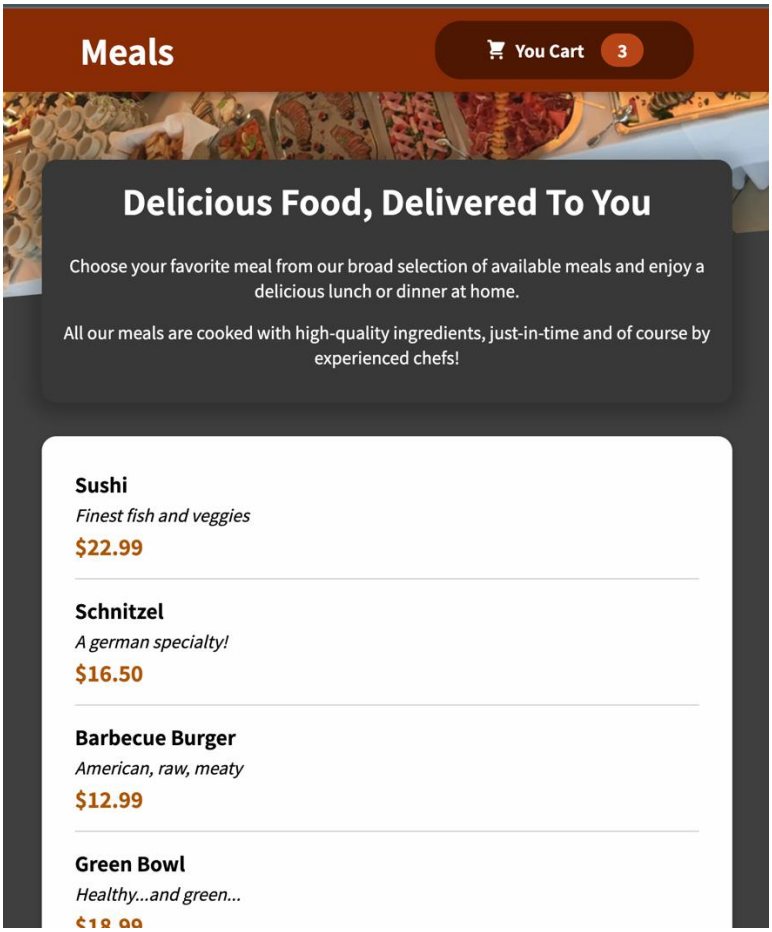
Adding Cart Button



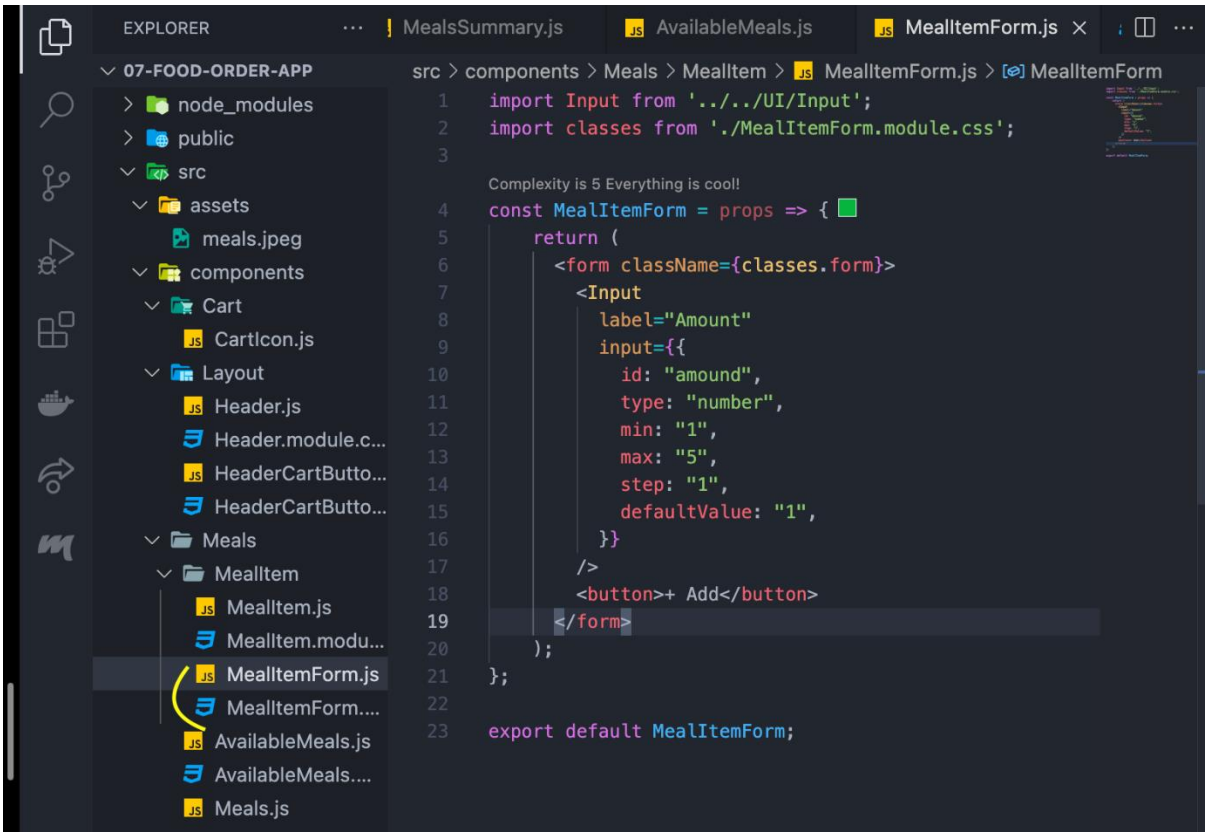


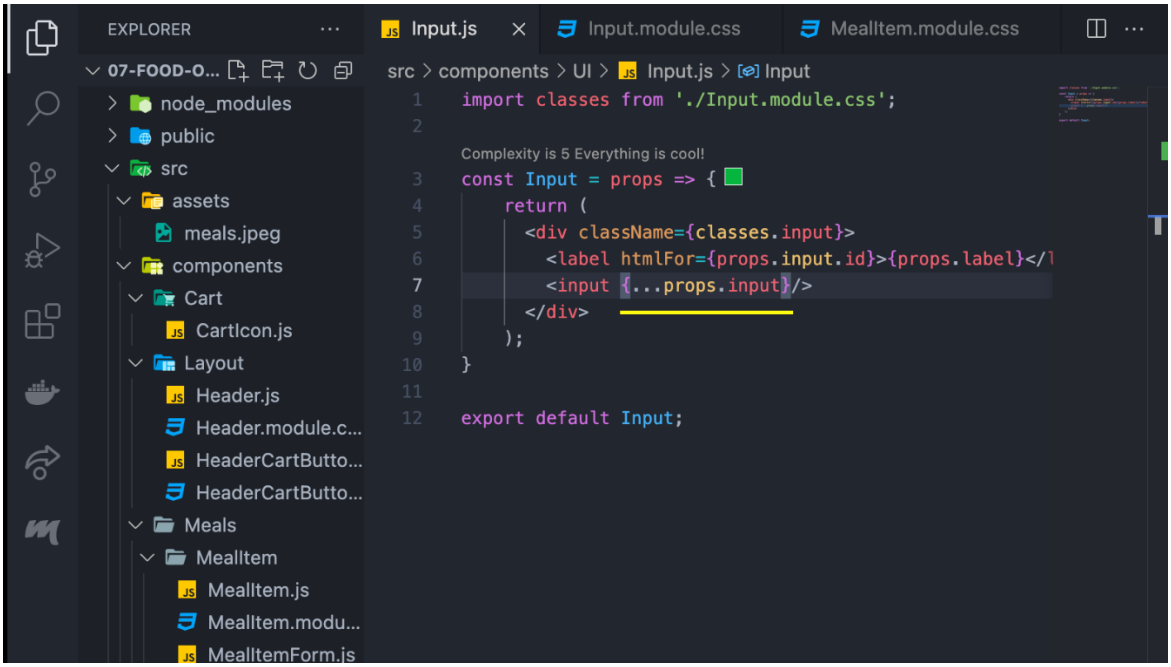
Adding a meals component





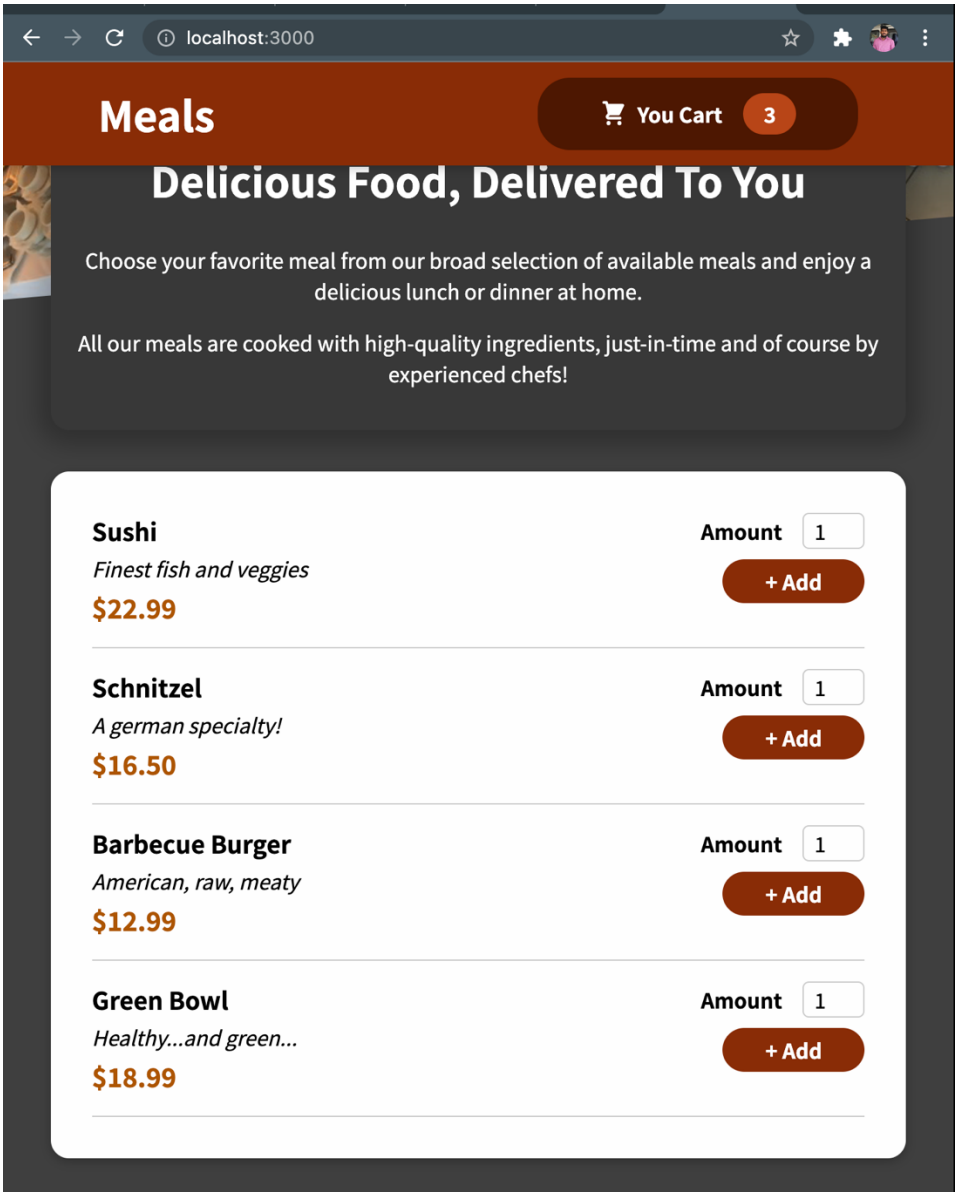
Adding a Form



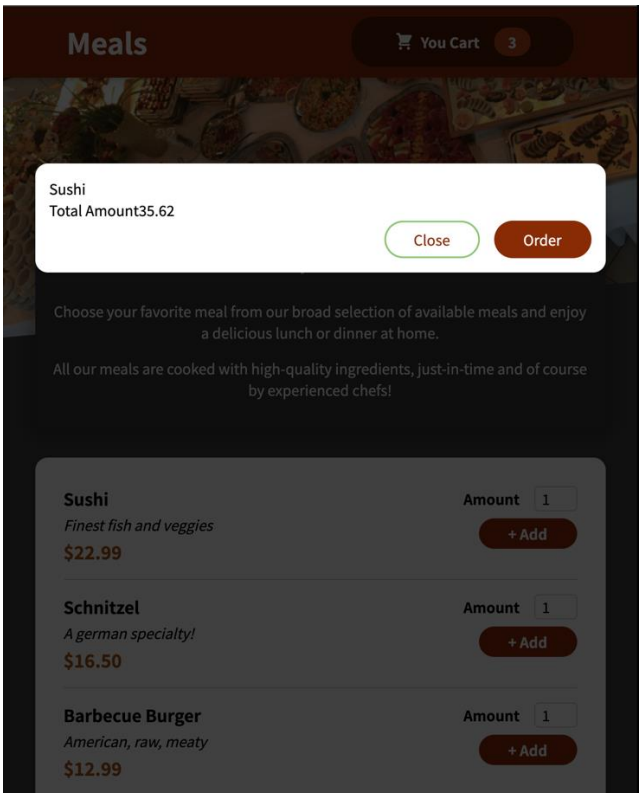

EXPLORER
Input.js
Input.module.css
MealItem.module.css
07-FOOD-O...
node_modules
public
src
assets
meals.jpeg
components
Cart
CartIcon.js
Layout
Header.js
Header.module.c...
HeaderCartButto...
HeaderCartButto...
Meals
MealItem
MealItem.js
MealItem.modu...
MealItemForm.js
src > components > UI > Input.js > Input
import classes from './Input.module.css';
Complexity is 5 Everything is cool!
const Input = props => {
return (

<label htmlFor={props.input.id}>{props.label}</
<input {...props.input}/>

);
}
export default Input;


localhost:3000
Meals
You Cart 3
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
Amount 1
+ Add
Schnitzel
A german specialty!
$16.50
Amount 1
+ Add
Barbecue Burger
American, raw, meaty
$12.99
Amount 1
+ Add
Green Bowl
Healthy...and green...
$18.99
Amount 1
+ Add

Adding a Model via React Portal




```jsx
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

```jsx
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}/>
      <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

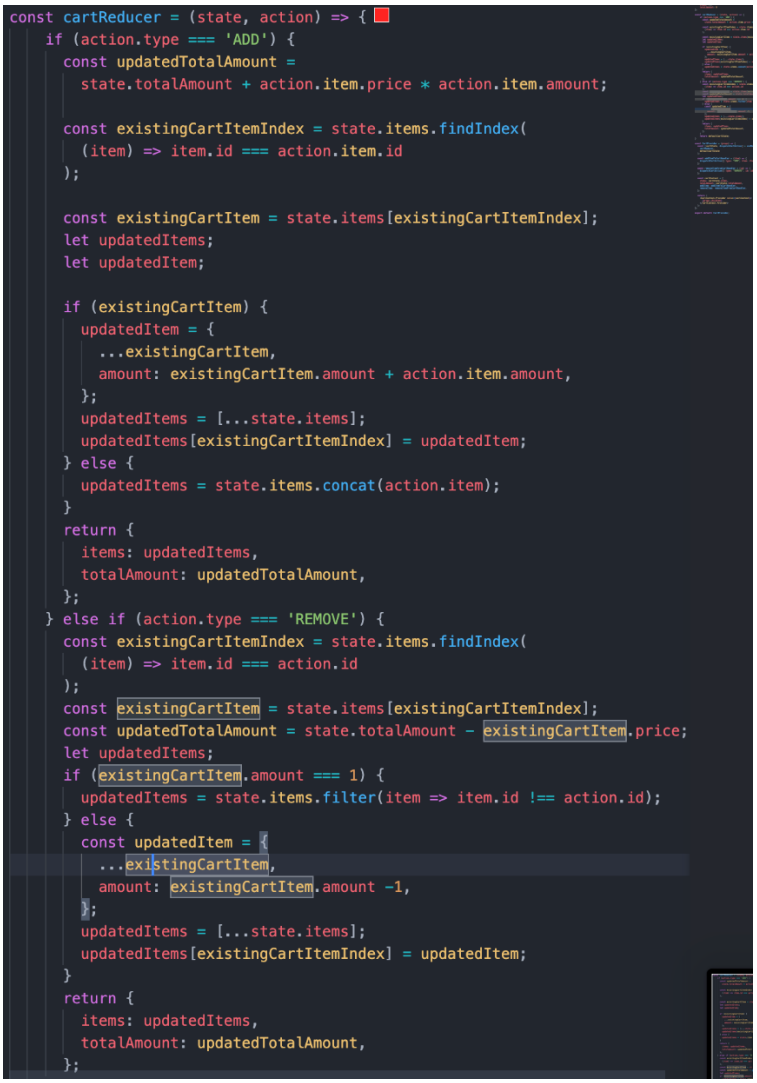
```
import CartContext from "./cart-context";
import { useReducer } from "react";

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.label}</
        <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(tr
  const amountInputRef = useRef();

  Complexity is 5 Everything is cool!
  const sumbitHandler = 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={sumbitHand
      <Input
        ref={amountInputRef}
        label="Amount"
        input={{
          id: "amound_" + props.id,
          type: "number",
          min: "1",
          max: "5",
          step: "1",
          defaultValue: "1",
        }}
      />
      <button>+ Add</button>
      {!amountIsValid && <p>Please enter a valid amount
    </form>
  );
```

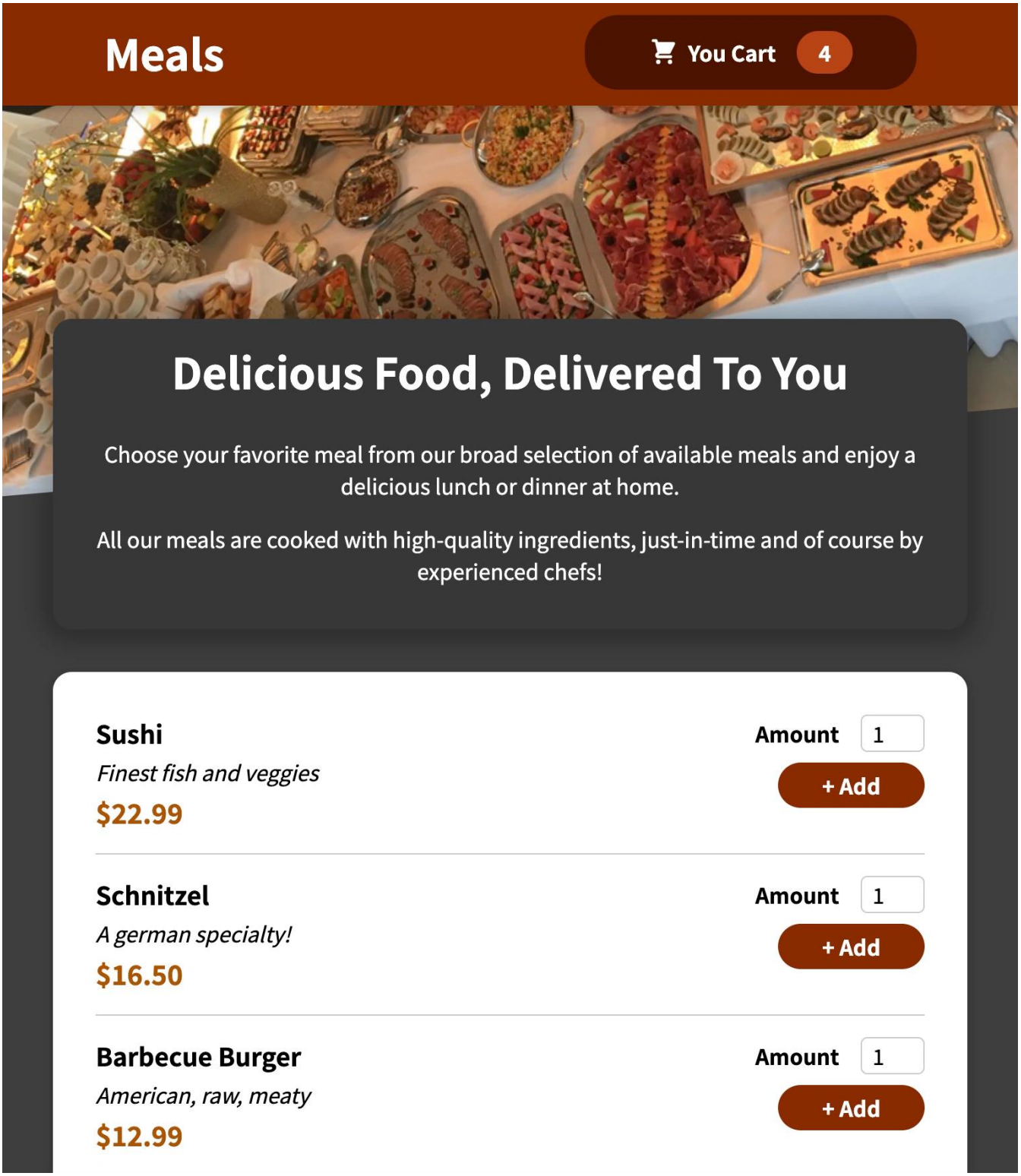
Reducer logic for adding and removing items from cart



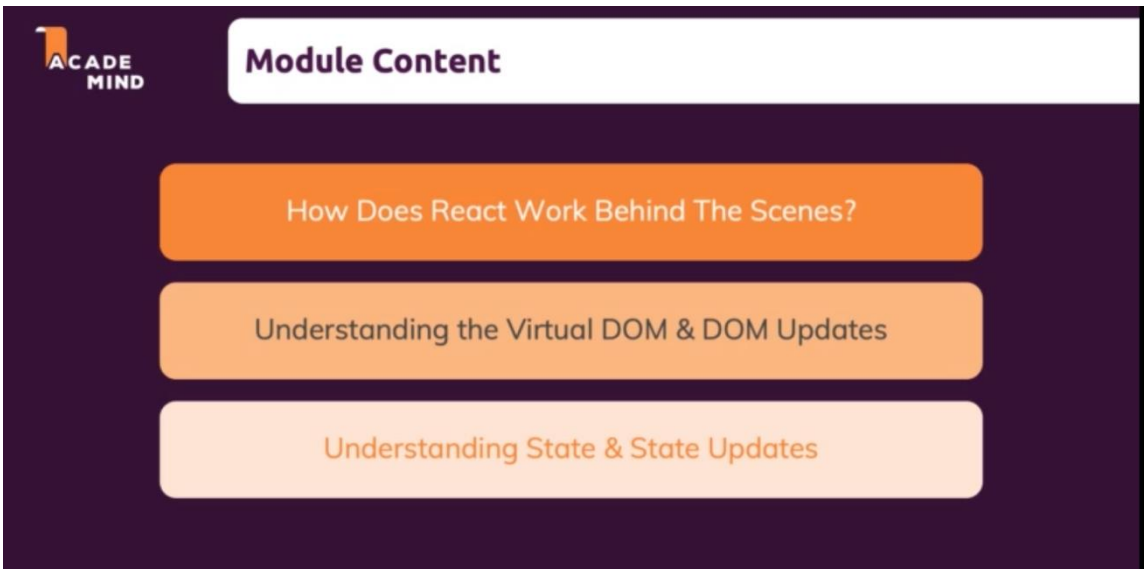
```
const cartReducer = (state, action) => {
  if (action.type === 'ADD') {
    const updatedTotalAmount =
      state.totalAmount + action.item.price * action.item.amount;

    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

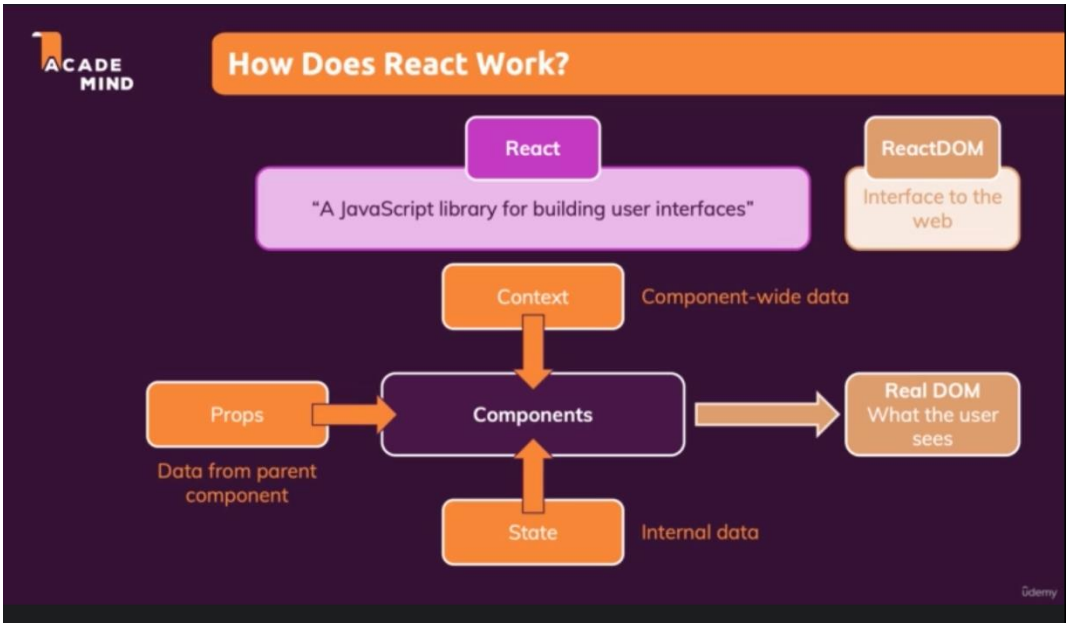
```
const btnClasses = `${classes.button} ${btnIsHighlighted ? classes.bump:''

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}>
```

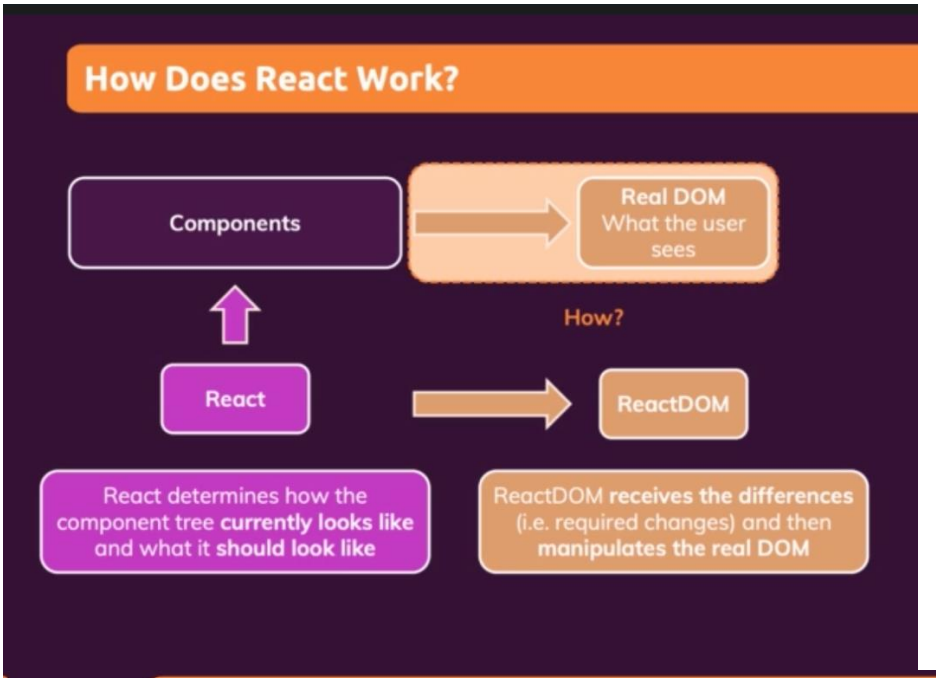

Meals
You Cart 4
Delicious Food, Delivered To You
Choose your favorite meal from our broad selection of available meals and enjoy a delicious lunch or dinner at home.
All our meals are cooked with high-quality ingredients, just-in-time and of course by experienced chefs!
Sushi
Finest fish and veggies
$22.99
Amount 1
+ Add
Schnitzel
A german specialty!
$16.50
Amount 1
+ Add
Barbecue Burger
American, raw, meaty
$12.99
Amount 1
+ Add

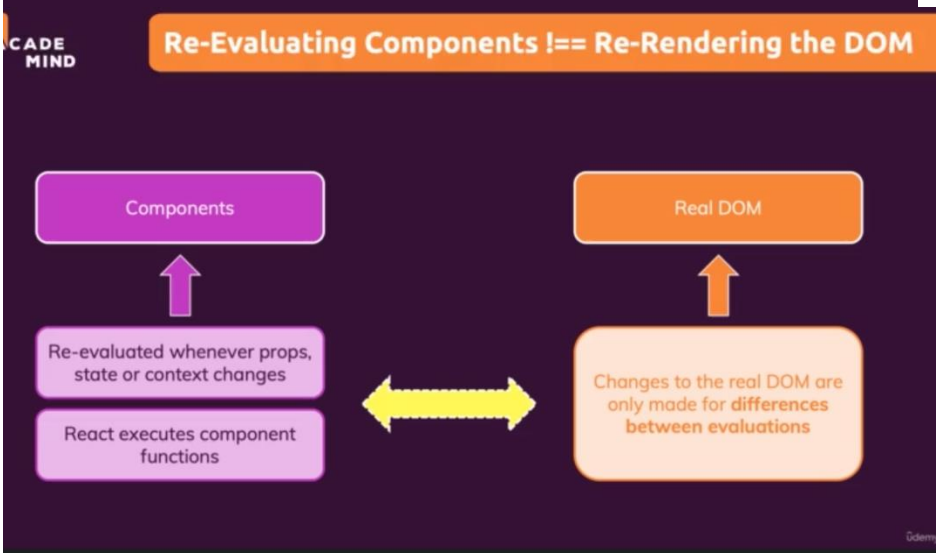
How React Works?

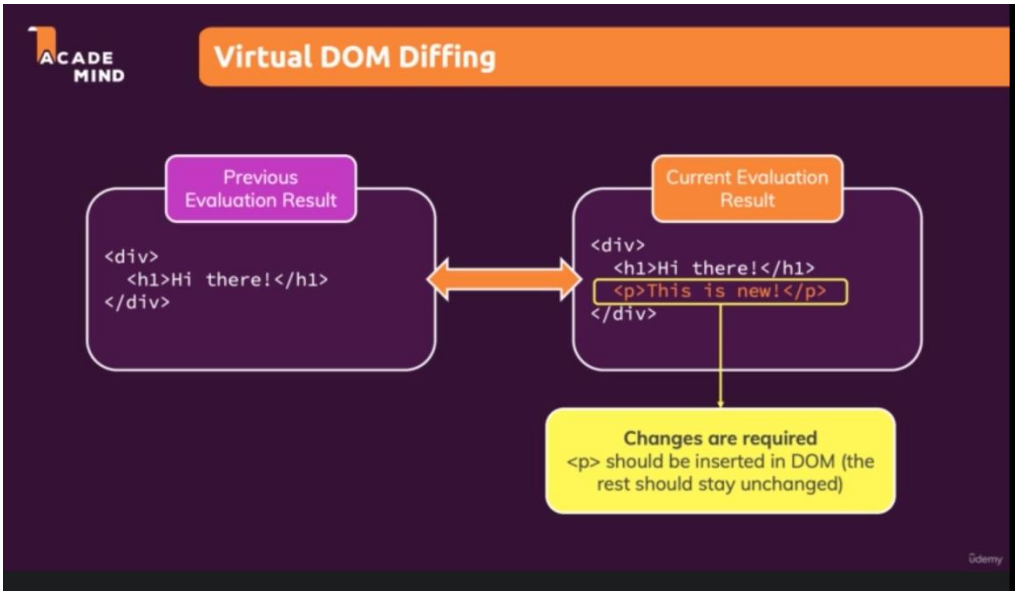


Understanding




How Does React Work?
Components
Real DOM
What the user sees
How?
React
ReactDOM
React determines how the component tree currently looks like and what it should look like
ReactDOM receives the differences (i.e. required changes) and then manipulates the real DOM


ACADEMIND
Re-Evaluating Components !== Re-Rendering the DOM
Components
Real DOM
Re-evaluated whenever props, state or context changes
React executes component functions
Changes to the real DOM are only made for differences between evaluations


ACADEMIND
Virtual DOM Diffing
Previous Evaluation Result

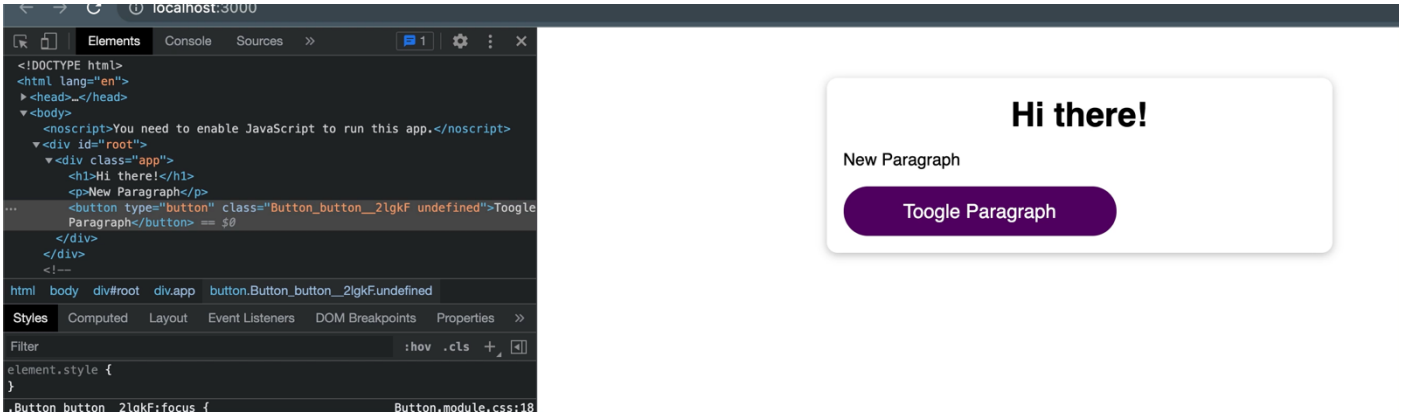
<h1>Hi there!</h1>

Current Evaluation Result

<h1>Hi there!</h1>
This is new!

Changes are required
 should be inserted in DOM (the rest should stay unchanged)

Demo - Inserting the element without refreshing the UI



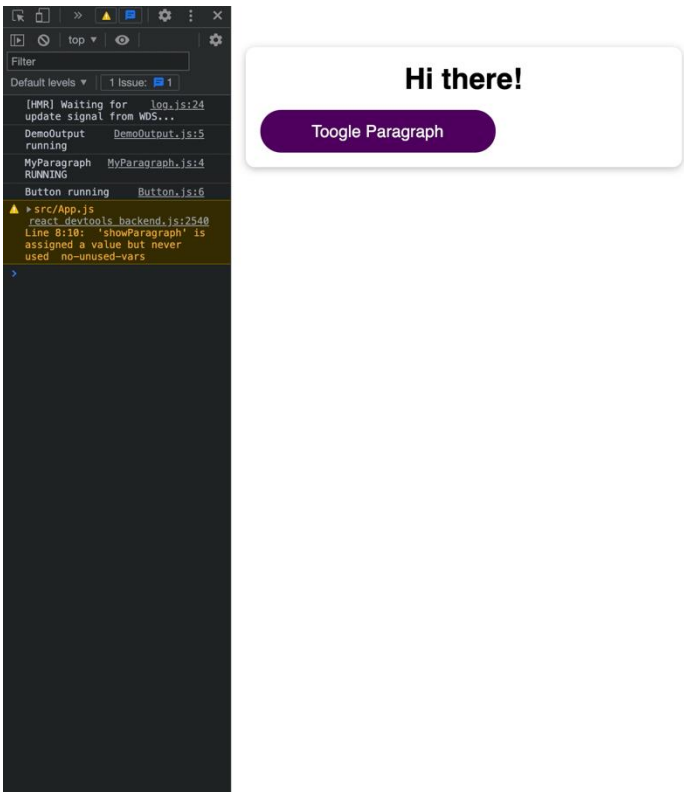

```
Complexity is 9 It's time to do something...
function App() {

  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;
```

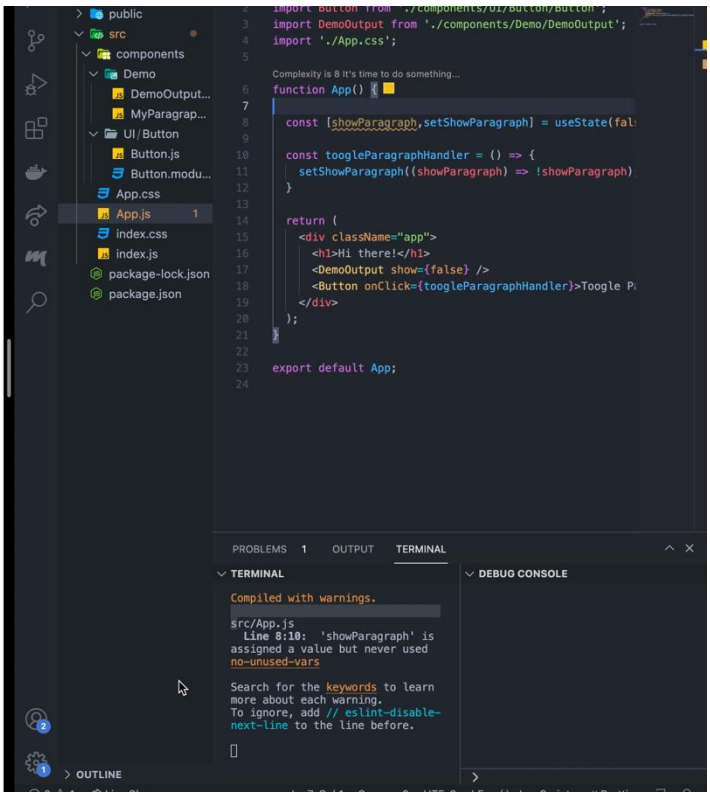
Child component re evaluation

Child component re-executed when parent component state changes.

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

  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.





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()

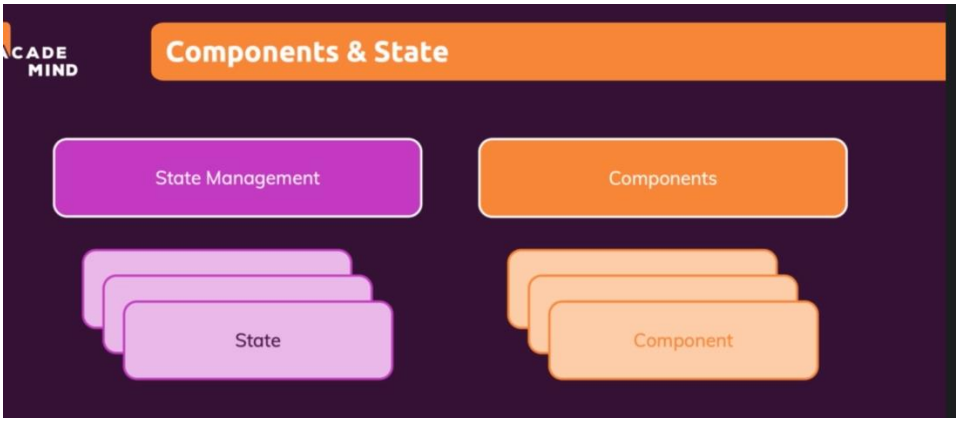
- useCallBack() hook help us to save a function to help preventing recreation of it when component function re-evaluated.
- Saves the object in memory and reuse it when component function re-evaluated.
- First argument is function that we want to store in memory and second argument is dependency array.

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

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

  const toogleParagraphHandler = useCallback(() => {
    setShowParagraph((showParagraph) => !showParagraph);
  },[]);

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

export default App;
```

Using callback with dependencies changes.

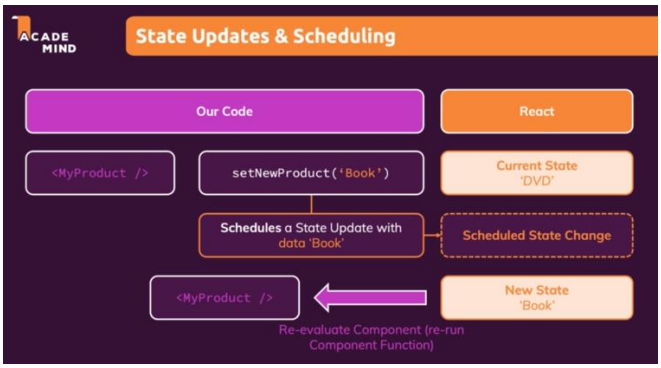
```
import React, { useState, useCallback } from 'react';
import Button from './components/UI/Button/Button';
import DemoOutput from './components/Demo/DemoOutput';
import './App.css';

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 toogleParagraphHandler = useCallback(() => {
    if (allowToggle){
      setShowParagraph((showParagraph) => !showParagraph);
    }
  },[allowToggle]);

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

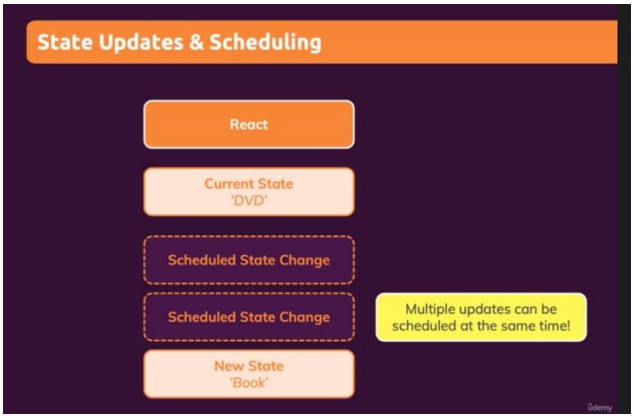
Component and state

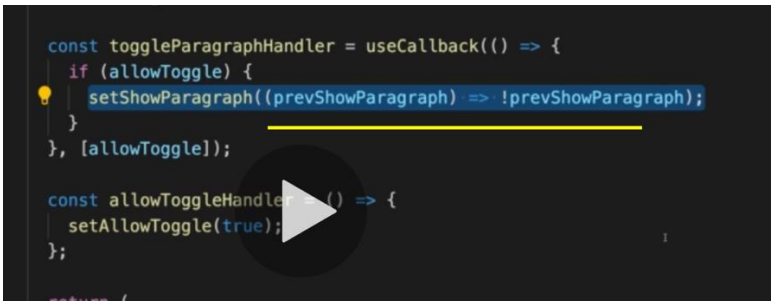


State is managed by react and connect with component.

State update and Scheduling



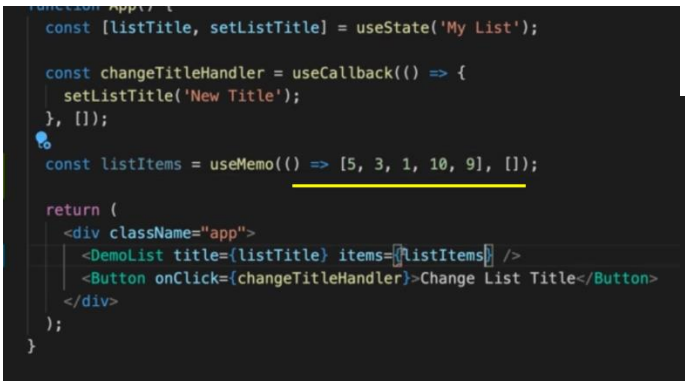




Due to the multiple state change, using the function in the state function helps. It always takes the previous state to update the state. Which means it always look into the latest state before updating the state.

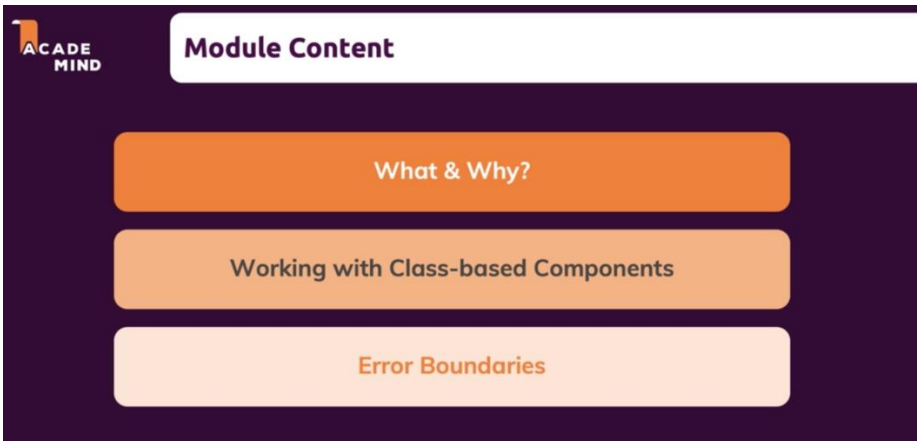
State patching happens when multiple states are updated together.

useMemo() hook

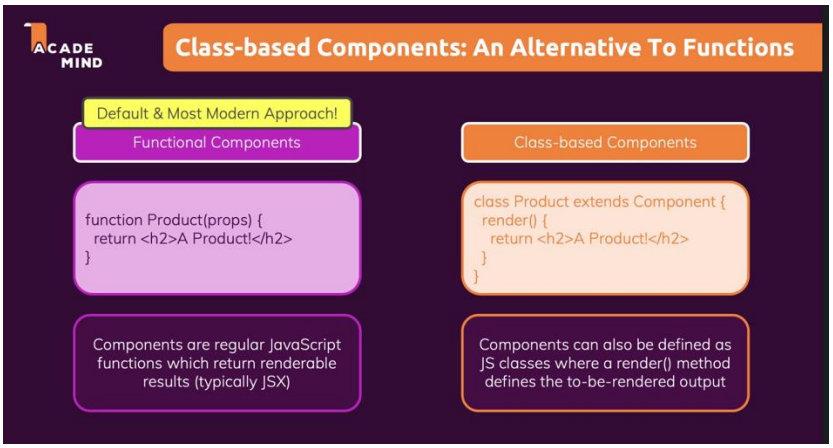


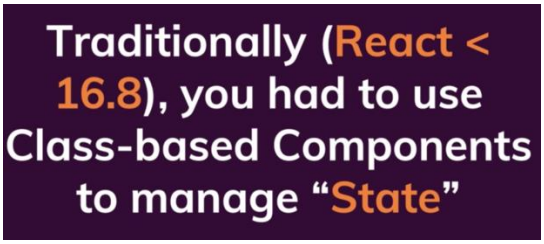
- useMemo() hook used to prevent the recreation of the object or function when component function reevaluated.

Class based Components



Class based components alternative to Functions





Functional component and class-based component can work together.

First class-based component

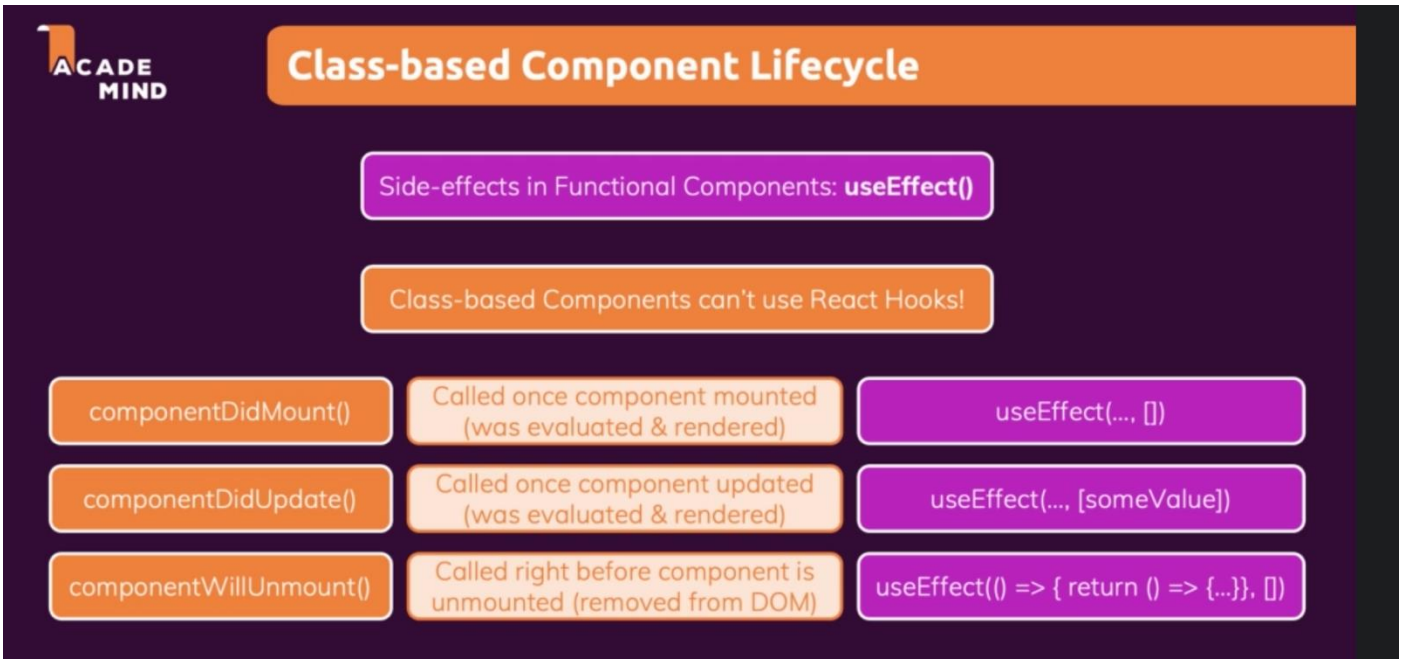
```
import classes from './User.module.css';
import {Component} from 'react';

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>
    );
  }
}
```

Component Lifecycle – using useEffect() functionality in Class based component

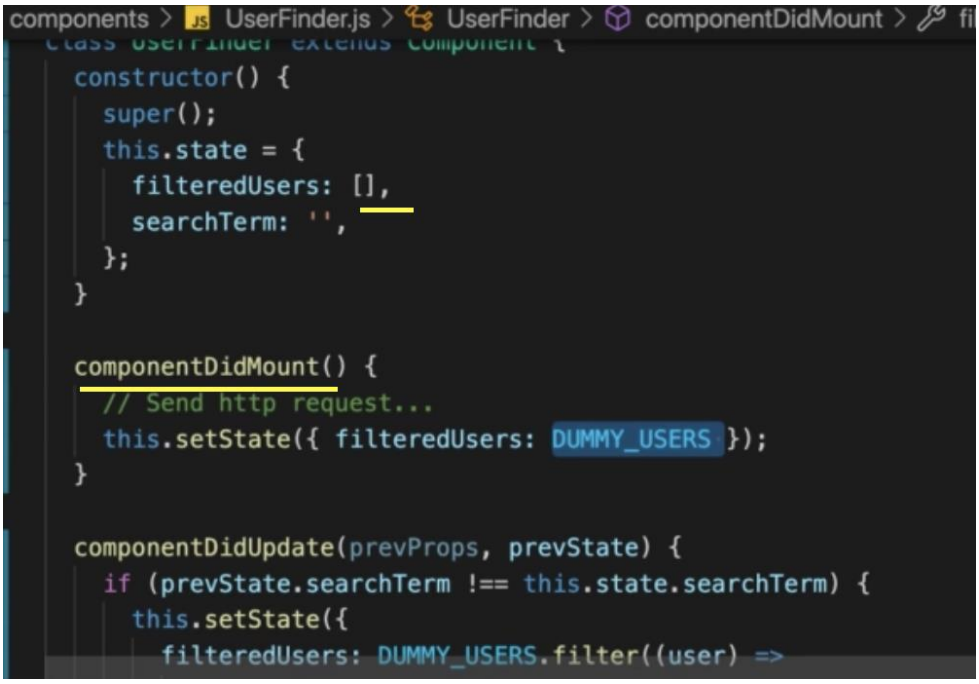



```
class UserFinder extends Component {
  constructor() {
    this.state = {
      filteredUsers: DUMMY_USERS,
      searchTerm: '',
    };
  }

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

componentDidUpdate()

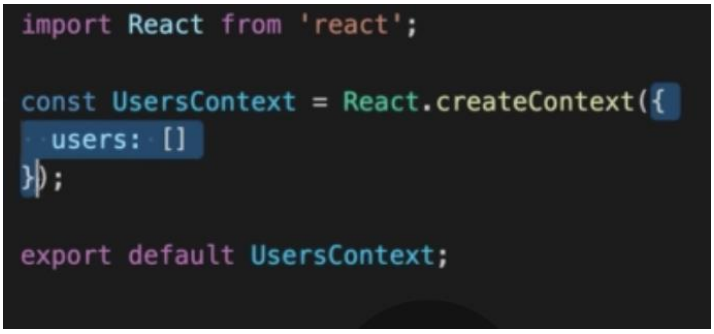
```
componentDidUpdate(prevProps, prevState) {
  if (prevState.searchTerm !== this.state.searchTerm) {
    this.setState({
      filteredUsers: DUMMY_USERS.filter((user) =>
        user.name.includes(this.state.searchTerm)
      ),
    });
  }
}
```

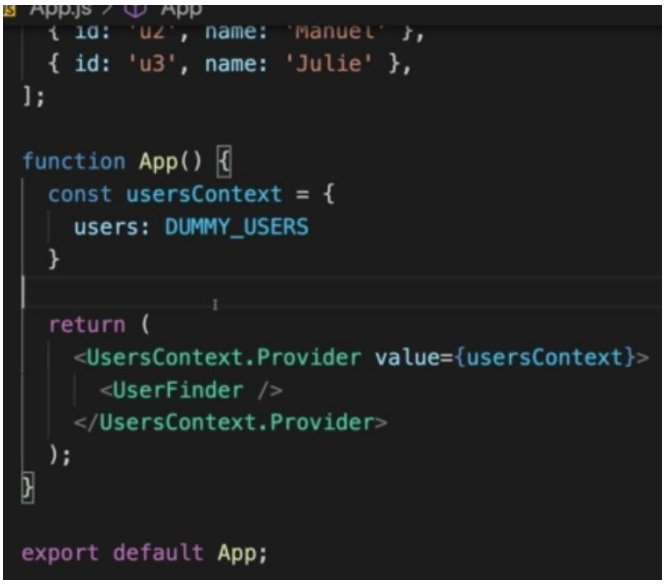
componentDidMount()



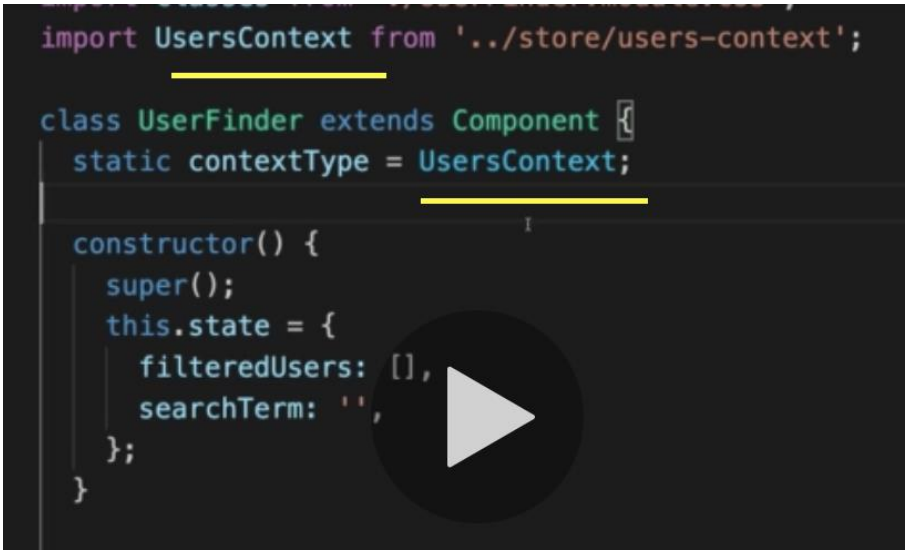
componentDidMount() function will be executed when component initialized.

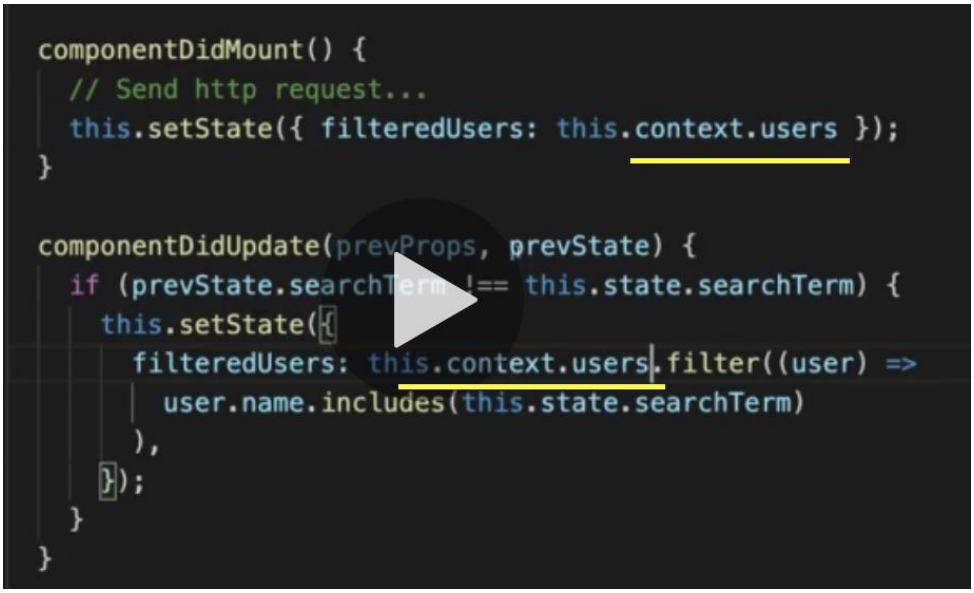
Using context in class-based component





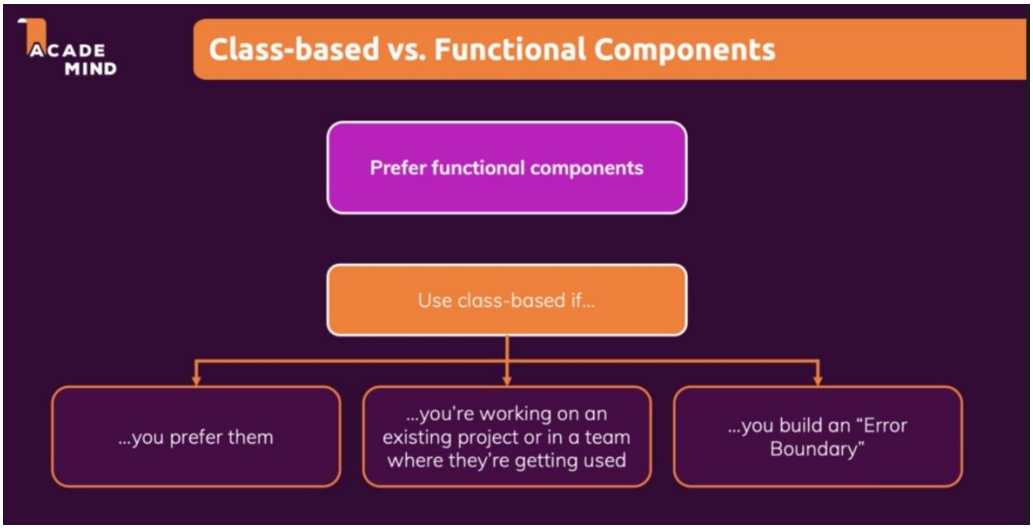
Only one context per class.





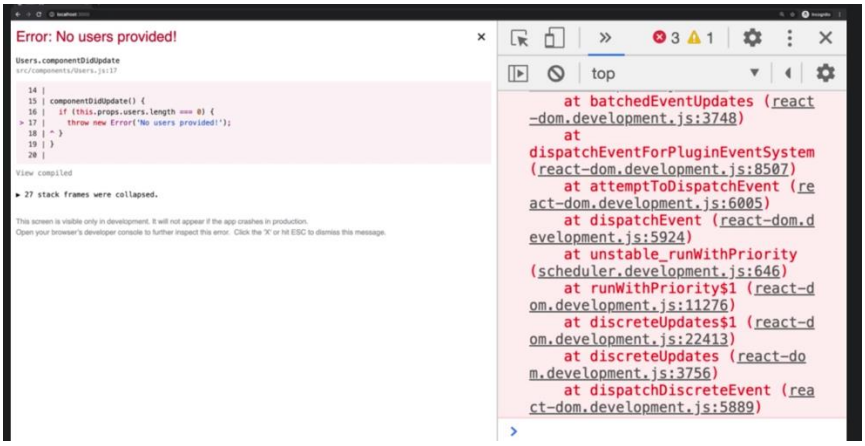
Functional components are leaner and flexible when compared to the class based component.

Class Based Component vs Functional component



Error Boundary

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



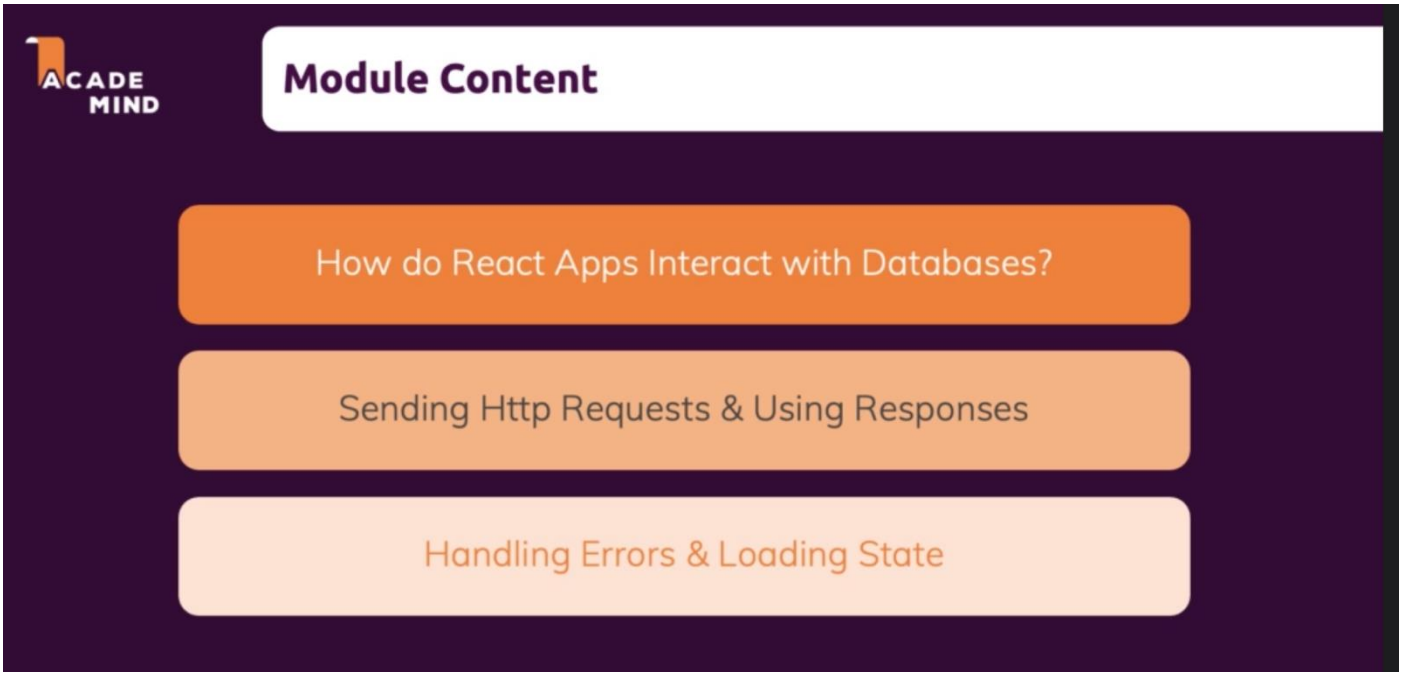
Having error in one component and handling it in another component.

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

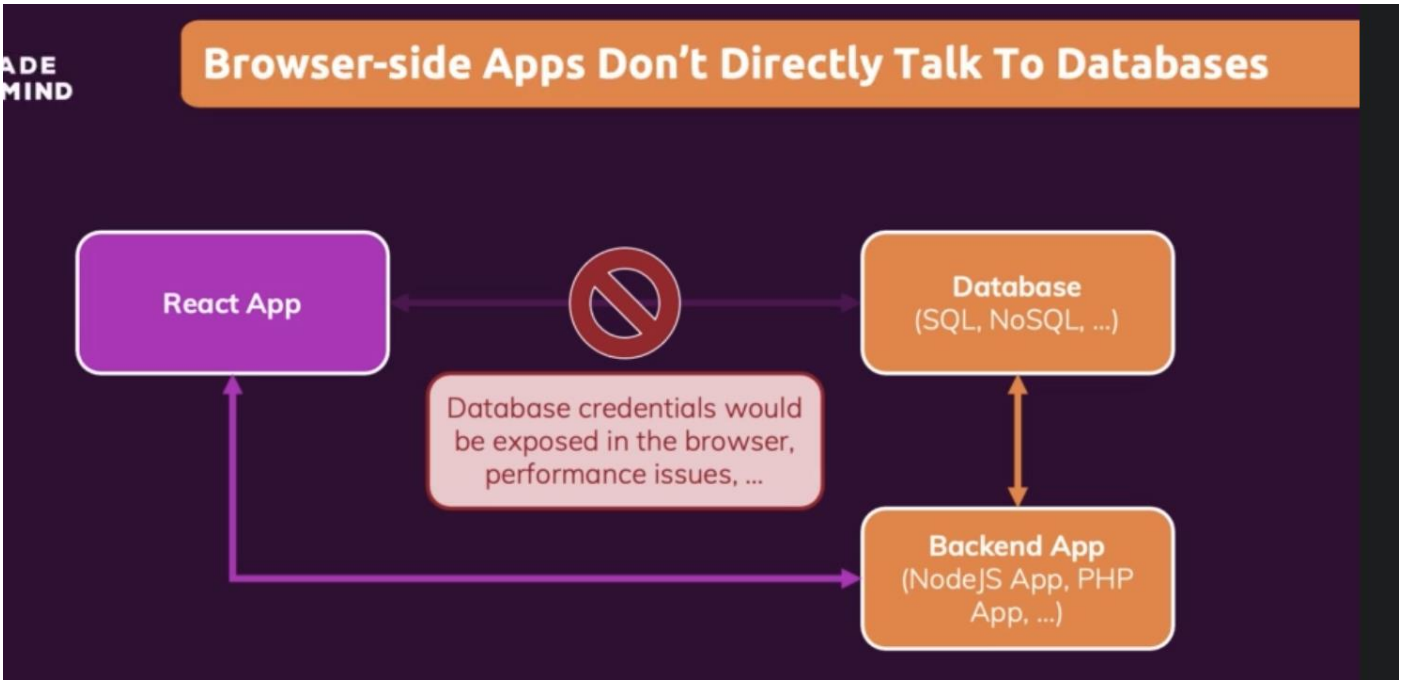
```
import { Component } from 'react';

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



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();
  Complexity is 3 Everything is cool!
  }).then(data => {
    const transformedMovies = data.results.map(movieData => {
      return {
        id : movieData.episod_id,
        title: movieData.title,
        openingText: movieData.opening_crawl,
        releaseDate: movieData.release_date
      }
    })
    setMovies(transformedMovies)
  });
}
```

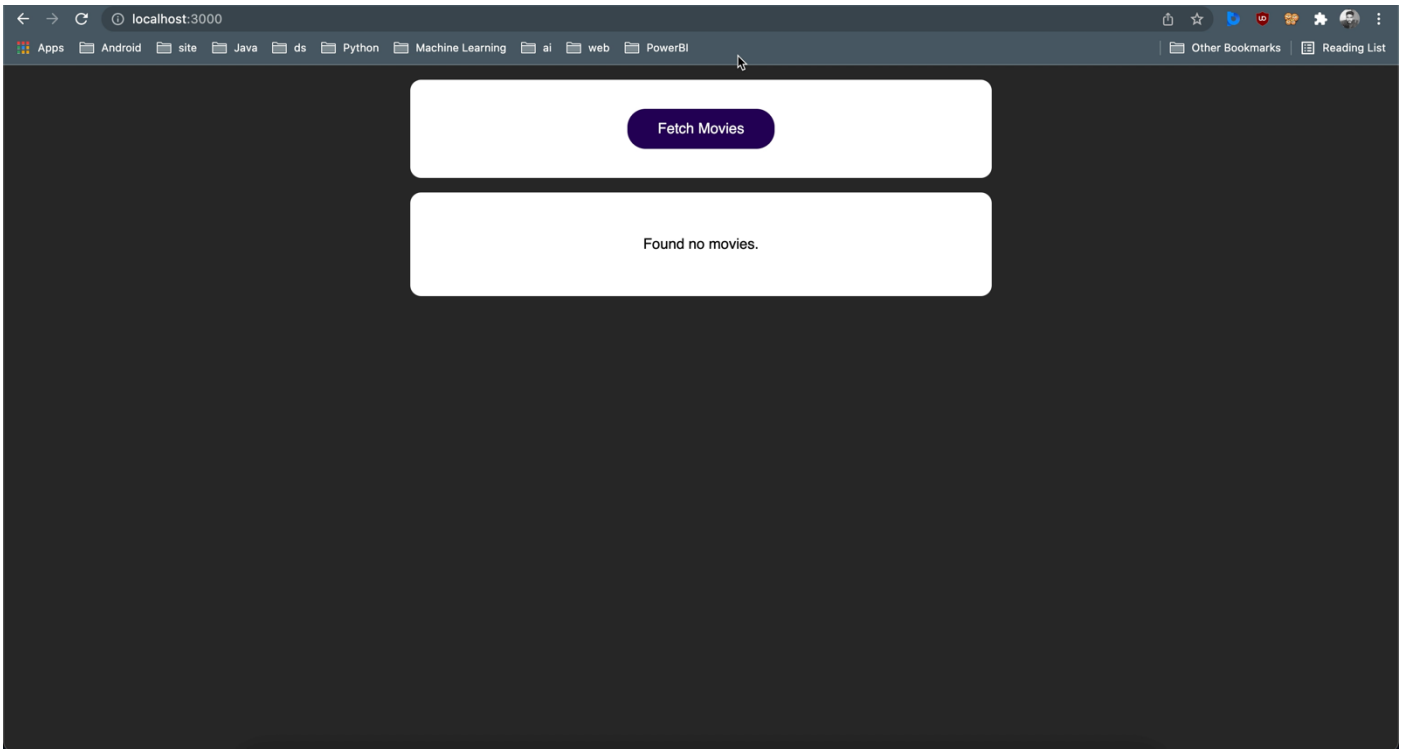
fetch() supports both GET and POST request. By default, GET request.

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.episod_id,
      title: movieData.title,
      openingText: movieData.opening_crawl,
      releaseDate: movieData.release_date
    }
  })
  setMovies(transformedMovies)
}
```

Handling the Loading with States

```jsx
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.episod_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

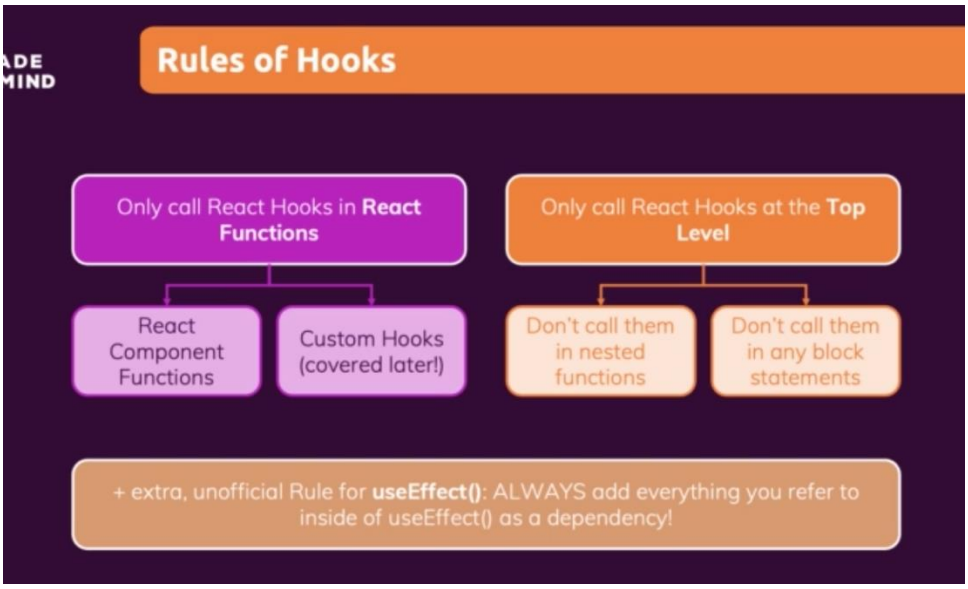
```
const [movies, setMovies] = useState([]);
const [isLoading, setIsLoading] = useState(false);
const [error, setError] = useState(null);

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

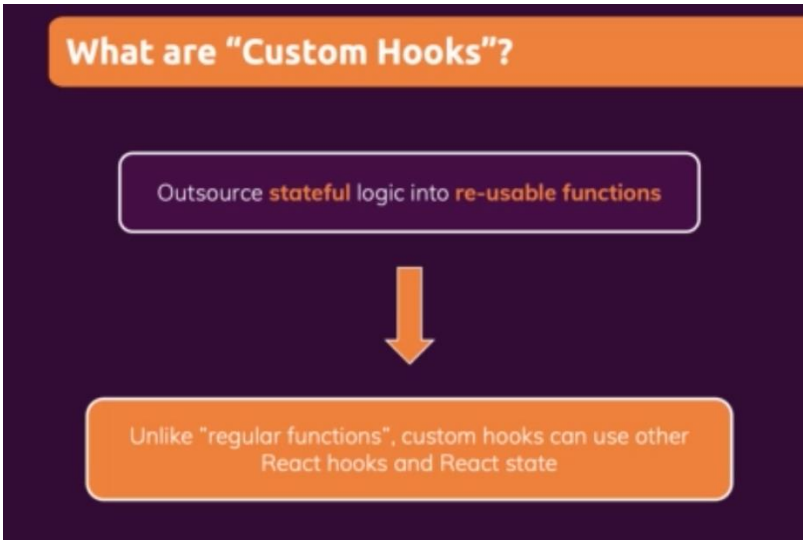
```
async function addMovieHandler(movie) {
  console.log(movie);
  const response = await fetch('https://react-http-5c577-default-rtdb.firebaseio.com/movies.json'
    method: 'POST',
    body: JSON.stringify(movie),
    headers: {
      'Content-Type':'application/json'
    }
  });
  const data = await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

```
const fetchMoviesHandler = useCallback(async () => {
  setIsLoading(true);
  setError(null);
  try {
    const response = await fetch('https://react-http-5c577-default-rtdb.firebaseio.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



What are Custom Hooks ?



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;
```

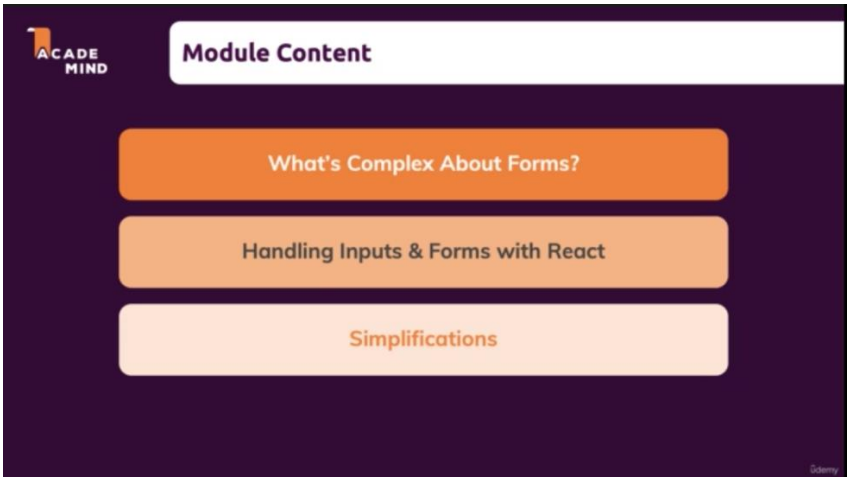
Custom Hook to handling Http Requests

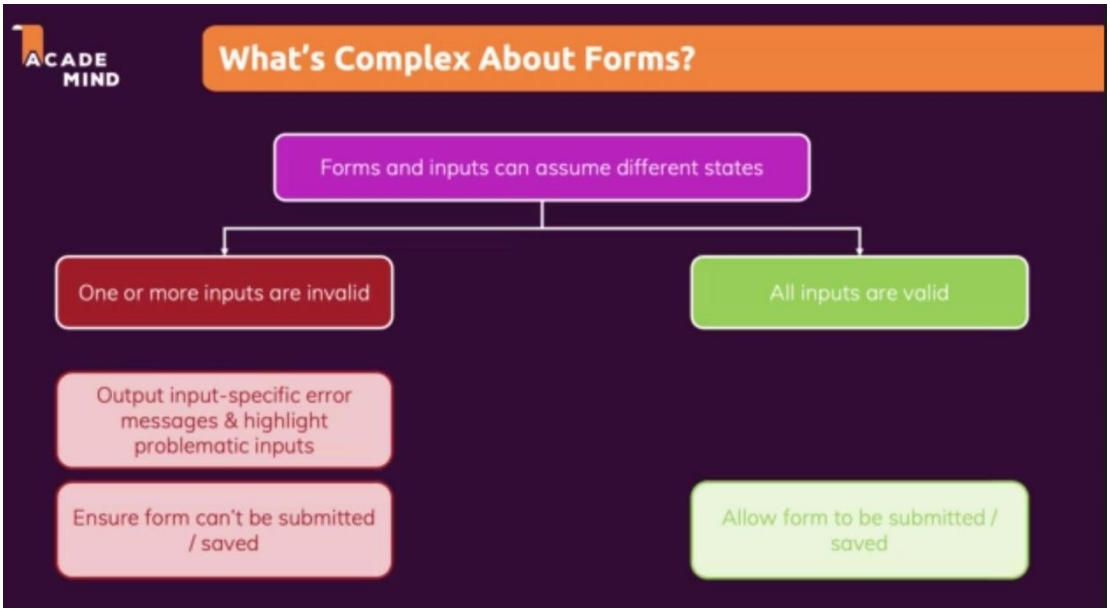
```
Complexity is 11 You must be kidding
const useHttp = () => {

    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-rtdb.firebaseio.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: sendTaskRequest} = useHttp();

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

  const enterTaskHandler = async (taskText) => {
    sendTaskRequest(
      {
        url: 'https://react-http-5c577-default-rtdb.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


ACADEMIND
Module Content
What's Complex About Forms?
Handling Inputs & Forms with React
Simplifications


ACADEMIND
What's Complex About Forms?
Forms and inputs can assume different states
One or more inputs are invalid
All inputs are valid
Output input-specific error messages & highlight problematic inputs
Ensure form can't be submitted / saved
Allow form to be submitted / saved


ACADEMIND
When To Validate?
When form is submitted
When a input is losing focus
On every keystroke
Allows the user to enter a valid value before warning him / her
Allows the user to enter a valid value before warning him / her
Warns user before he / she had a chance of entering valid values
Avoid unnecessary warnings but maybe present feedback "too late"
Very useful for untouched forms
If applied only on invalid inputs, has the potential of providing more direct feedback

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}
          />
```

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}
        />
      {nameInputIsInvalid && <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}
        />
      {nameInputIsInvalid && <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>
```

```
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

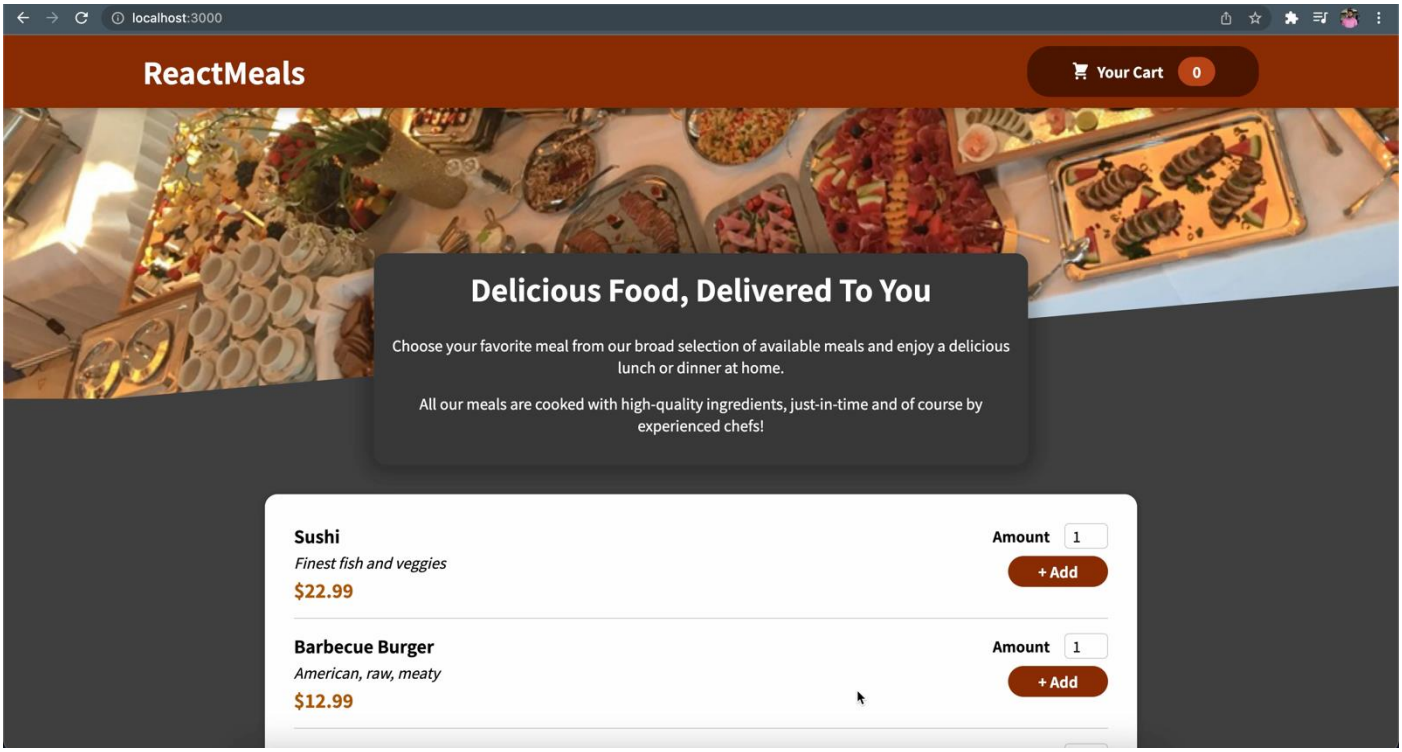
```
import { useReducer } from "react";

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, initialInput

    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

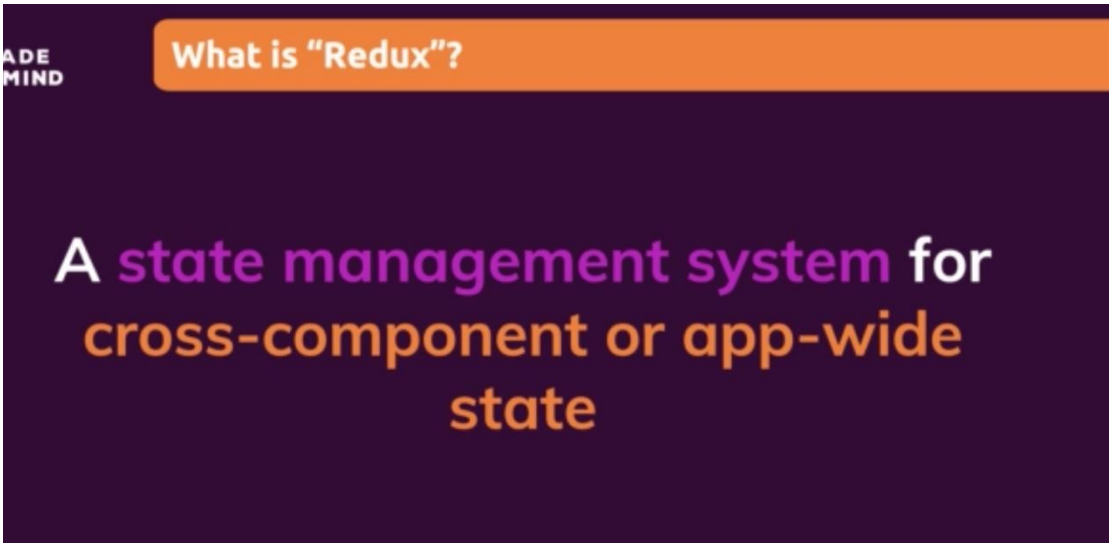
Reusing the useHttp hook

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

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

    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;
```

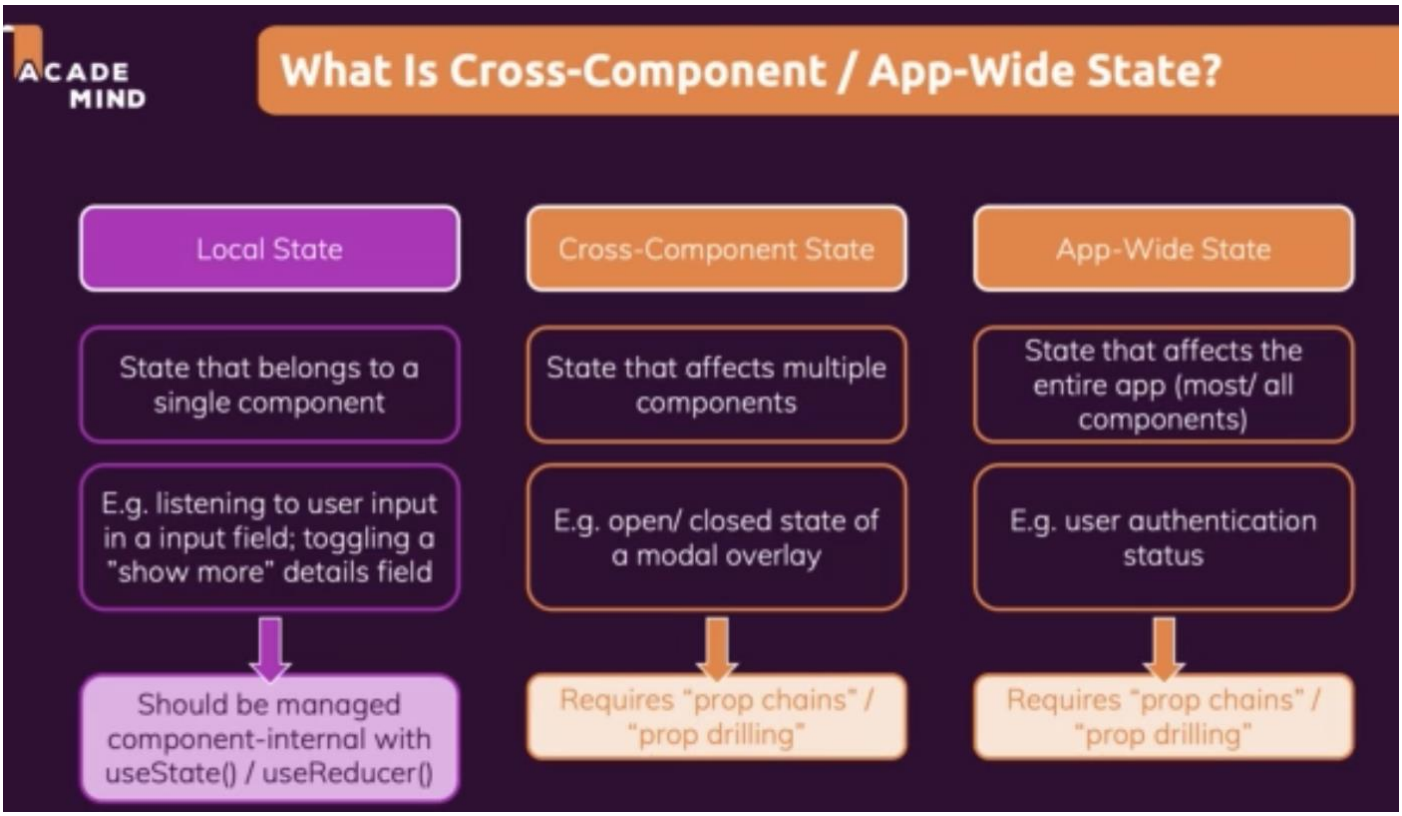
Demo


localhost:3000
ReactMeals
Your Cart 0
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
Amount 1
+ Add
Barbecue Burger
American, raw, meaty
$12.99
Amount 1
+ Add

Redux

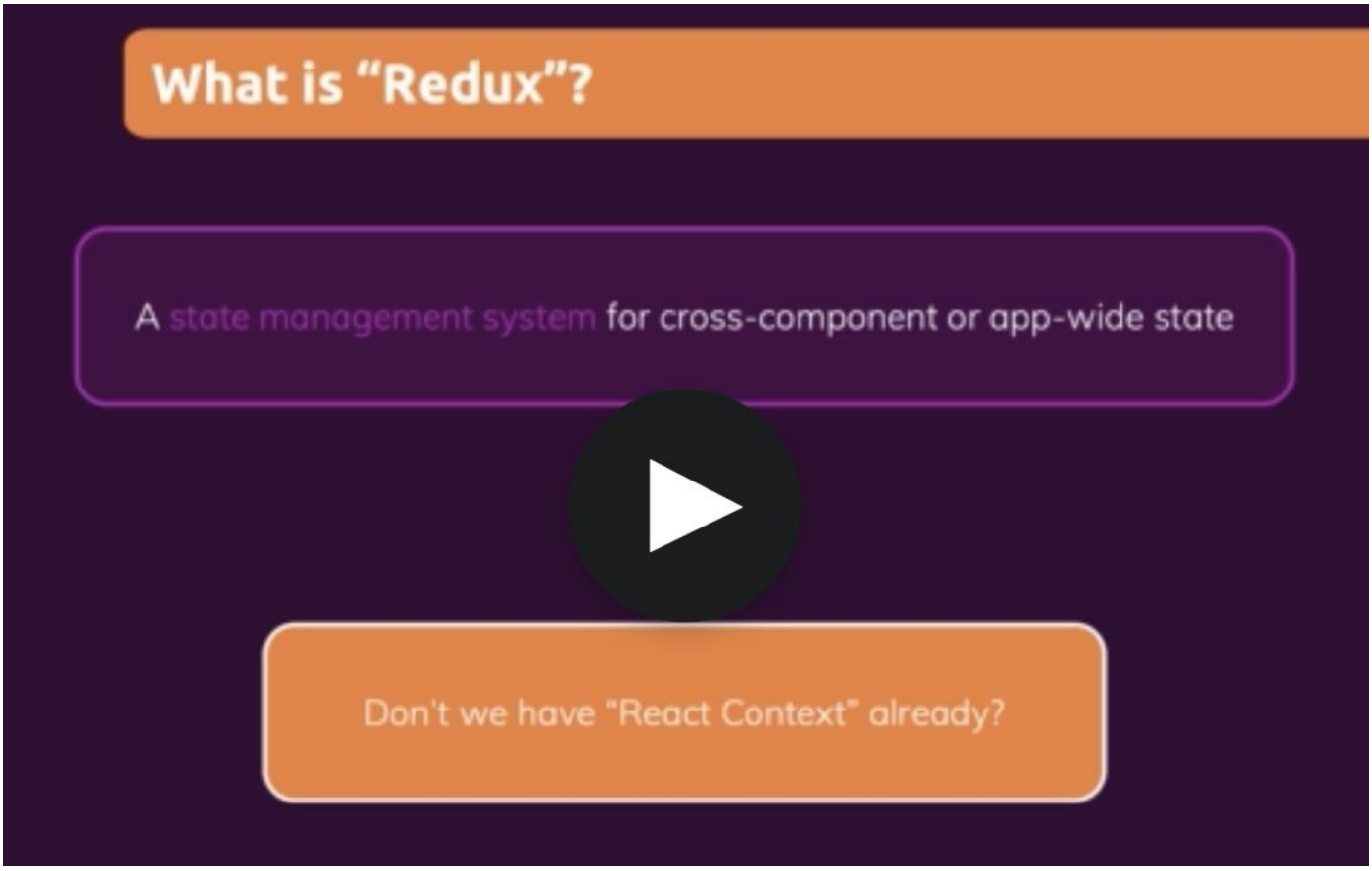

What is "Redux"?
A state management system for cross-component or app-wide state

What is Cross-Component / App-Wide State?

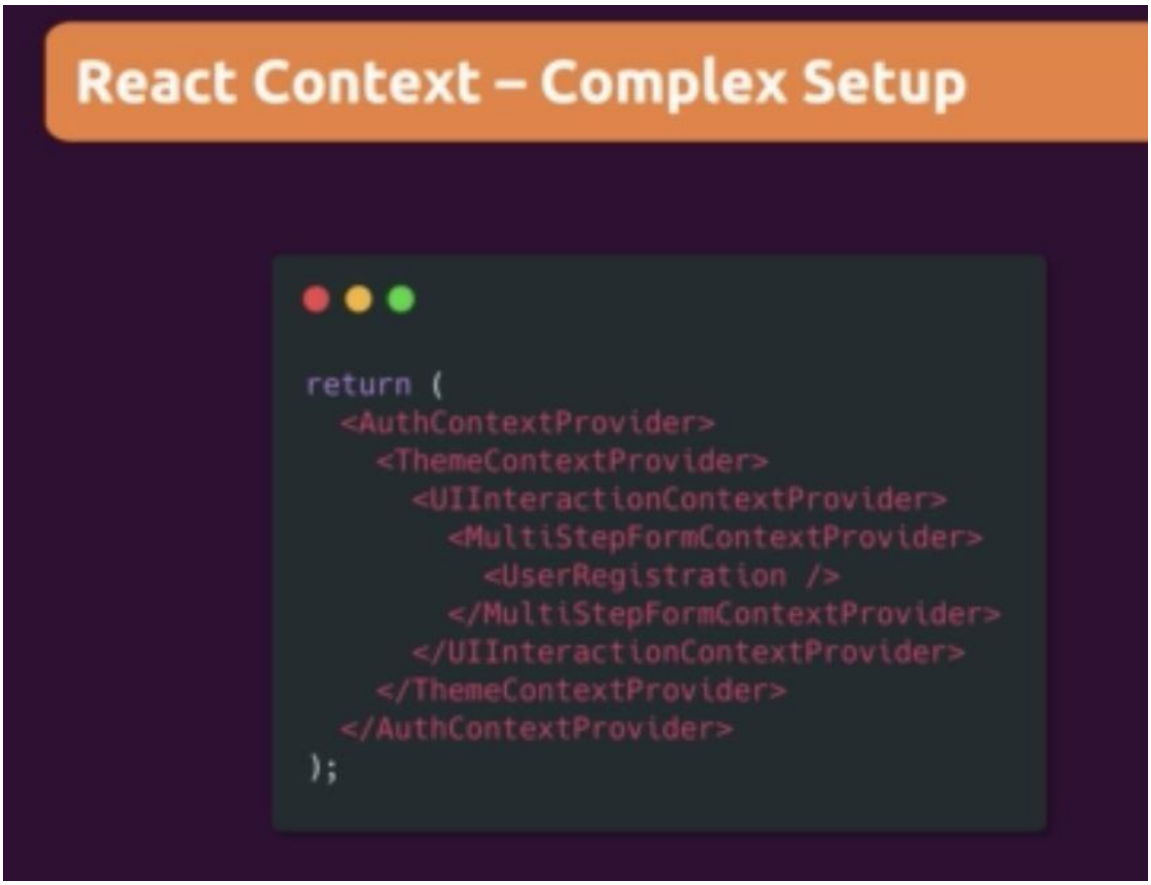


Redux used to manage the cross-component state similar to react context.

Then Why do need a Redux when we already context Provider

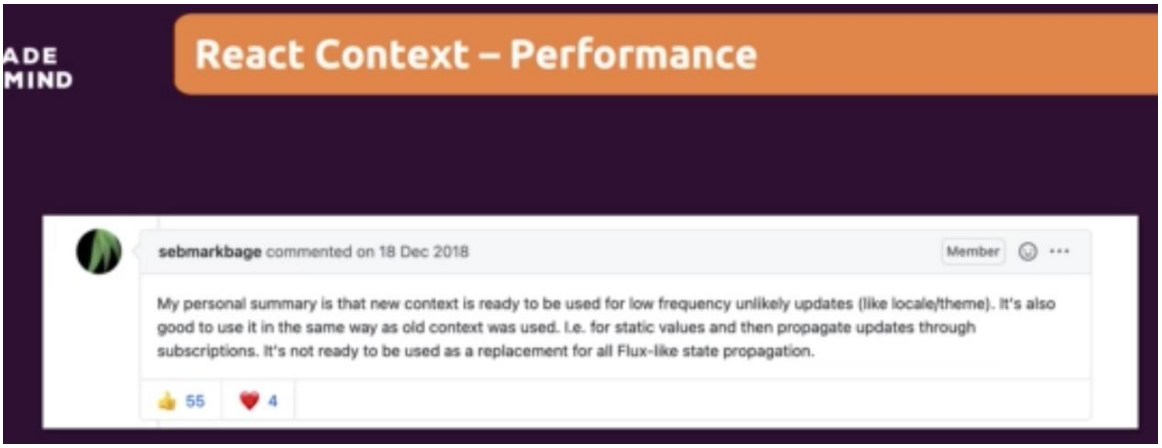


Disadvantage of Context Provider

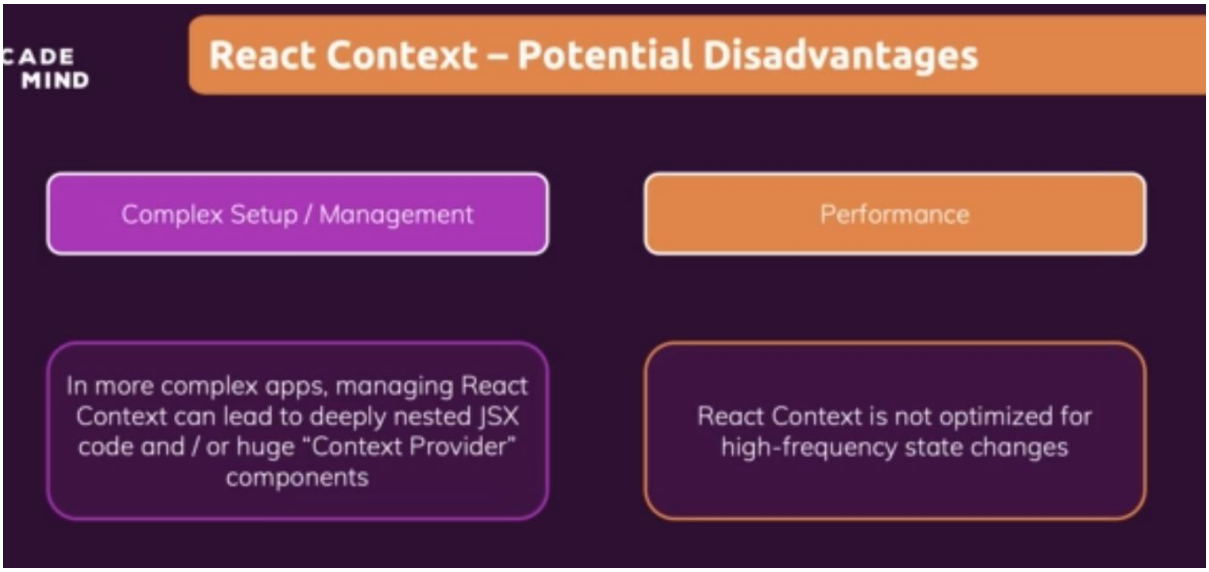



```
function AllContextProvider() {
  const [isAuth, setIsAuth] = useState(false);
  const [isEvaluatingAuth, setIsEvaluatingAuth] = useState(false);
  const [activeTheme, setActiveTheme] = useState('default');
  const [ ... ] = useState(...);

  function loginHandler(email, password) { ... };

  function signupHandler(email, password) { ... };

  function changeThemeHandler(newTheme) { ... };

  ...

  return (
    <AllContext.Provider>
```

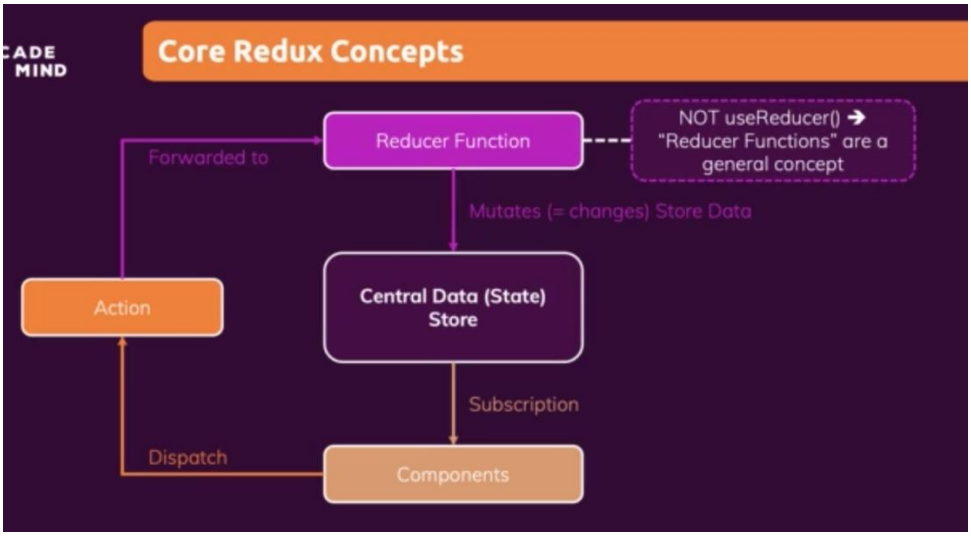
Context provider become large on large enterprise application.



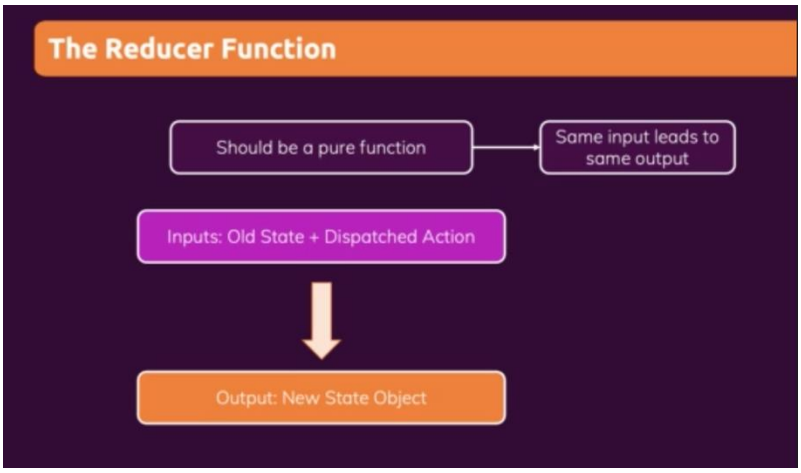
Redux is flux like state propagation.



How Redux Works?



Redux Core Concept



Redux Demo

redux-demo.js package.json

16-redux-demo > redux-demo.js > ...

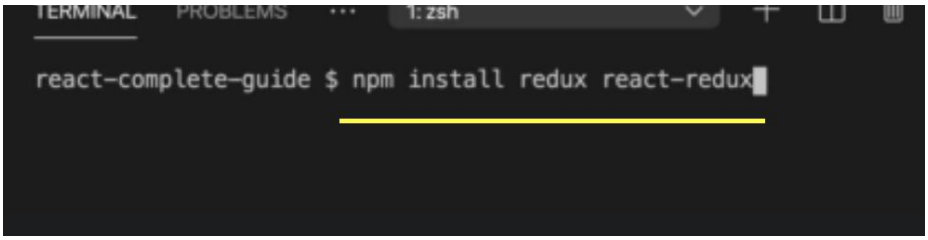
```
1   const redux = require('redux')
2
    Complexity is 5 Everything is cool!
3   const counterReducer = (state = { counter: 0 }, action) => {
4       if (action.type === 'increment') {
5           return {
6             counter: state.counter + 1,
7           };
8       }
9       if (action.type === 'decrement') {
10          return {
11            counter: state.counter - 1,
12          };
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'});
```

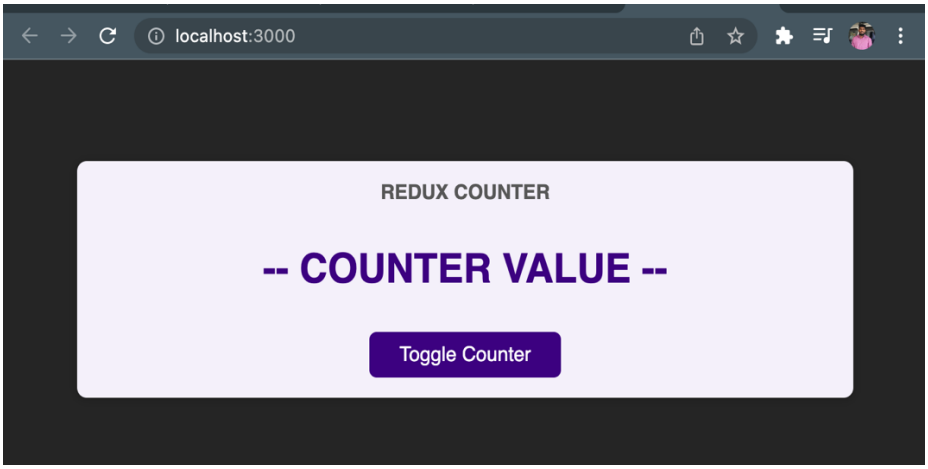
PROBLEMS OUTPUT TERMINAL

TERMINAL ZSH - 16-REDUX-DE

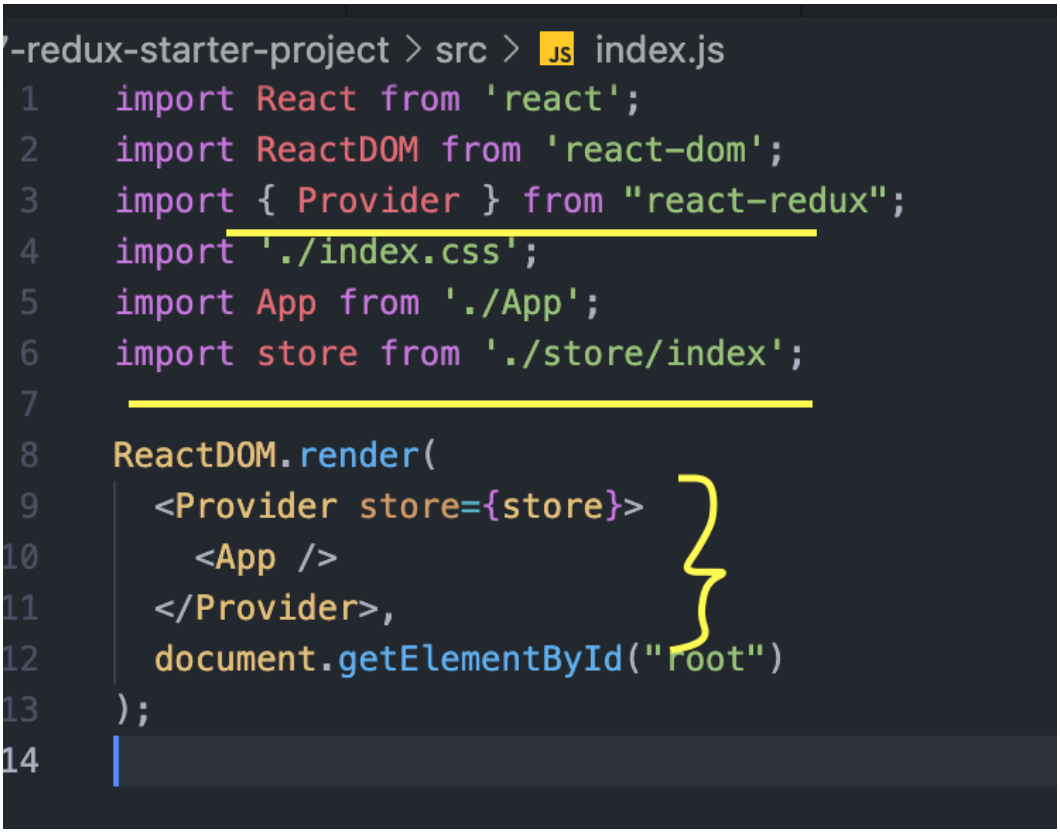
```
→  16-redux-demo node redux-demo.js
{ counter: 1 }
{ counter: 2 }
{ counter: 1 }
→  16-redux-demo
```

Required redux package for react project





Registering the redux store in react app



```
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;
```

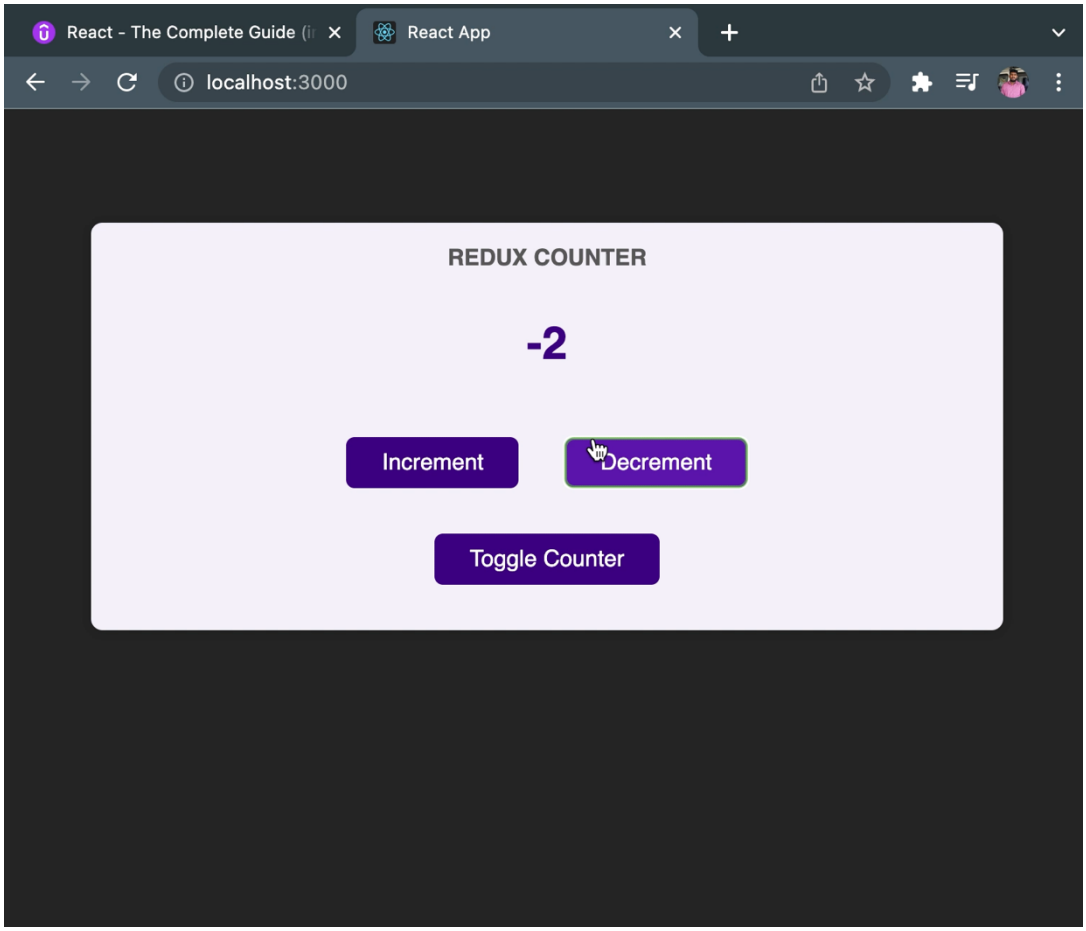
Use the redux store in react component

```
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

```
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</bu
          <button onClick={this.decrementHandler.bind(this)}>Decrement</bu
        </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

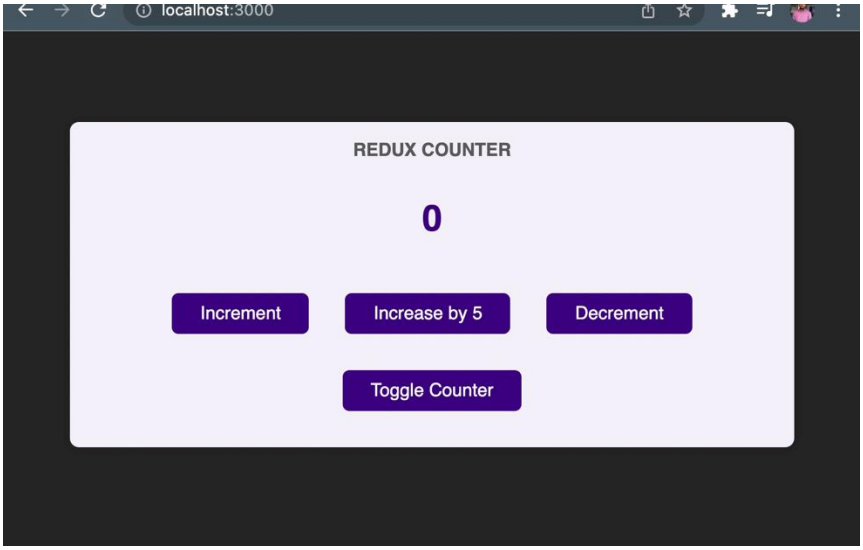


Passing data via dispatch action

```
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;
```

```jsx
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

```
import { useSelector, useDispatch } from 'react-redux';

import classes from './Counter.module.css';
import { INCREMENT } from '../store/index';

const Counter = () => {
  const dispatch = useDispatch();
  const counter = useSelector((state) => state.counter);
  const show = useSelector((state) => state.showCounter);

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

  const increaseHandler = () => {
    dispatch({ type: 'increase', amount: 10 });
  };
```

```
TERMINAL  PROBLEMS  OUTPUT  ...  1: node
```

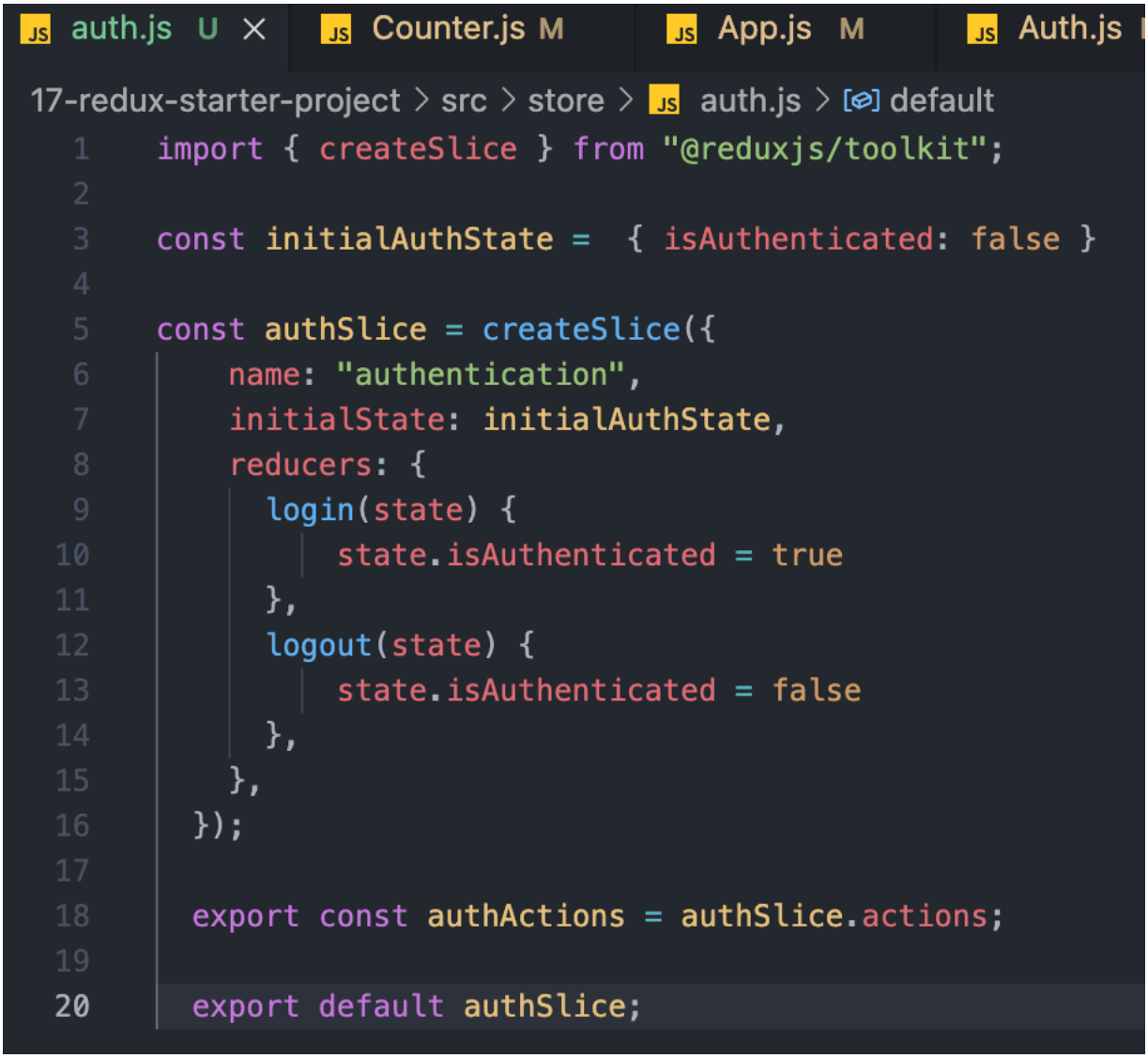
```
TERMINAL  PROBLEMS  OUTPUT  ...  1: zsh

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;
```

```
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>
```

Using Multiple States



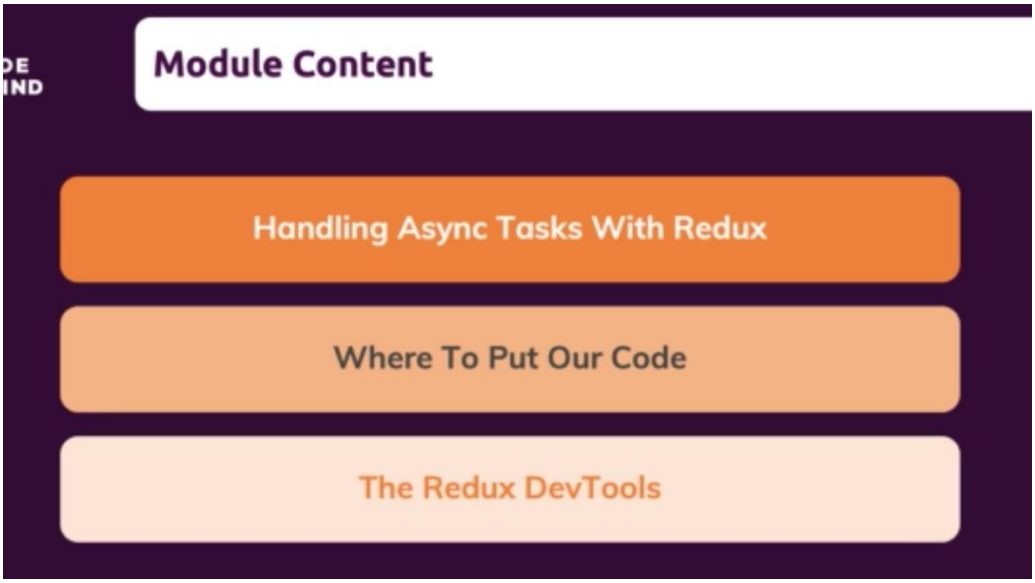

17-redux-starter-project > src > components > Header.js > ...
import { authActions } from '../store/auth.js';
Complexity is 15 You must be kidding
const Header = () => {
const dispatch = useDispatch();
const isAuth = useSelector(state => state.auth.isAuthenticated)
const logoutHandler = (event) => {
dispatch(authActions.logout());
}
return (
<header className={classes.header}>
<h1>Redux Auth</h1>
{isAuth && (
<nav>

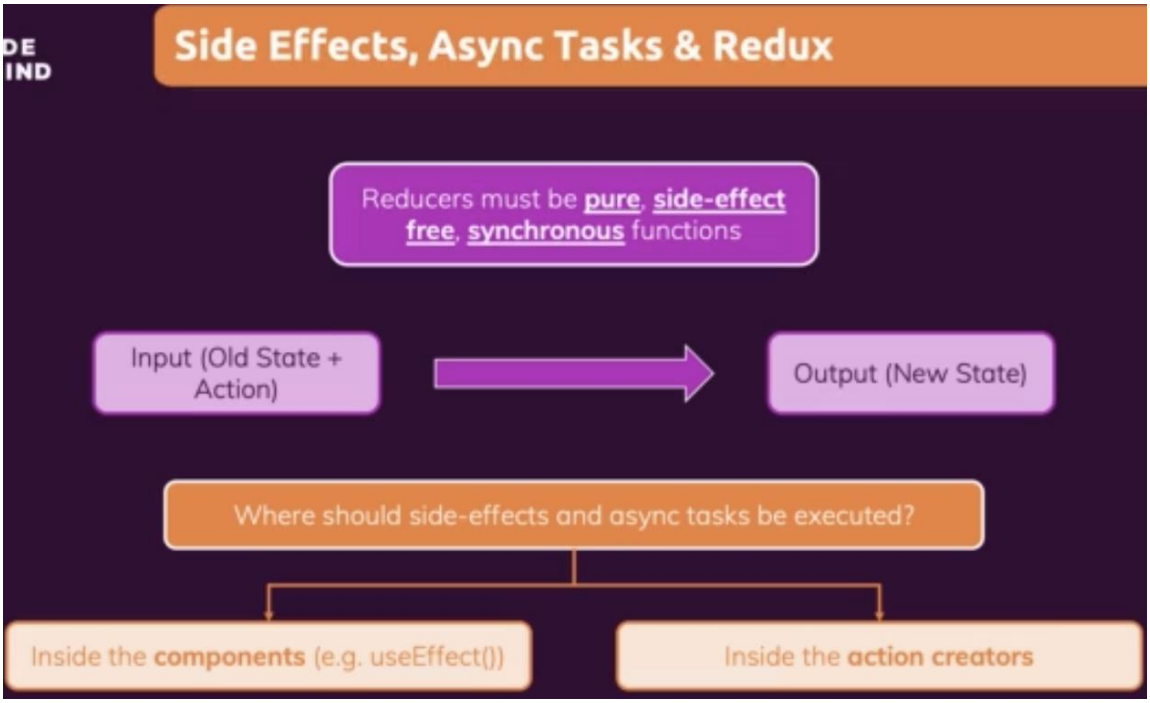
My Products

My Sales

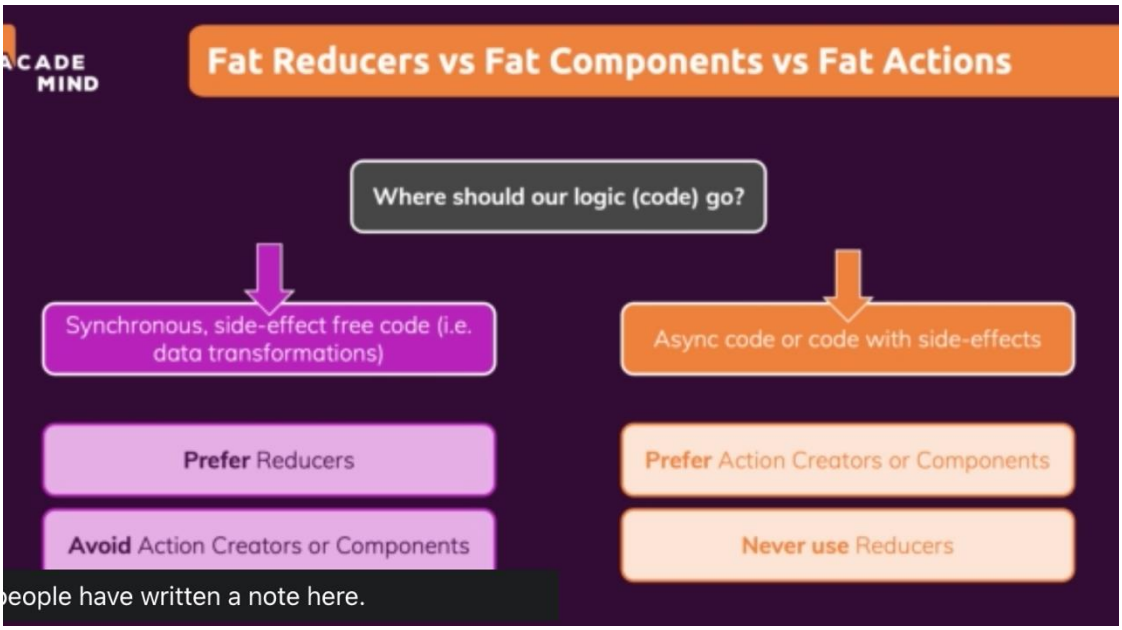
<button onClick={logoutHandler}>Logout</button>

</nav>
)}
</header>
);





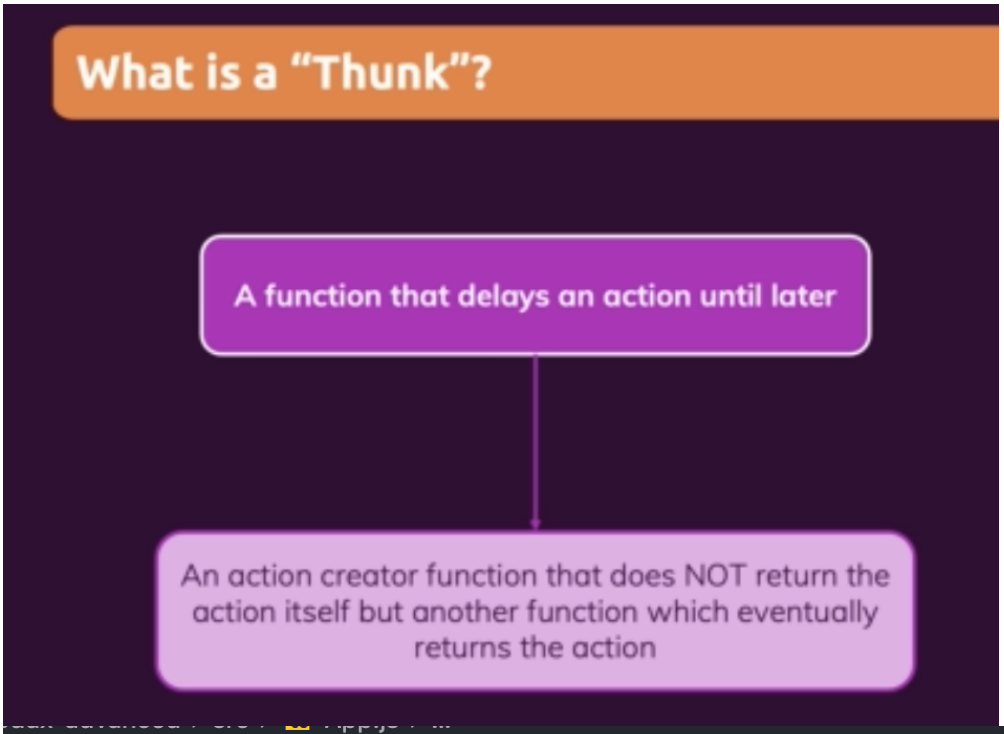
npm install @reduxjs/toolkit
npm install react-redux




```jsx
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;
```

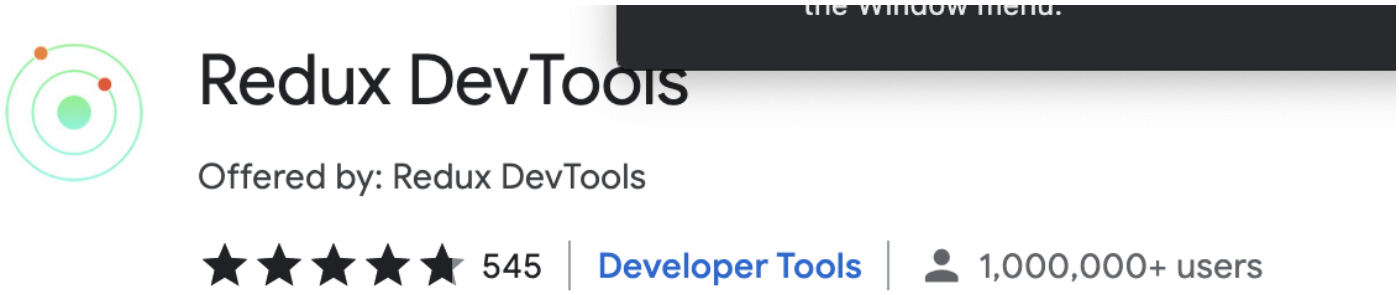
Adding the notification.

```
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-rtdb.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]);
```

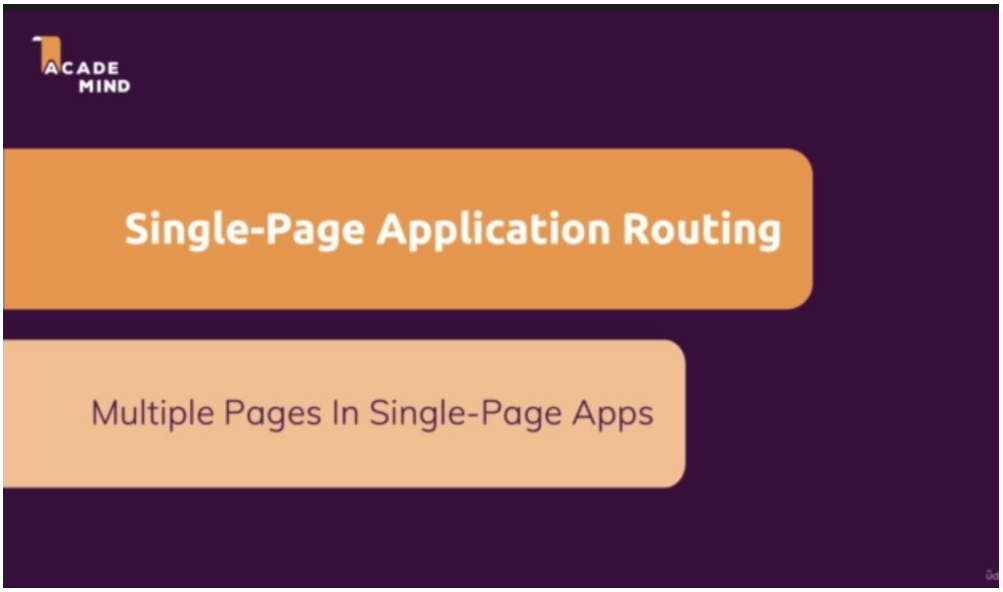


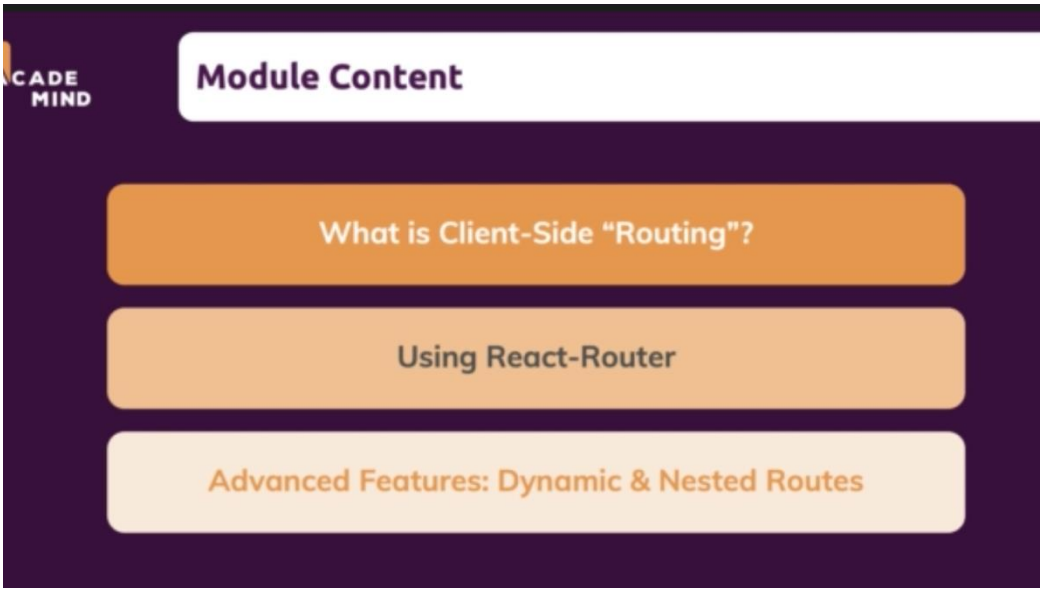

```jsx
import { uiActions } from "./store/ui-slice";
import Notification from './components/UI/Notification';
import { sendCartData } from "./store/cart-slice";

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-rtdb.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!",
              })
            );
        }

    };
};
```

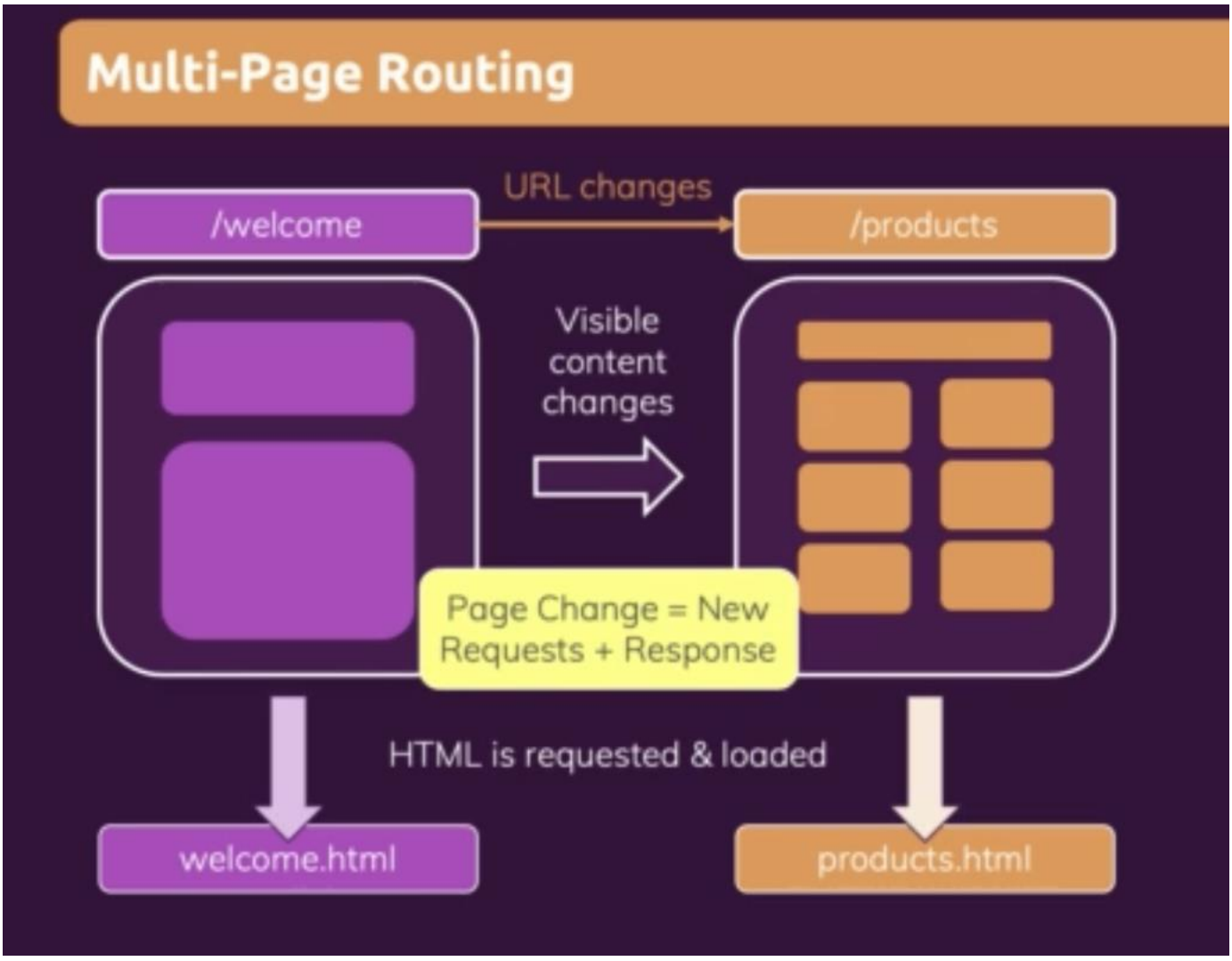


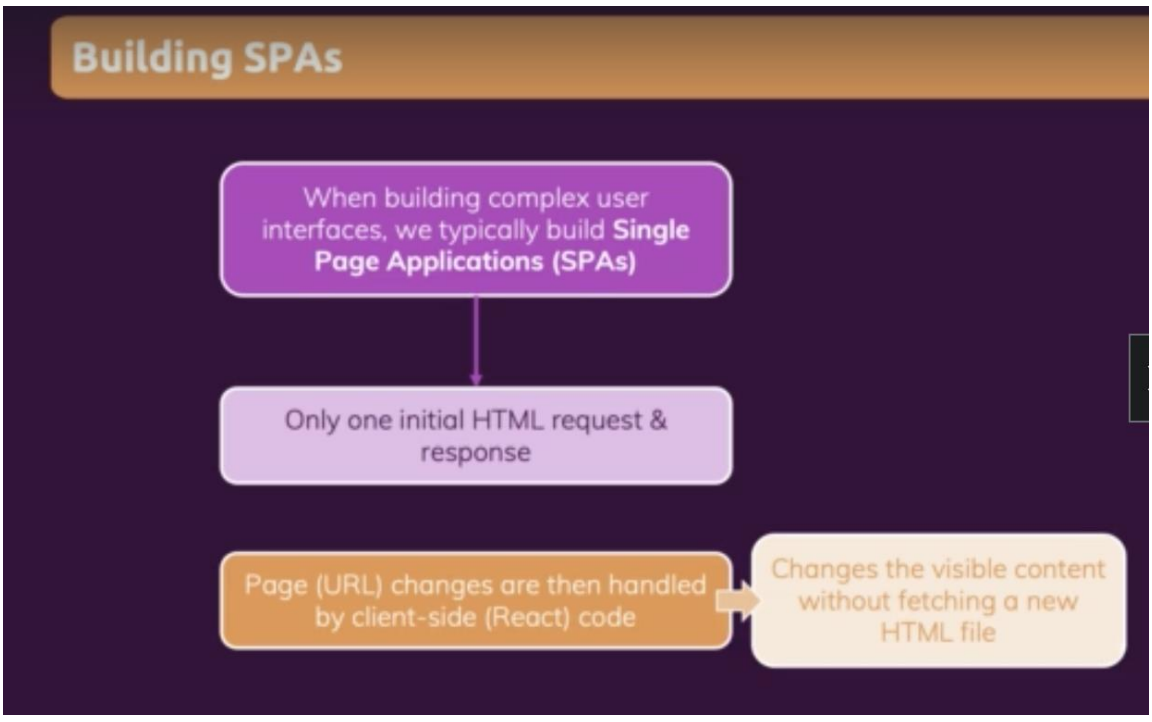
React Router





Why routing?


Multi-Page Routing
URL changes
/welcome
/products
Visible content changes
Page Change = New Requests + Response
HTML is requested & loaded
welcome.html
products.html


Building SPAs
When building complex user interfaces, we typically build Single Page Applications (SPAs)
Only one initial HTML request & response
Page (URL) changes are then handled by client-side (React) code
Changes the visible content without fetching a new HTML file

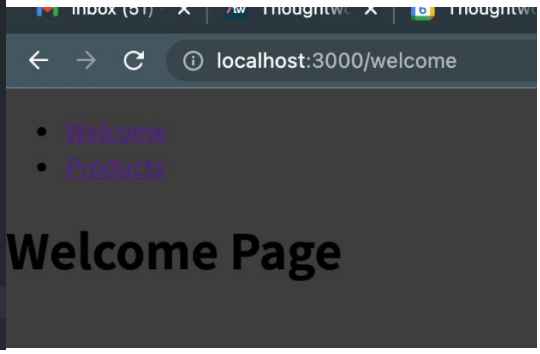
Router Basic

```
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;
```

```
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

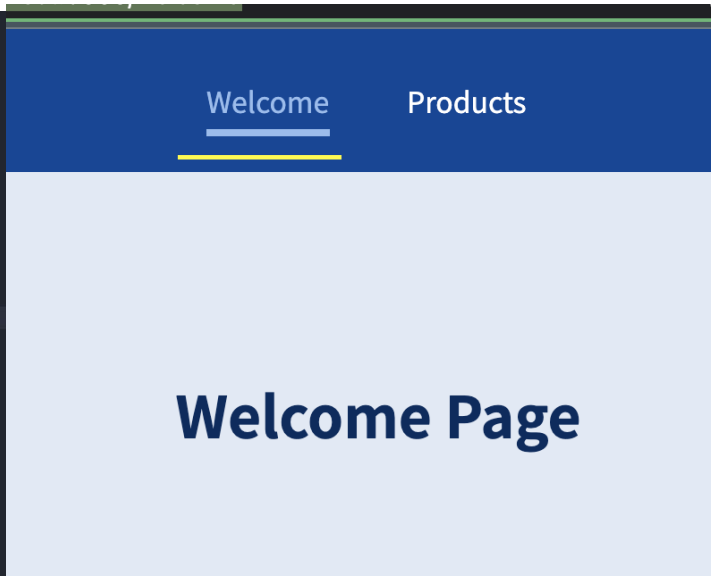
```jsx
import { Link } from "react-router-dom";

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;
```



Using the Link will prevent the page reload.

Using NavLinks to navigate to Route

```jsx
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;
```

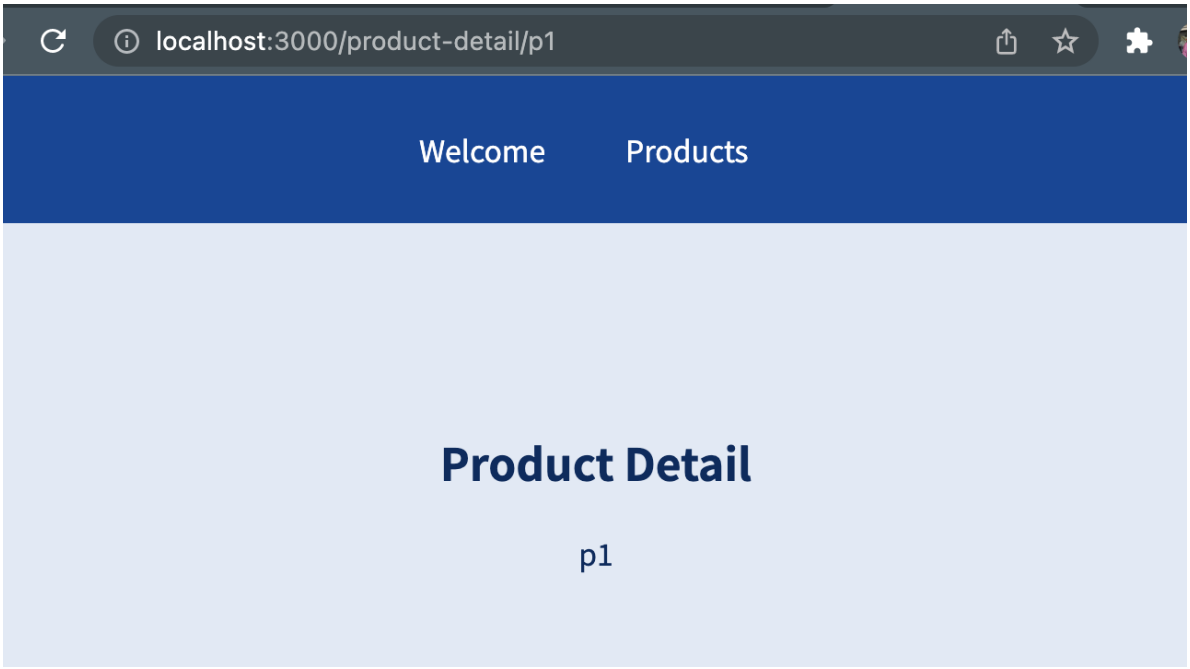


NavLink is like Link. It helps to add a CSS class to active link.

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;
```

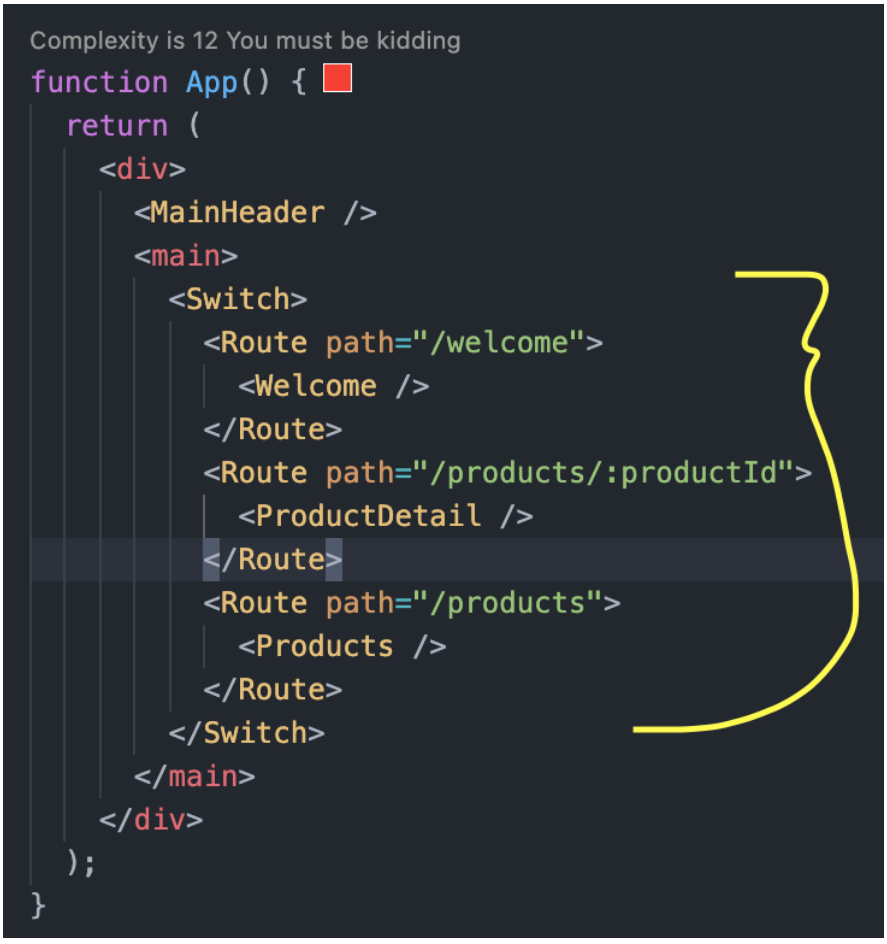
```
import { useParams } from "react-router-dom";

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

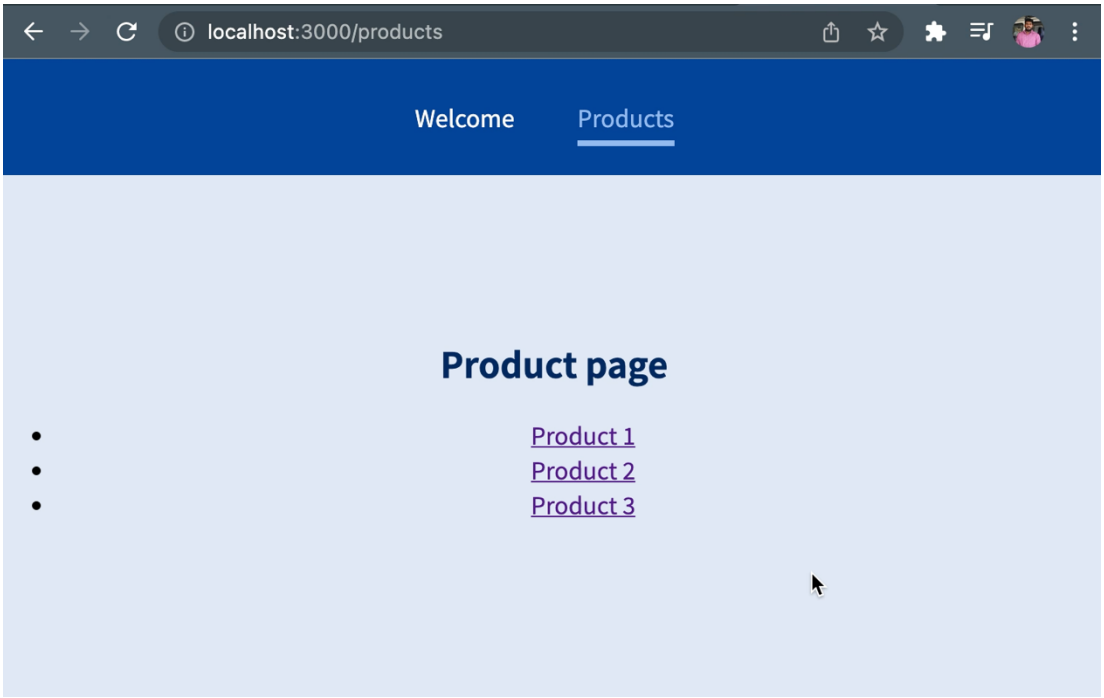
Switch considers the <Route> orders

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



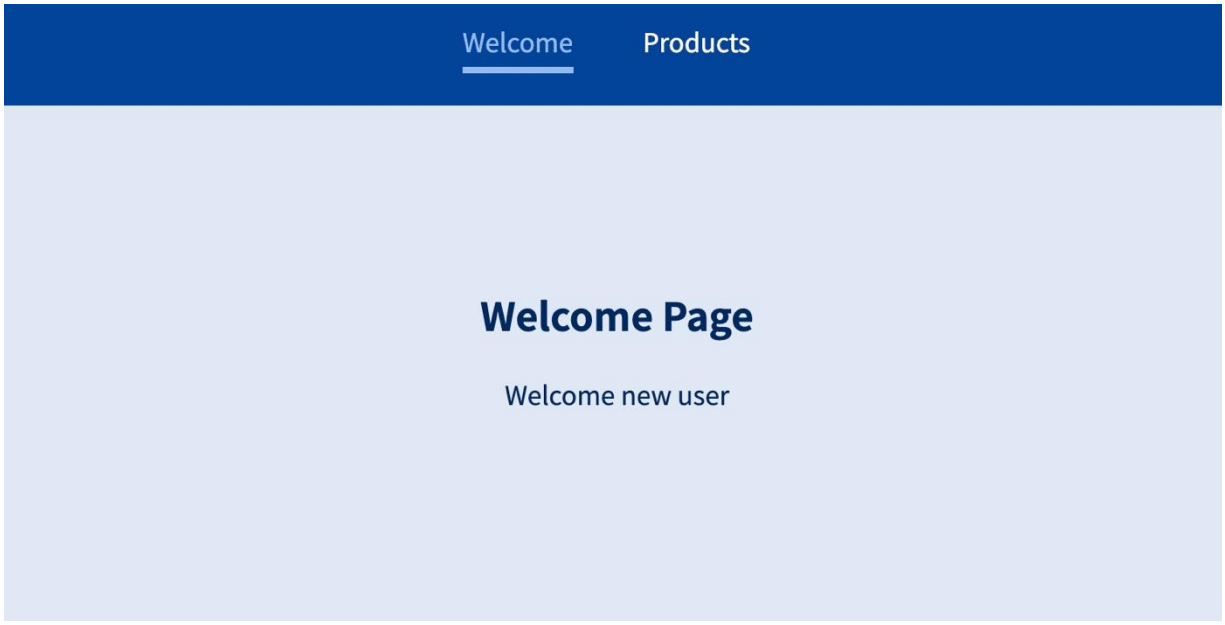
If we don’t want the <Switch> to consider the order, we can use “exact”.

```
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>
  );
}
```

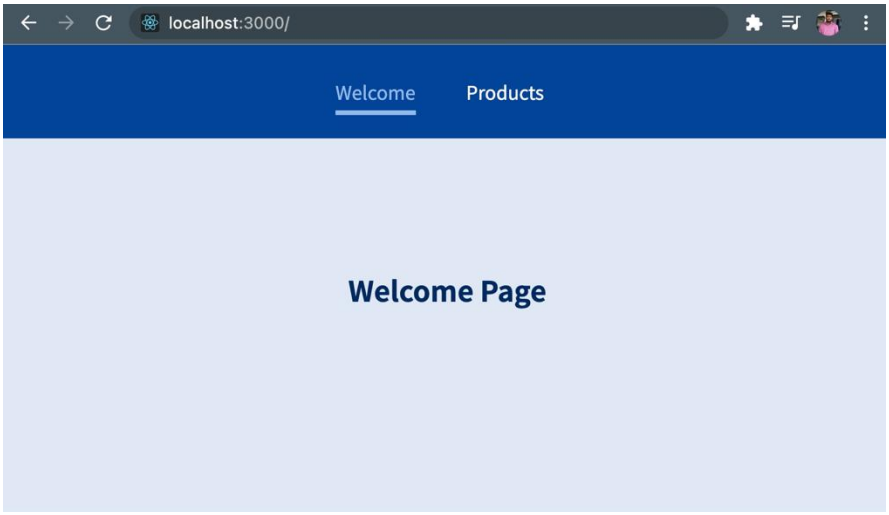


Working with Nested Routes

```
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



```jsx
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;
```

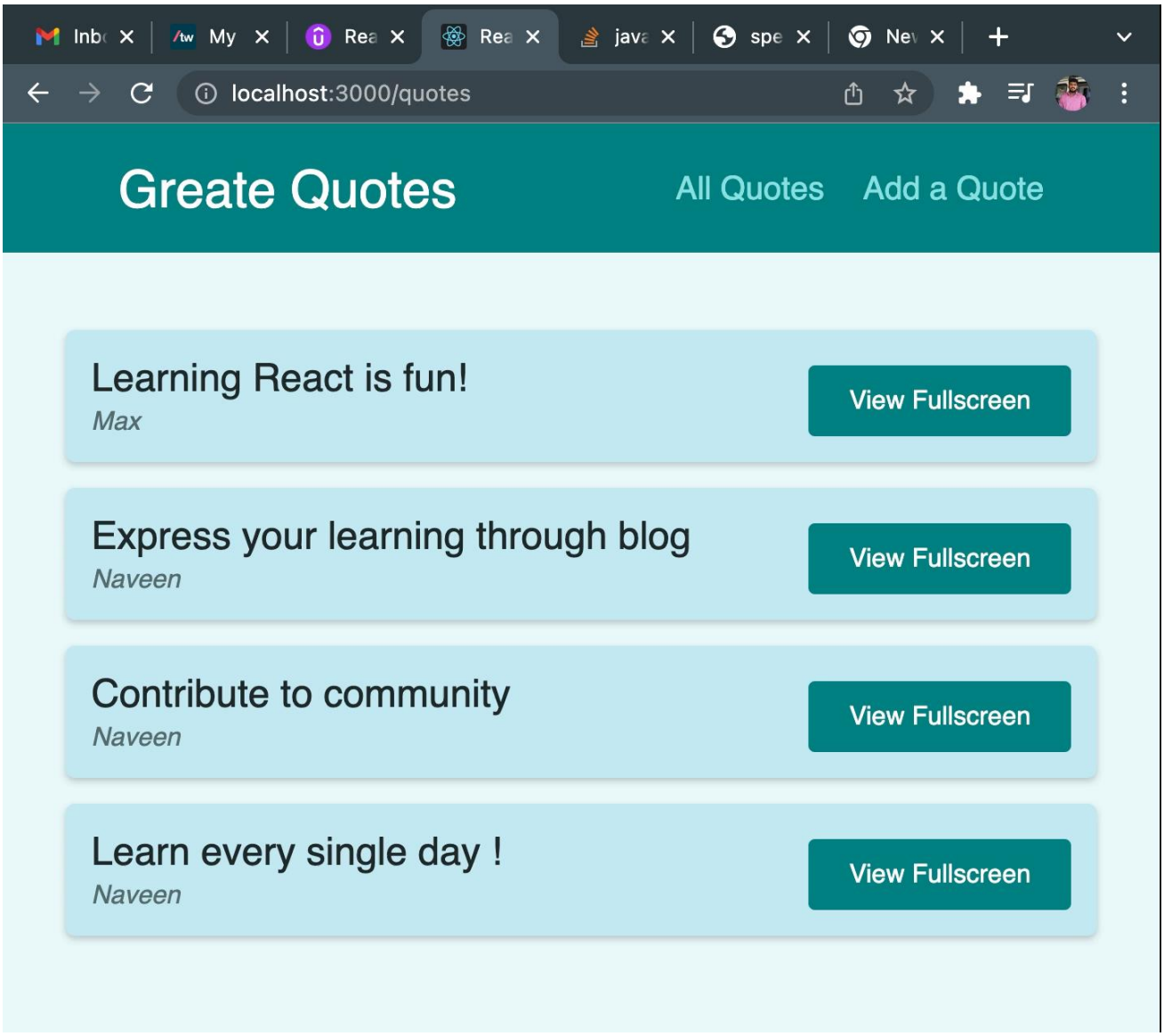
Route for handling unknown URL paths

```jsx
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>
    </L
```

Programmatic Navigation

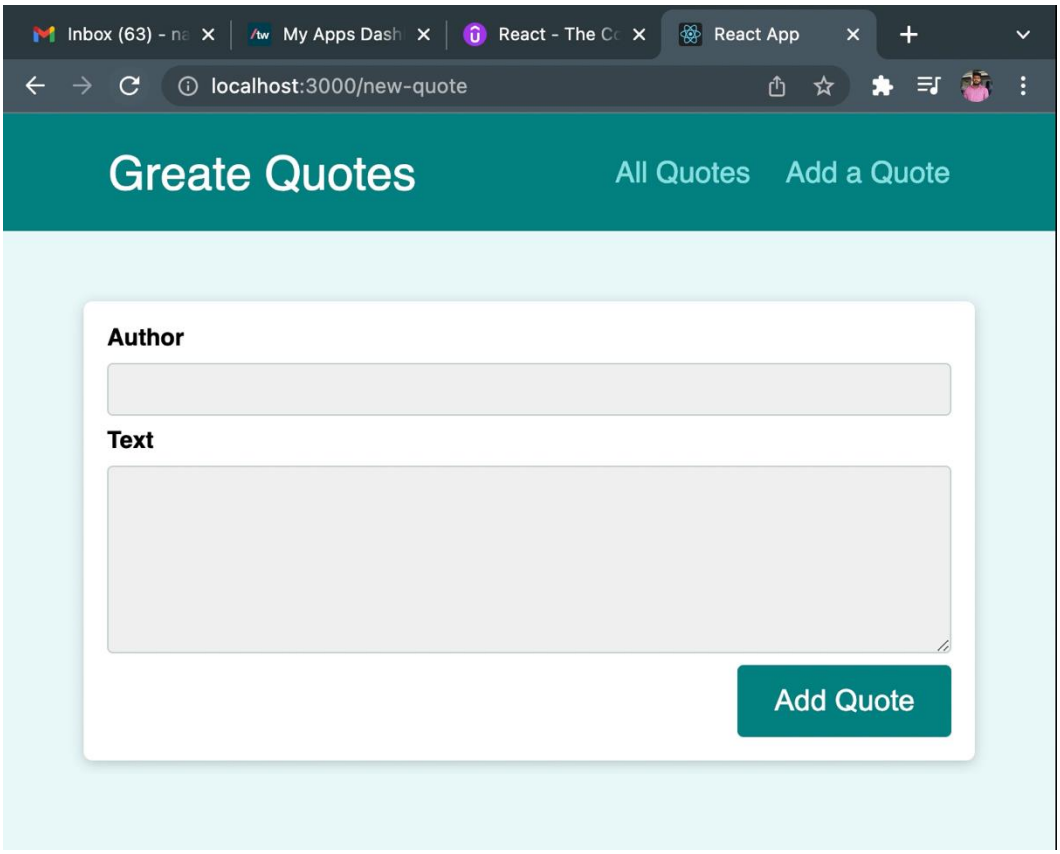
```
Complexity is 4 Everything is cool!
const NewQuote = () => {

    const history = useHistory();

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

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

Quote App Implementation So Far

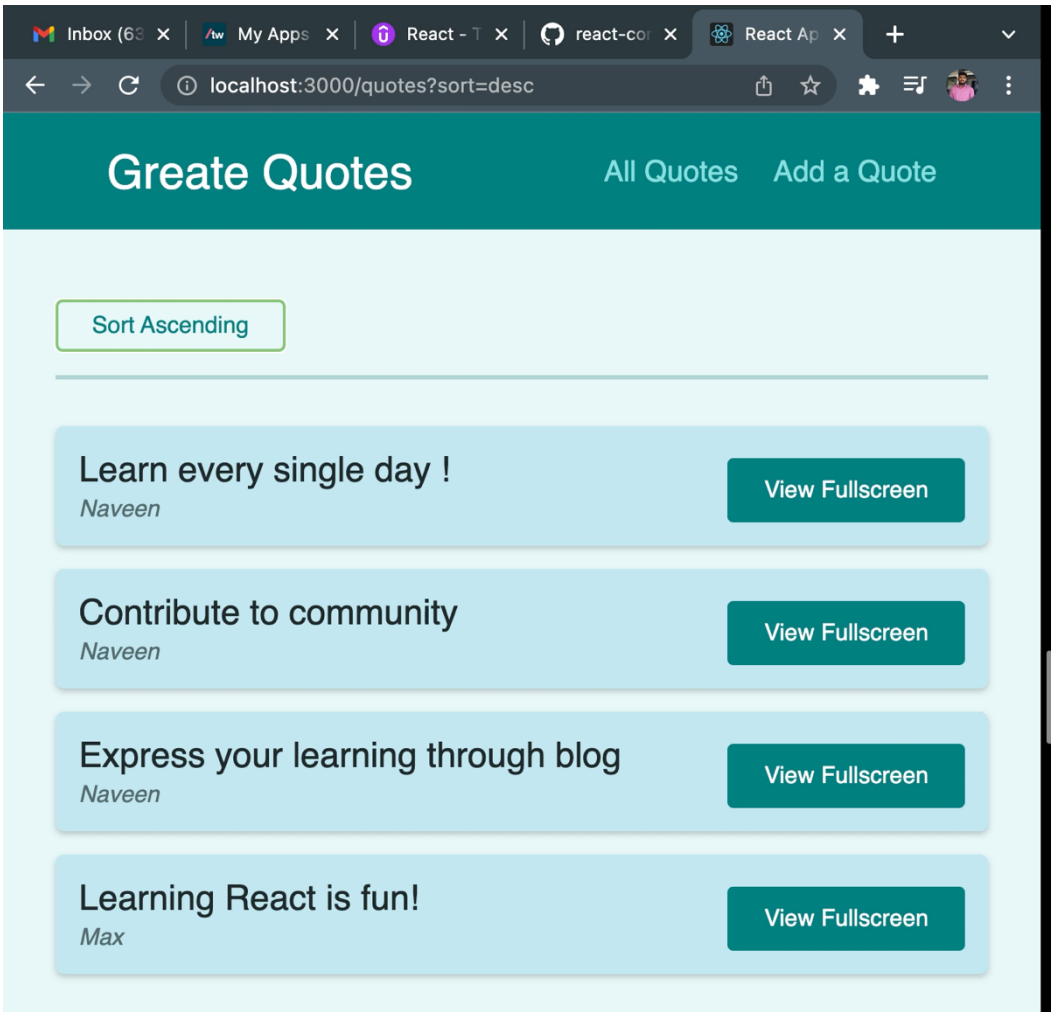


Prompting User on Navigation

```
}

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}
```

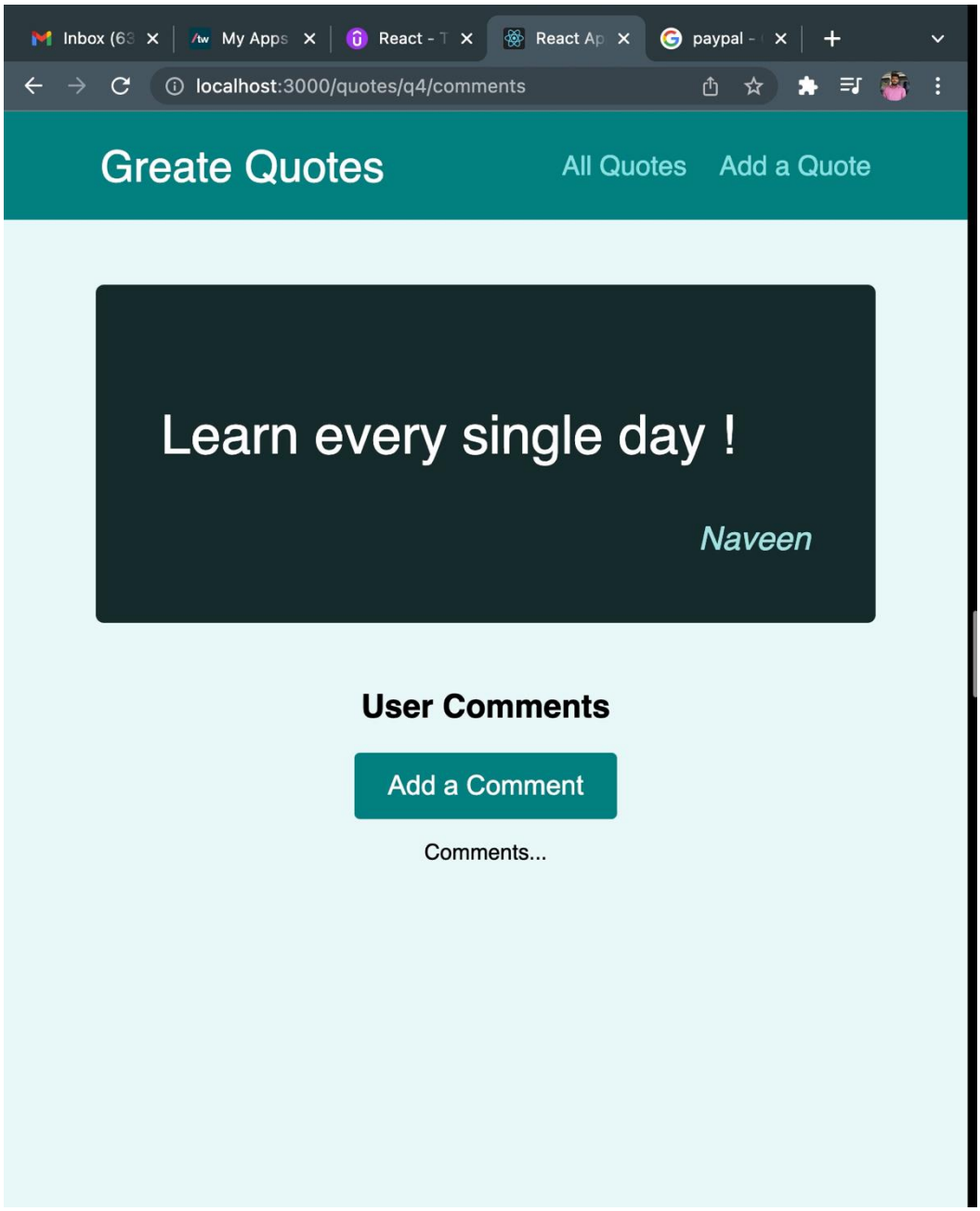


Working with Query Parameters



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"}`,
  });
};
```

```
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>
  );
}
```

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 className={(navData) => navData.isActive ? classes.active : '' } to='
 Welcome
/NavLink>
i>
>
NavLink className={(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>
BLEMS  OUTPUT  TERMINAL  SQL CONSOLE  DEBUG CONSOLE  node
```

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.

```
1  mport { Routes, Route } from 'react-router-dom';
2
3  onst 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>
0     </section>
1   );
2  ;
3
4  xport default Welcome;
5
```

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>
  );
}
```

OBLEMS OUTPUT TERMINAL SQL CONSOLE DEBUG CONSOLE node

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>
```

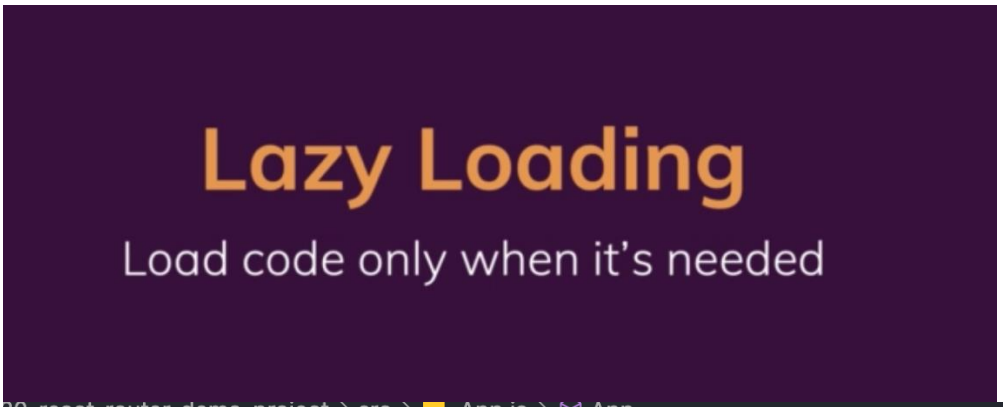
We can specify number for navigate back and forth.

```
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

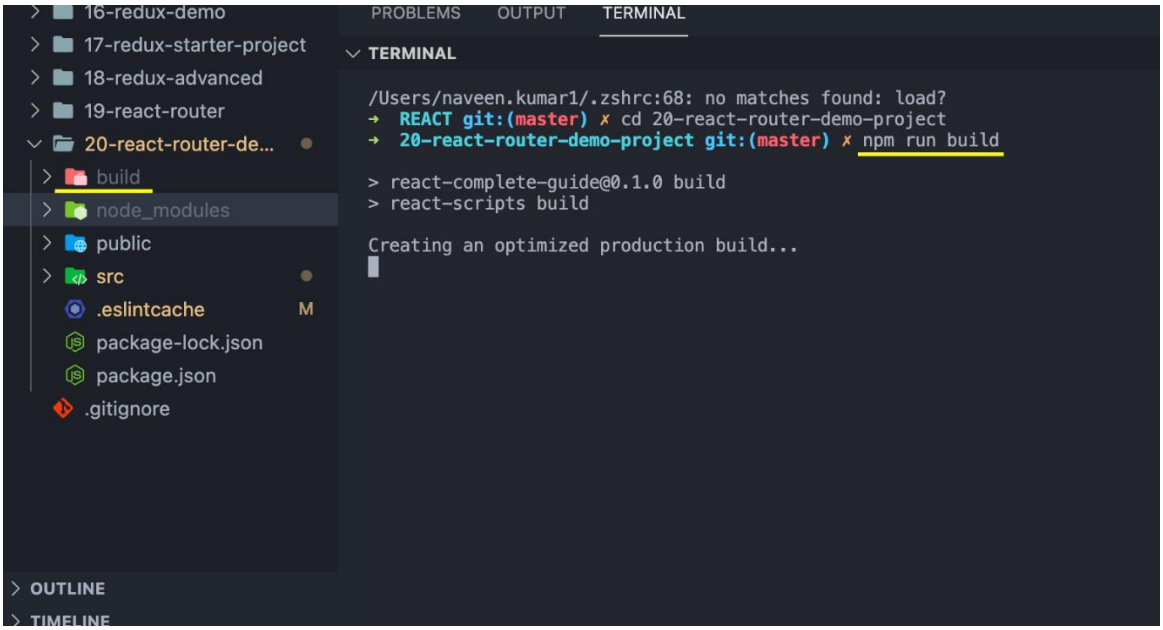
Deployment Step



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

```
import React, { Suspense } from 'react';
import {Redirect, Route, Switch} from 'react-router-dom';
import Layout from './components/layout/Layout';
import LoadingSpinner from './components/UI/LoadingSpinner';

const NewQuote = React.lazy(()=>import('./pages/NewQuotes'));
const AllQuotes = React.lazy(()=>import('./pages/AllQuotes'));
const NotFound = React.lazy(()=>import('./pages/NotFound'));
const QuoteDetails = React.lazy(()=>import('./pages/QuoteDetails'));

Complexity is 17 You must be kidding
function App() {
  return (
    <Layout>
      <Suspense
        fallback={
          <div className="centered">
            <LoadingSpinner />
          </div>
        }
      >
        <Switch>
          <Route path="/" exact>
            <Redirect to="/quotes" />
          </Route>
          <Route path="/quotes" exact>
            <AllQuotes />
          </Route>
          <Route path="/quotes/:quoteId">
            <QuoteDetails />
          </Route>
          <Route path="/new-quote">
            <NewQuote />
          </Route>
          <Route path="*">
            <NotFound />
          </Route>
        </Switch>
      </Suspense>
    </Layout>
  );
}

export default App;
```

Building the code for production



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 ve
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 ve
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.

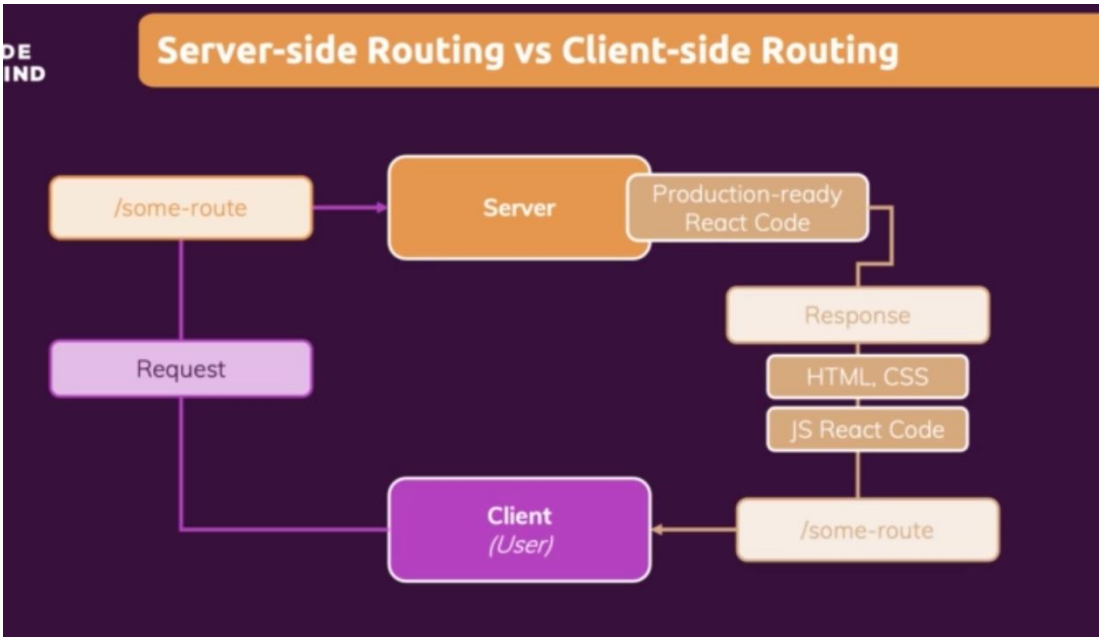
Next

```
→  20-react-router-demo-project git:(master) ✗ firebase login
i  Firebase optionally collects CLI usage and error reporting information to help improve our produ
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
 ○ Realtime Database: Configure a security rules file for Realtime Database and (optionally) provision
 ○ Firestore: Configure security rules and indexes files for Firestore
 ○ Functions: Configure a Cloud Functions directory and its files
❯◉ Hosting: Configure files for Firebase Hosting and (optionally) set up GitHub Action deploys
 ○ Hosting: Set up GitHub Action deploys
 ○ Storage: Configure a security rules file for Cloud Storage
 ○ 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-99dc5 (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
```

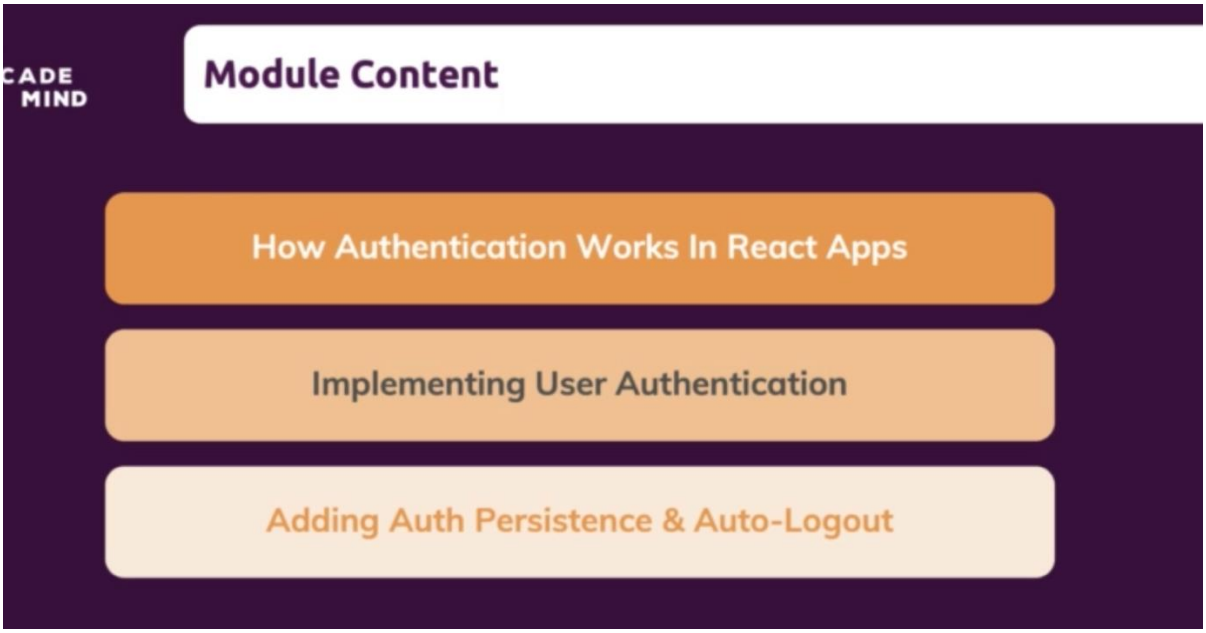


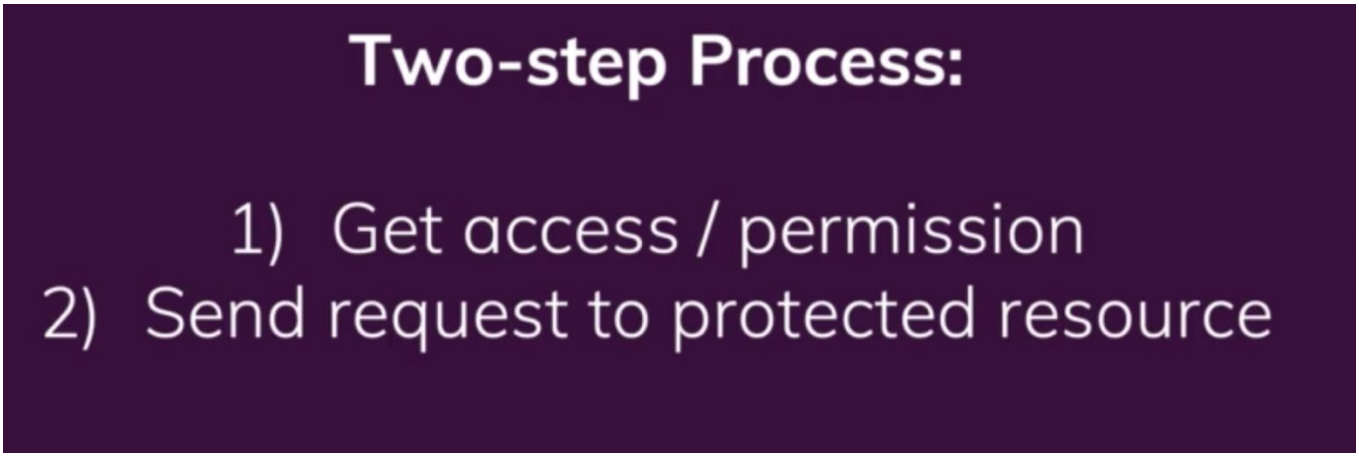
Since we are having a single page application, we need to configure the server accordingly, otherwise server will consider the new url as separate request.

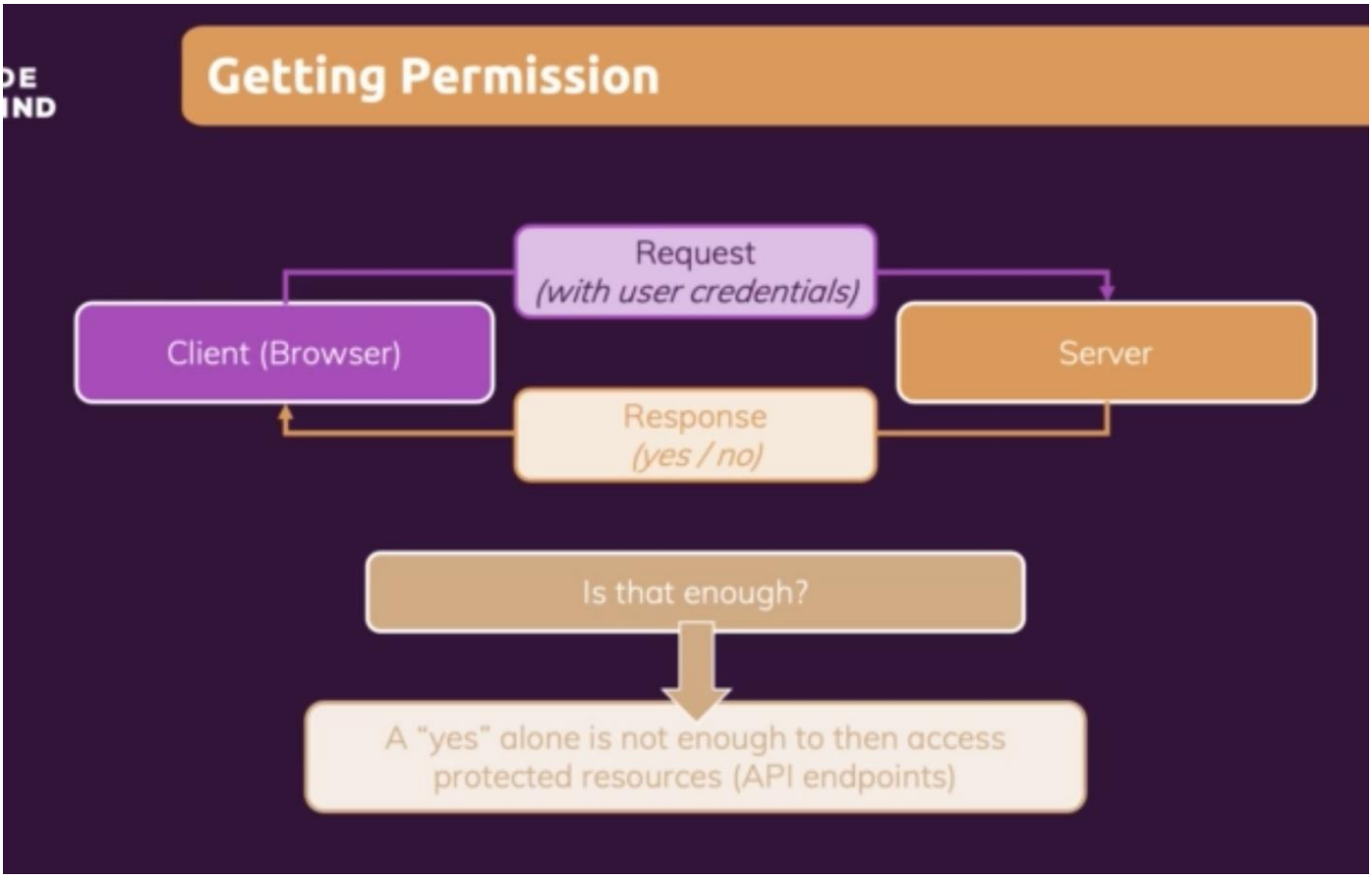
```
You're about to initialize a Firebase project in this directory:

  /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 fea
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!
```

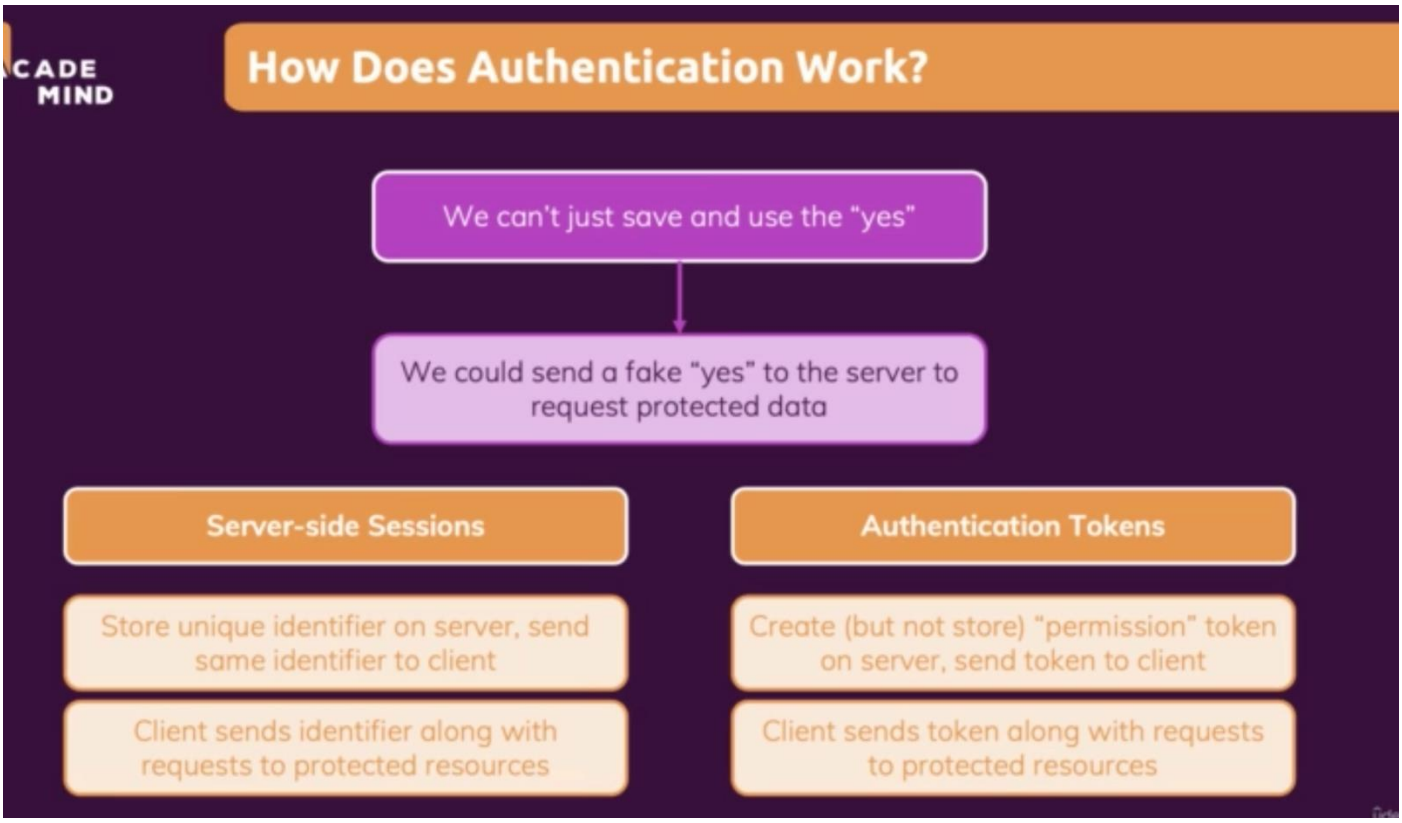
```
→  20-react-router-demo-project git:(master) firebase deploy

=== 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.firebase.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





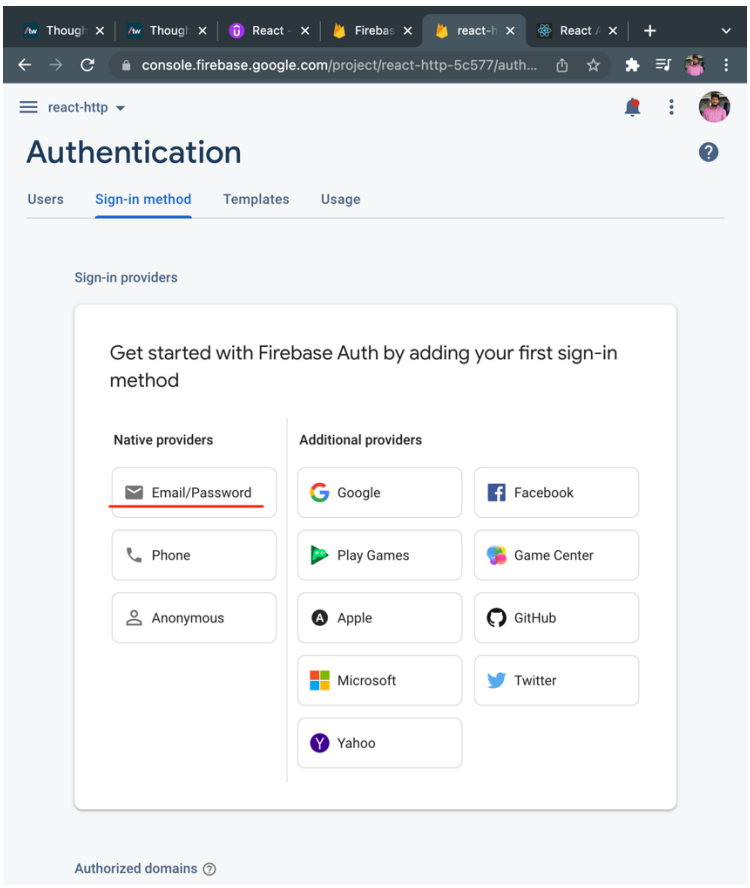

Getting Permission
Request
(with user credentials)
Client (Browser)
Server
Response
(yes / no)
Is that enough?
A "yes" alone is not enough to then access protected resources (API endpoints)


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

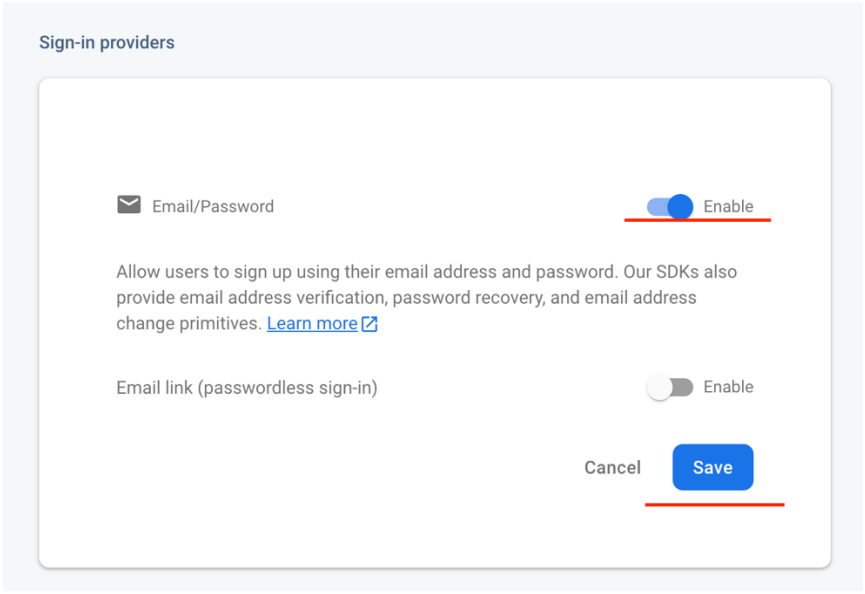
Adding User Signup

https://console.firebase.google.com/project/react-http-5c577/authentication/providers

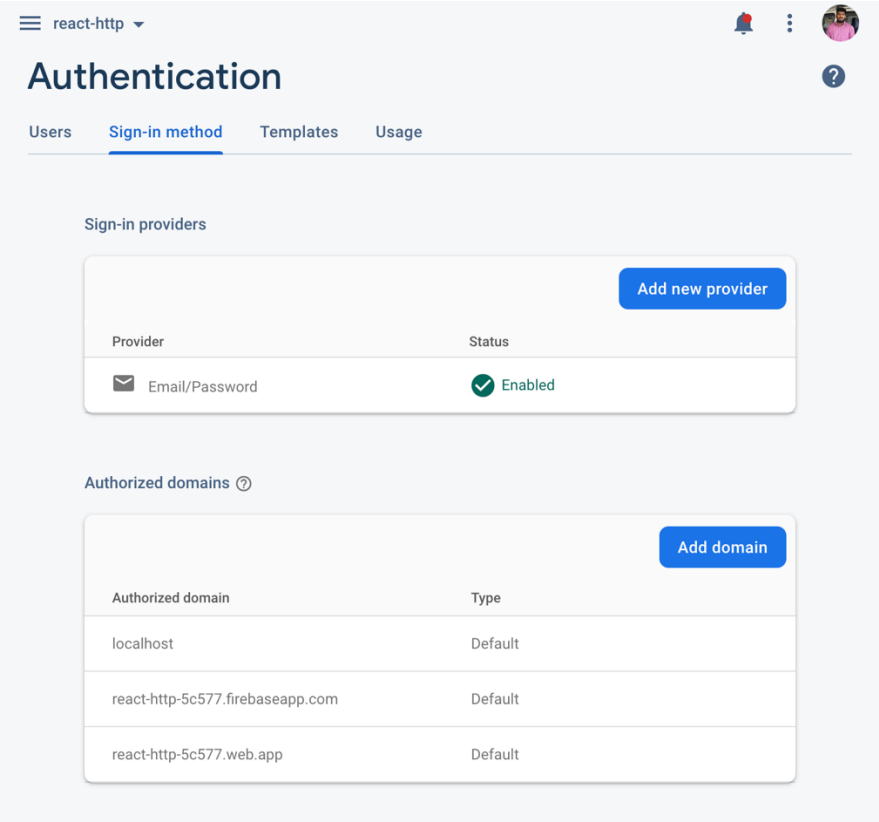
Choosing an authentication from firebase project.



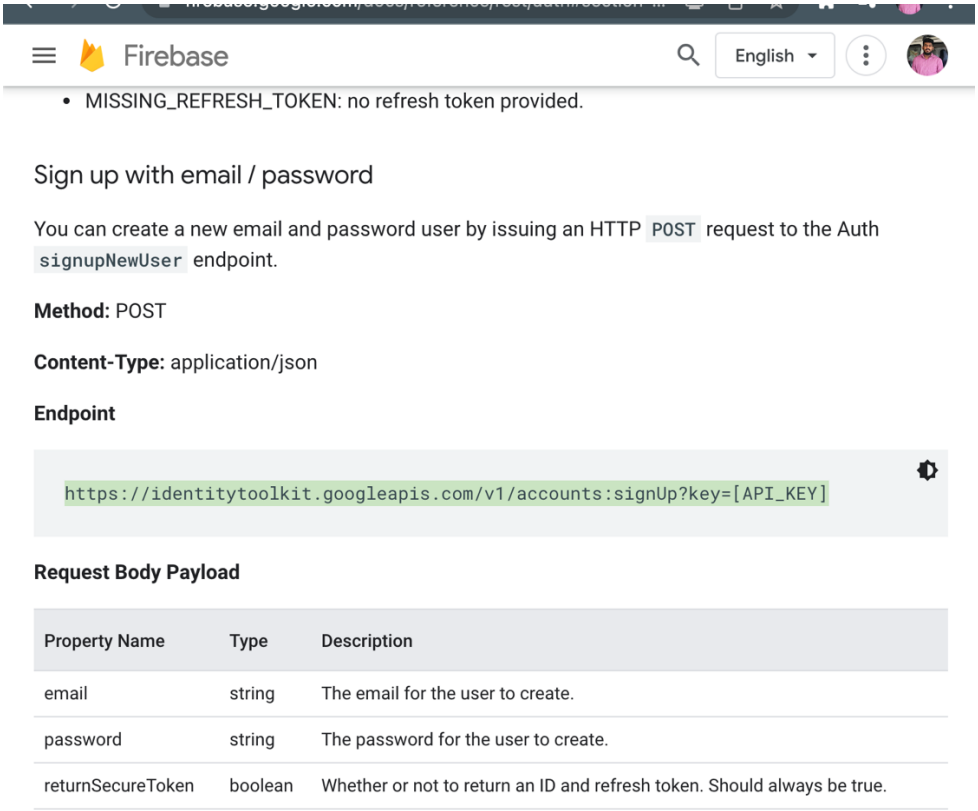
Saving the authentication type.



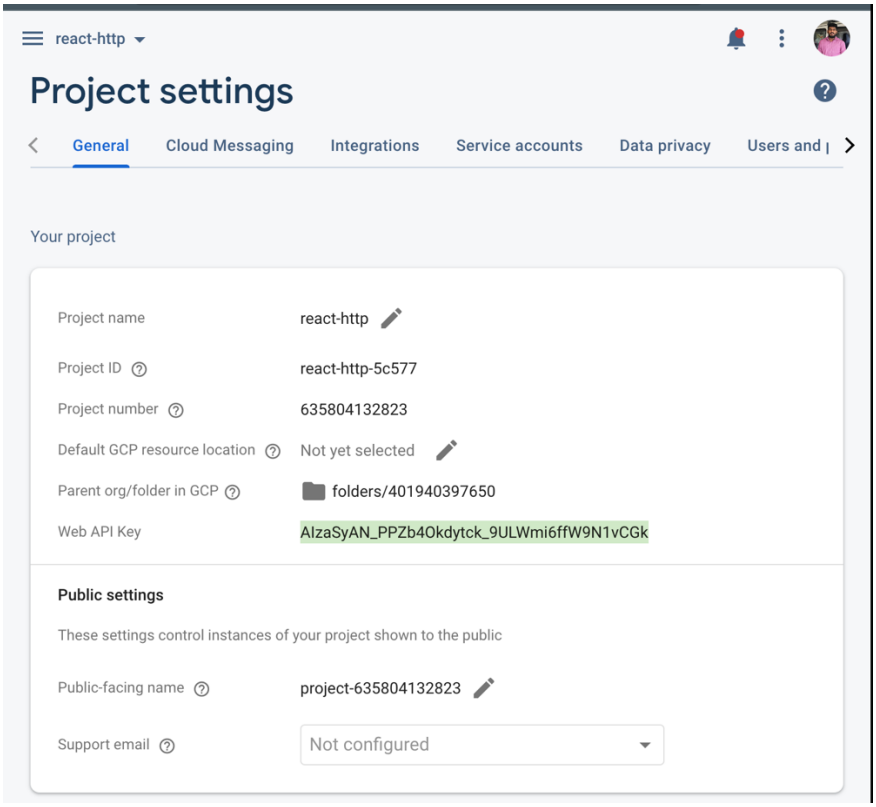
Sign-in provider.



Firebase Auth Rest API



Web API Key from firebase project setting for sign up



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 (isLogin){

  } else {
    fetch(
      "https://identitytoolkit.googleapis.com/v1/accounts:signUp?key=AIzaSyAN_PPZb4Okdytck_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);
        });
      }
    });
  }
};
```

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

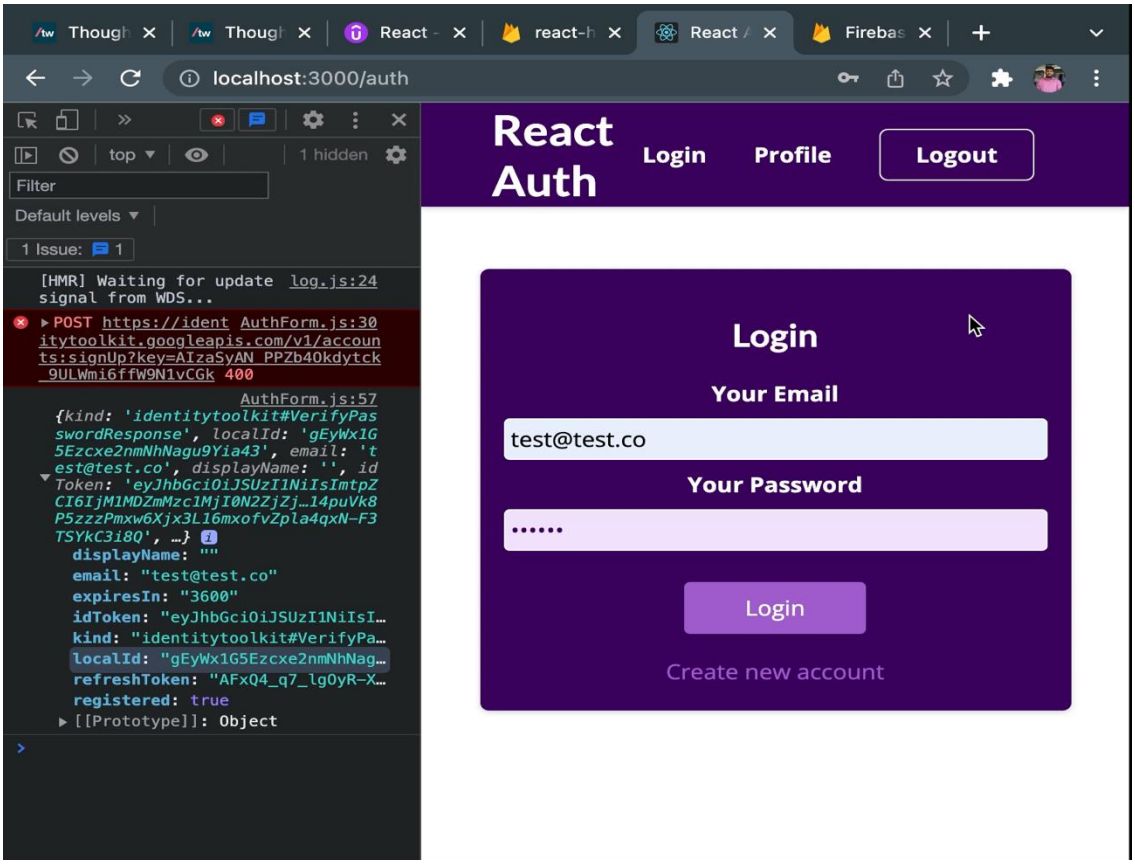
Content-Type: application/json

Endpoint

```
https://identitytoolkit.googleapis.com/v1/accounts:signInWithPassword?key=[API_KEY
```

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.



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 (isLogin){
    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

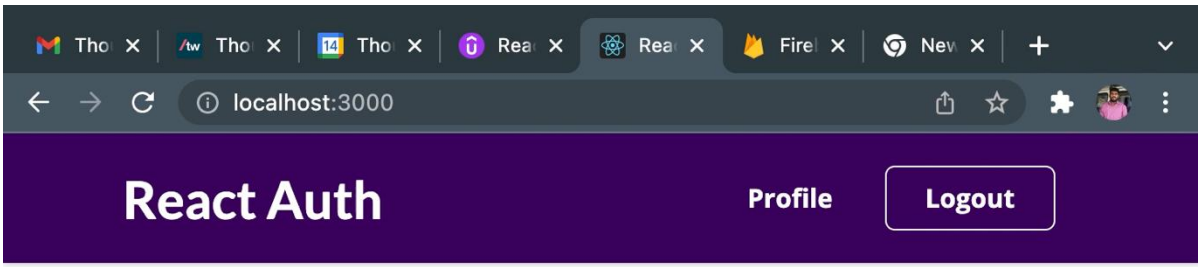
```
Complexity is 5 Everything is cool!
export const AuthContextProvider = (props) => {
    const [token, setToken] = useState(null);
    const userIsLoggedIn = !!token;

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

    }

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

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

    return (
```

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

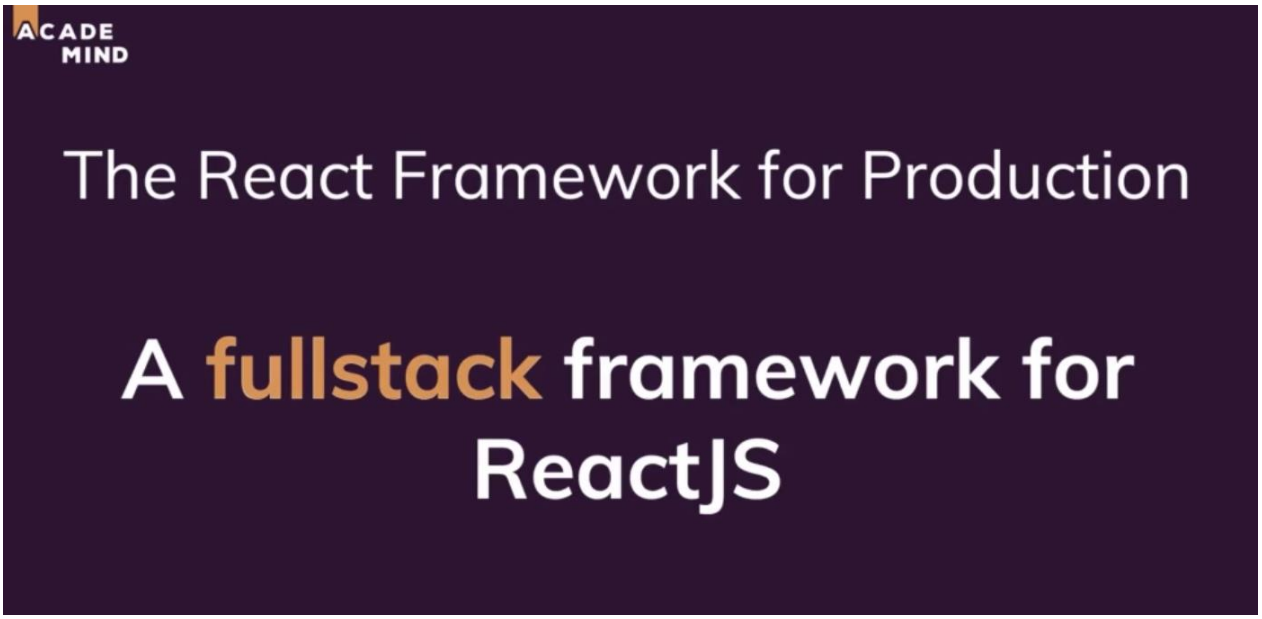
```
const logoutHandler = () => {
    setToken(null);
    localStorage.removeItem("login_token");

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

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

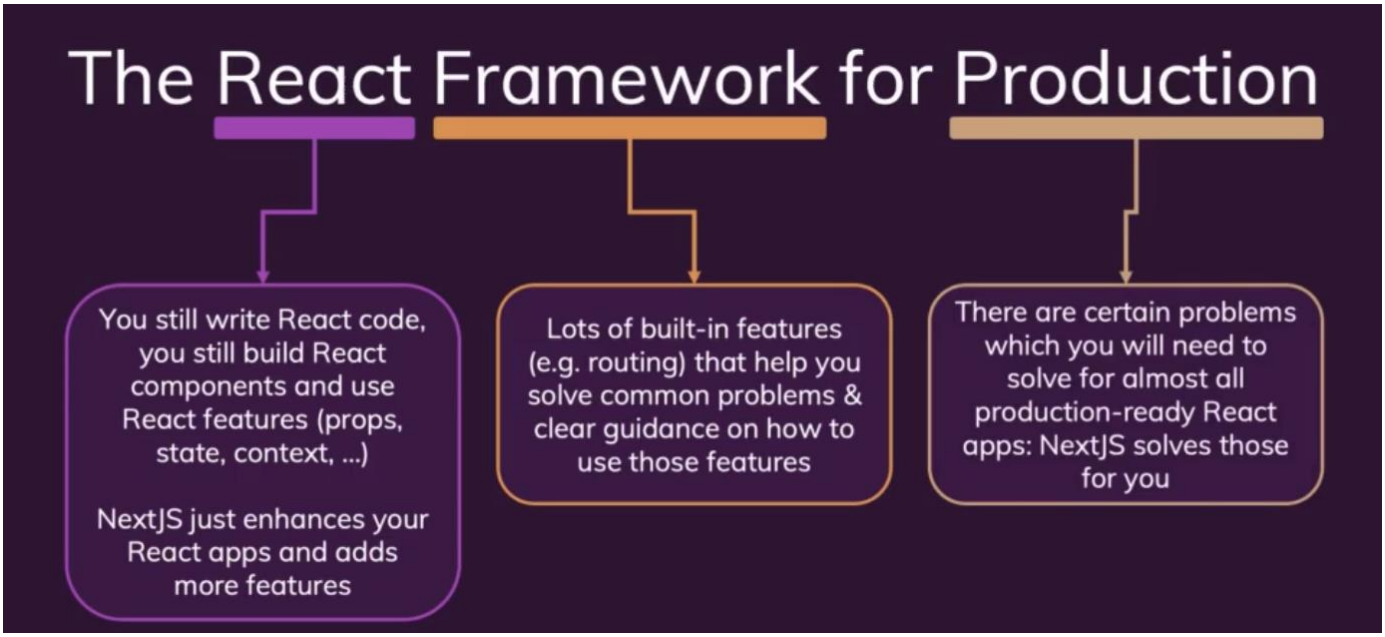


Welcome on Board!

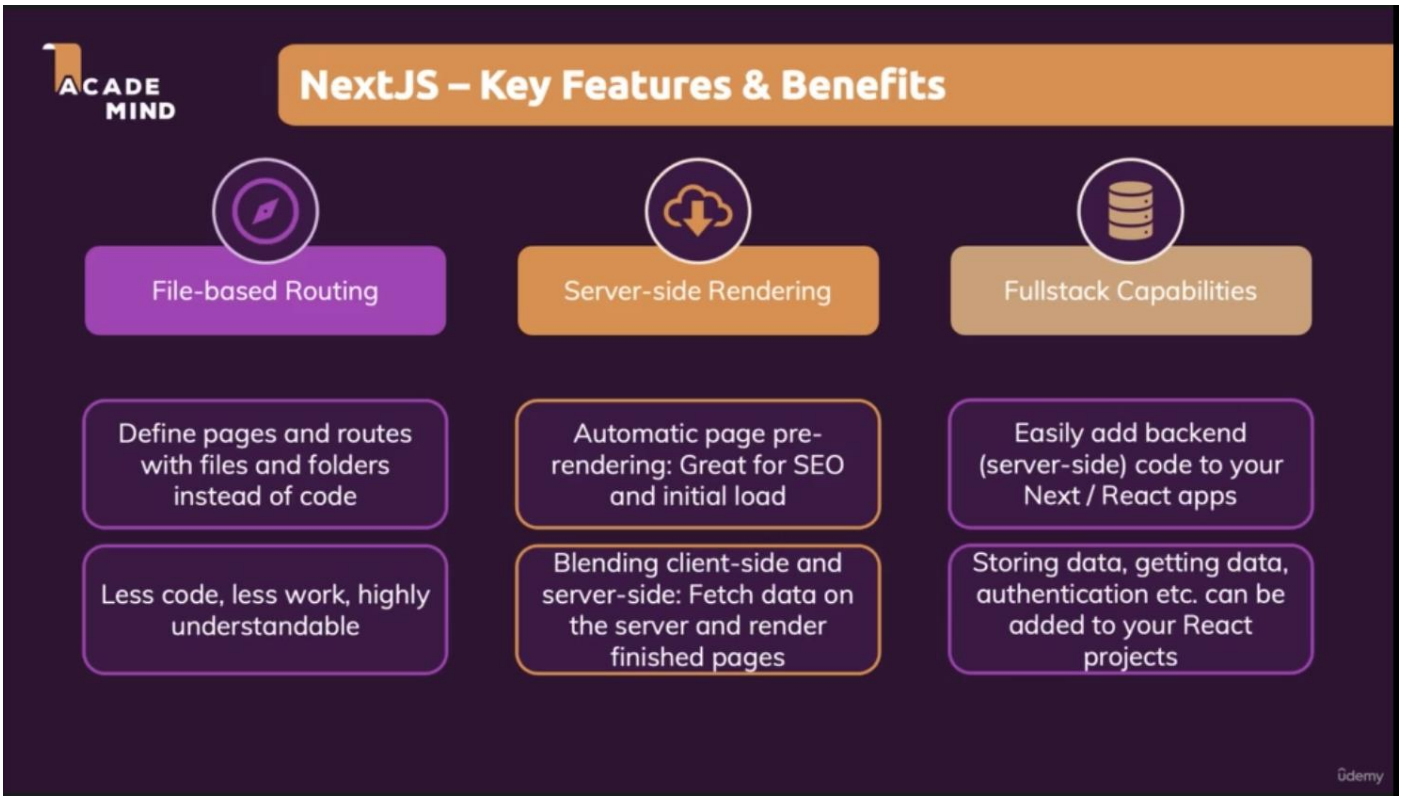
Next JS

Why Next JS?



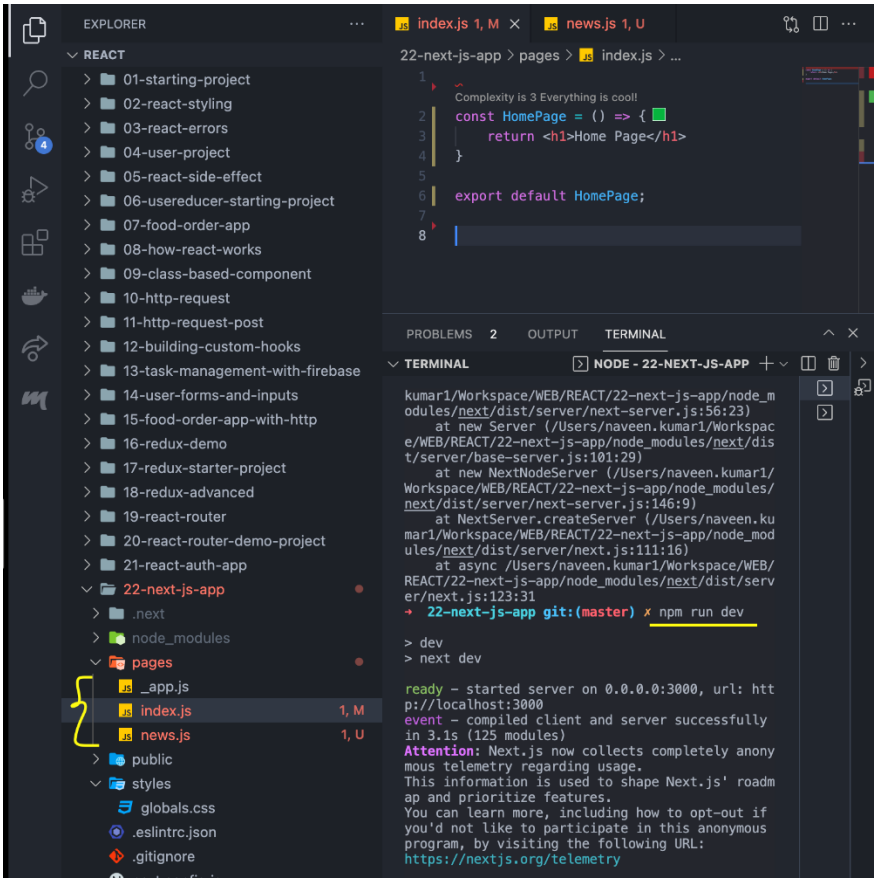


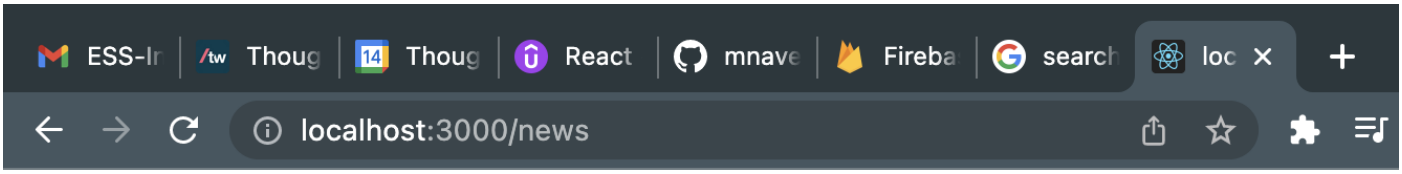
Next JS Key Feature and Benefits



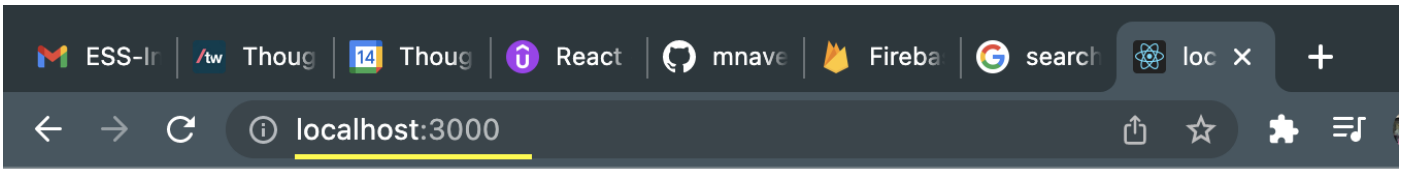
Create Next JS Project

npx create-next-app



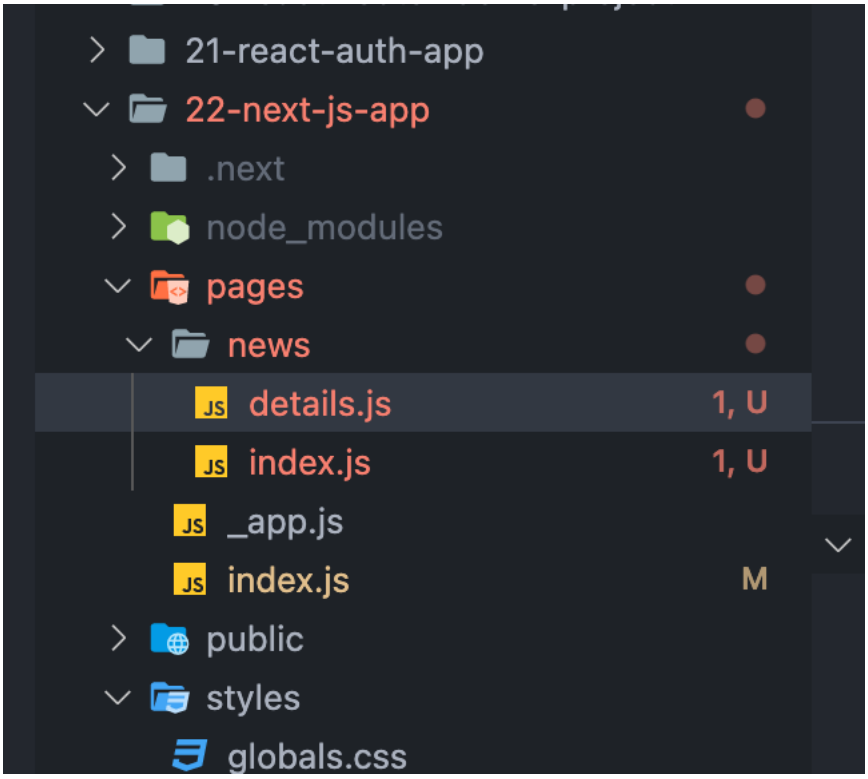


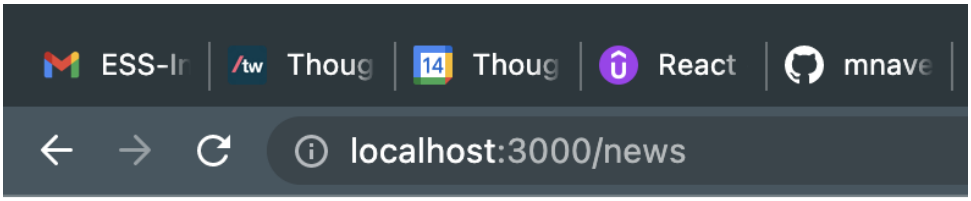
News Page



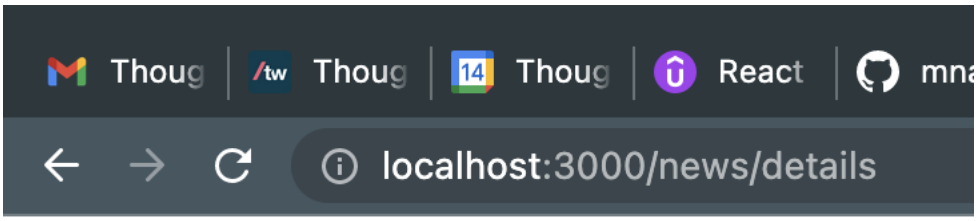
Home Page

Nested Pages





News Page

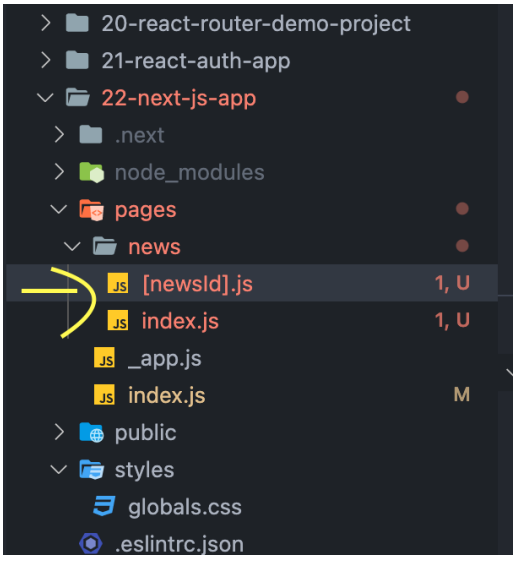


Detail Page

Dynamic Page

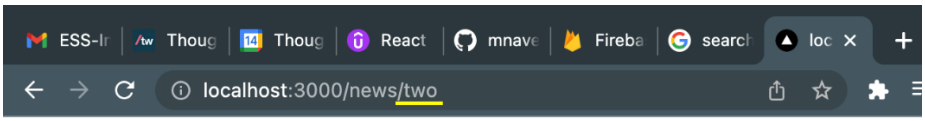
Create the file name with square bracket. Here, newsId is a placeholder. Now this placeholder accept any values.

domainname/news/1
domainname/news/2



Detail Page

Extracting a Dynamic Parameter Value

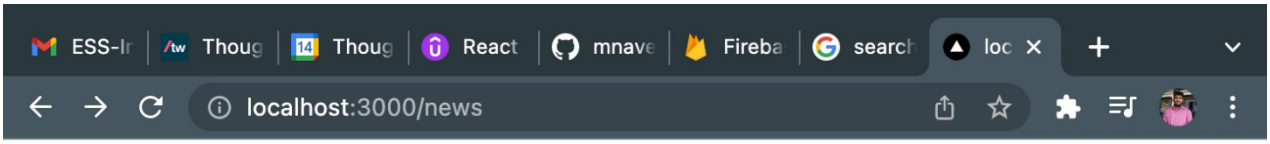


Detail Page two

```
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

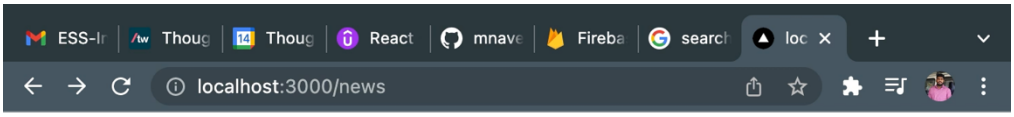
```jsx
import { Fragment } from "react";

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.

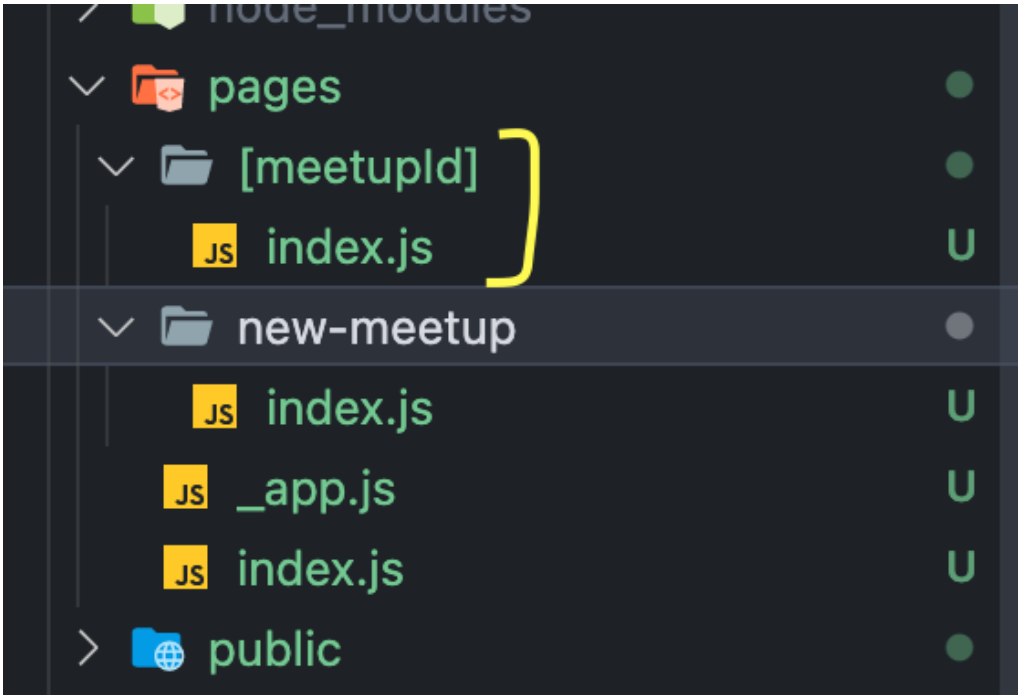


News Page

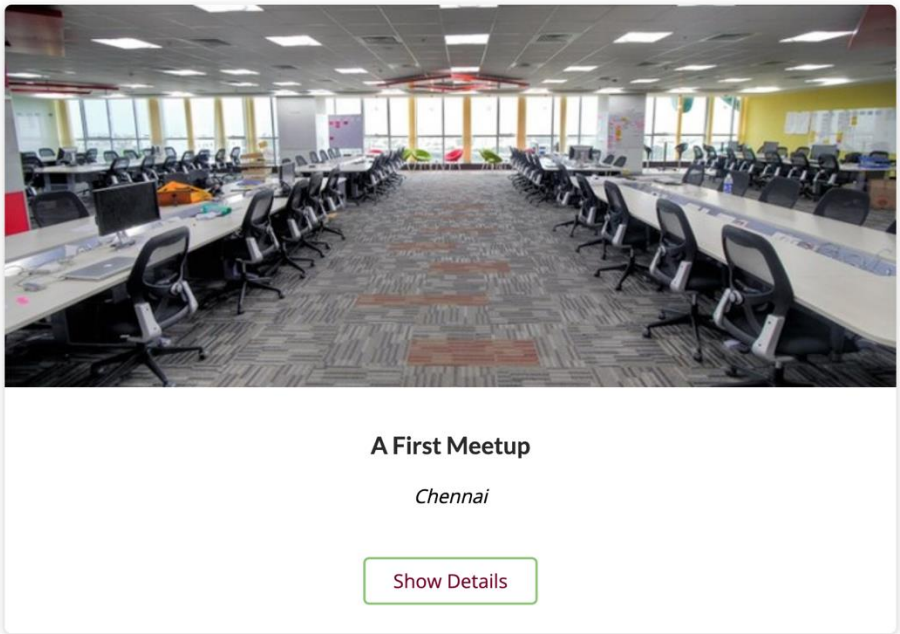
- Artical 1
- Artical 2
- Artical 3
- Artical 4

```jsx
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



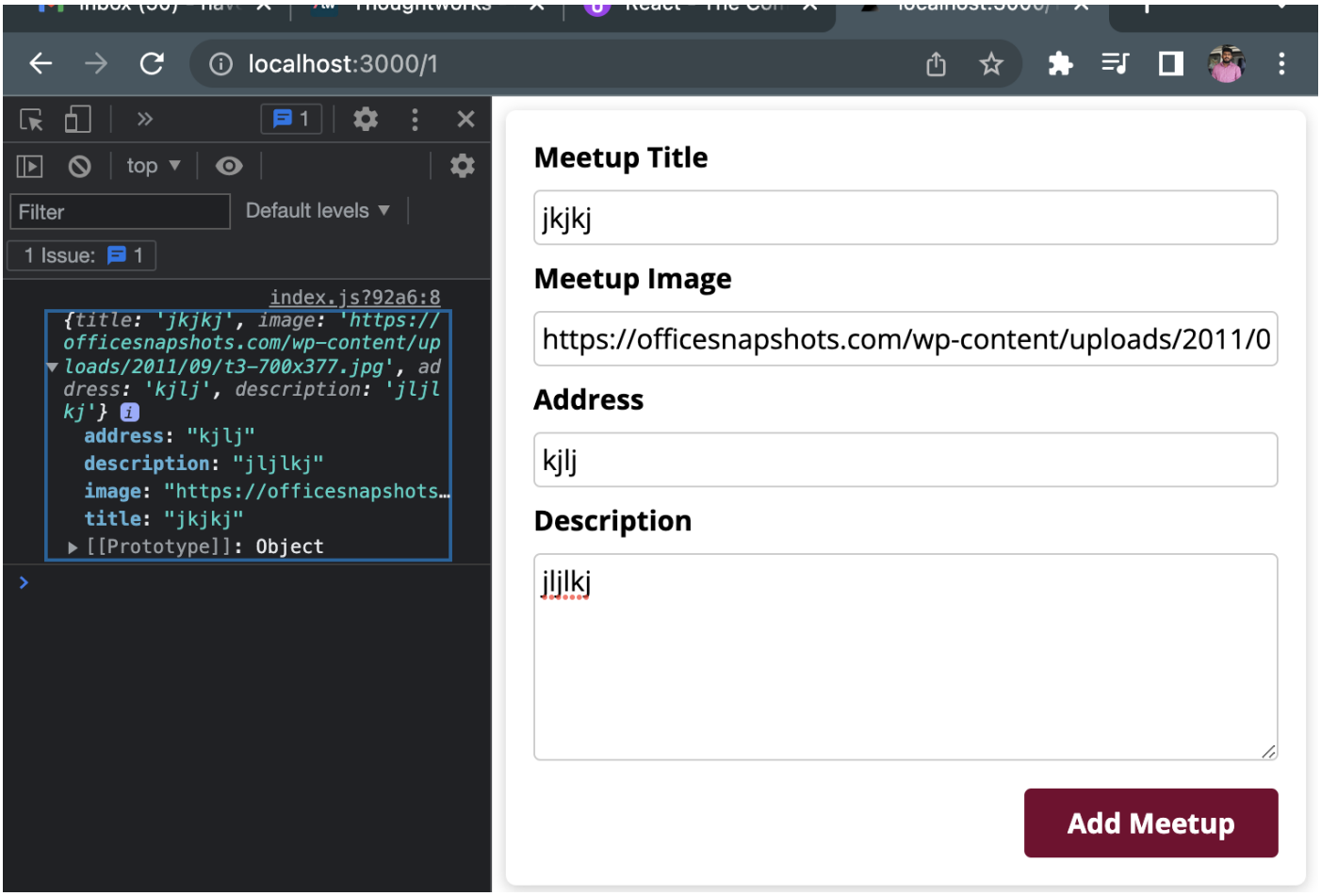
```
import MeetupList from '../components/meetups/Meetup

const DUMMY_MEETUPS = [
    {
        id: 'm1',
        title: 'A First Meetup',
        image: 'https://officesnapshots.com/wp-conte
        address: 'Chennai',
        description: 'This is a first meetup'
    },
    {
        id: 'm2',
        title: 'A Second Meetup',
        image: 'https://officesnapshots.com/wp-conte
        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

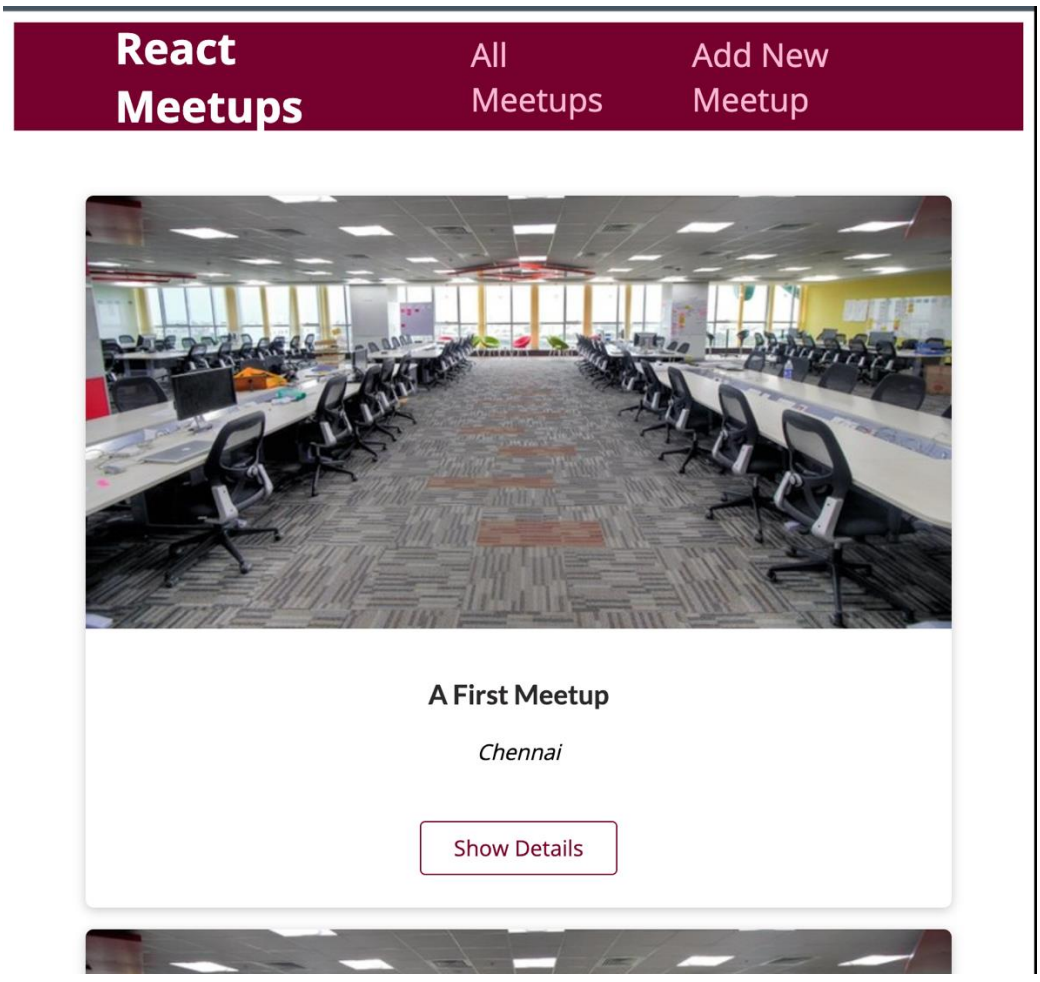
Show Details




```
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;
```

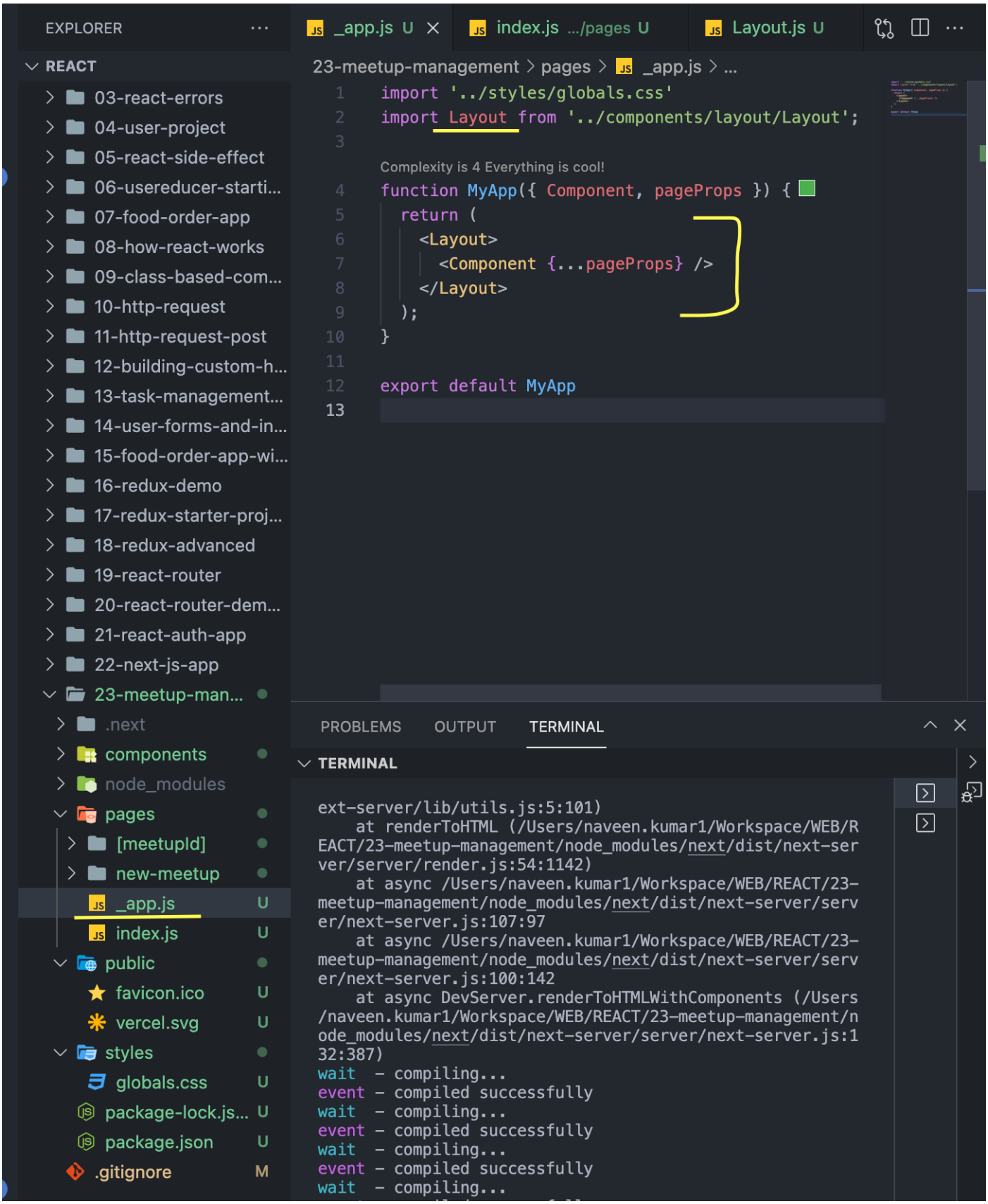
Adding a Navigations

```
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</d
      <nav>
        <ul>
          <li>
            <Link href='/'>All Meetups</Link>
          </li>
          <li>
            <Link href='/new-meetup'>Add New Meetup<
          </li>
        </ul>
      </nav>
    </header>
  );
}

export default MainNavigation;
```

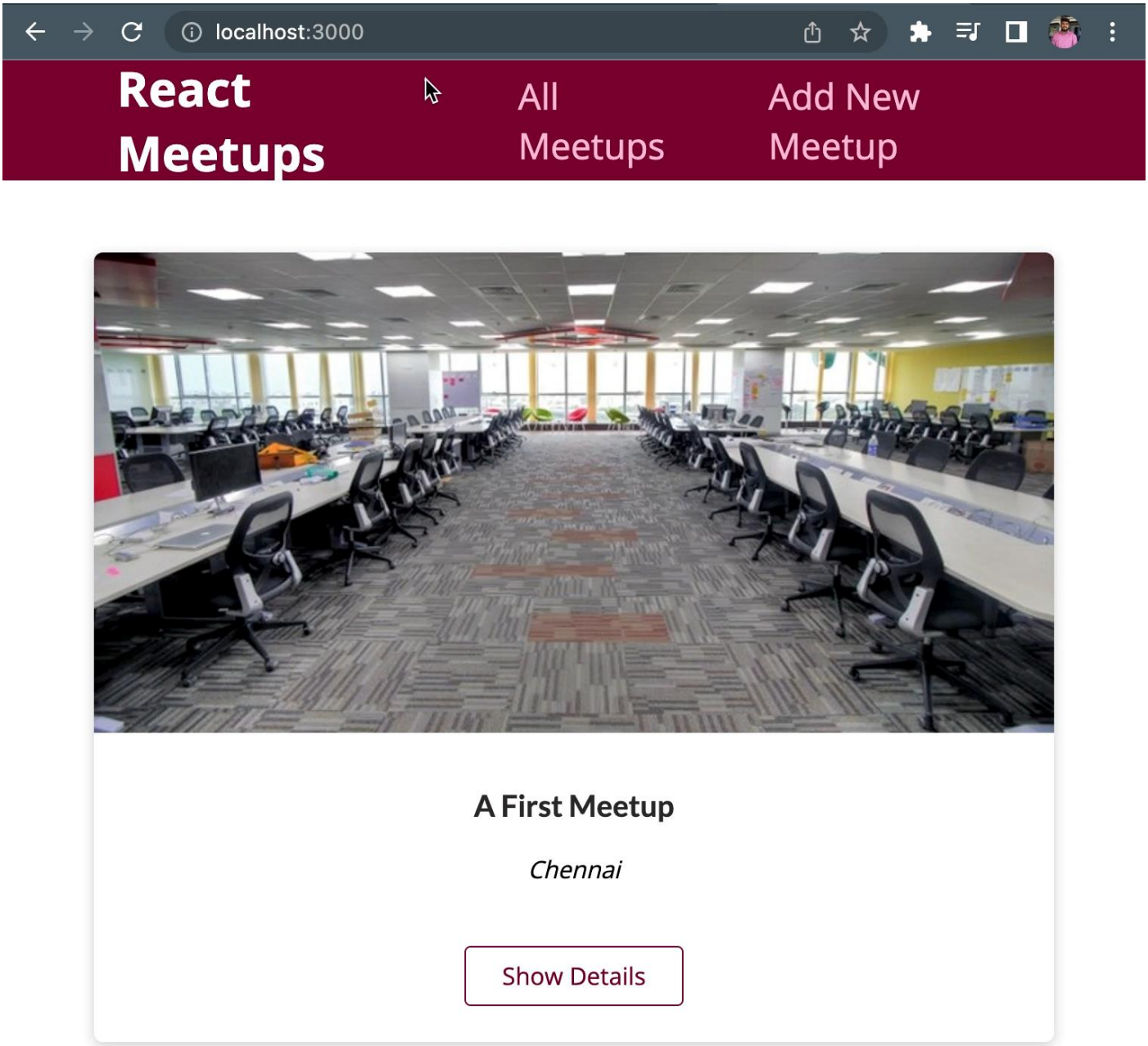


Adding a navigation bar to entire pages

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

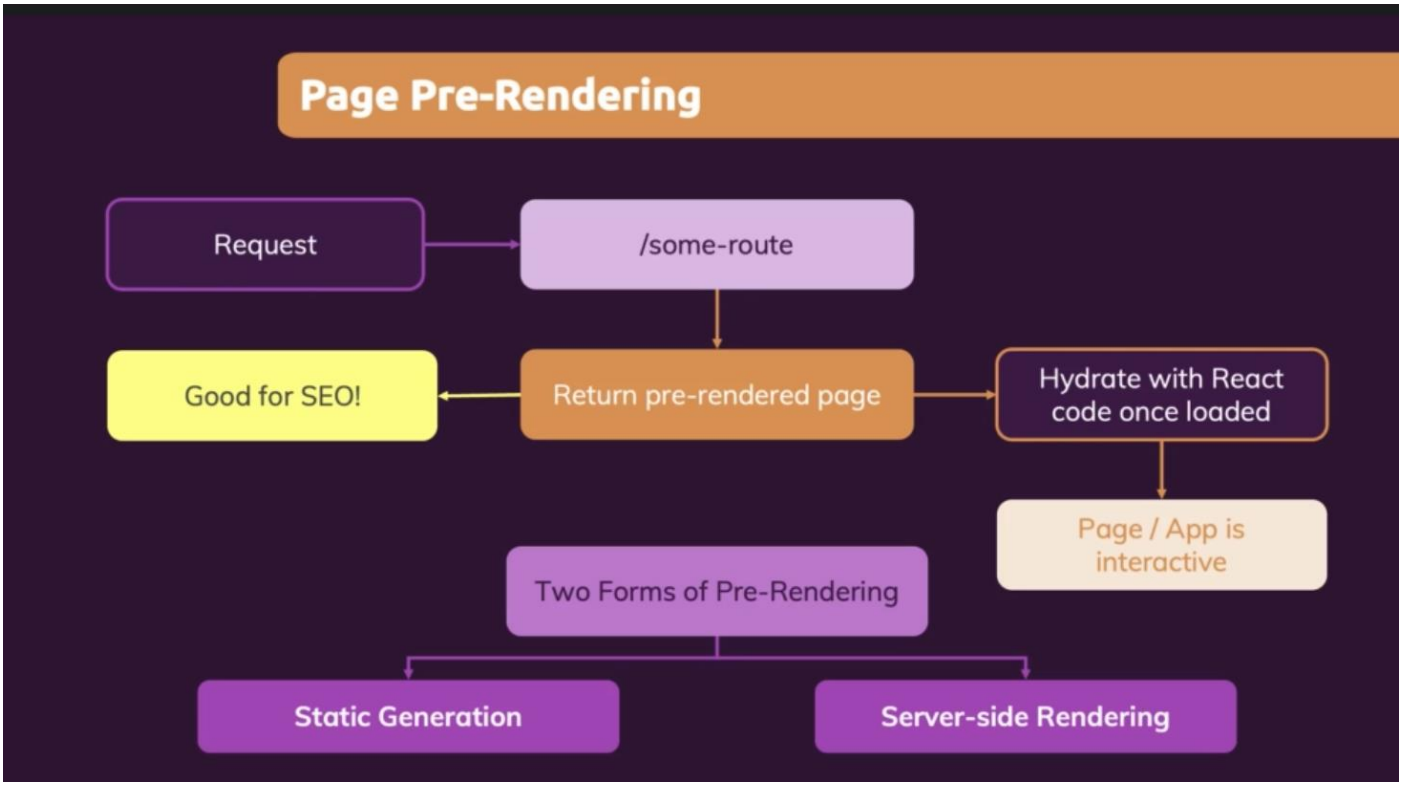


Meetup App - Implementation so far



Next JS helps fetch the data for pre-rendering.

Data Fetching for static pages



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 the before the component function. It is used to provide the data which component function requires. Also, this special function will be executed when 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                                                                    Size     First Load JS
┌ ● /                                                                   691 B          65.1 kB
├   └ css/a328f70f1500a47f59a4.css                                      409 B
├   /_app                                                               0 B            64.4 kB
├ ○ /[meetupId]                                                         498 B          64.9 kB
├   └ css/f13f2474f39af1e03147.css                                      81 B
├ ○ /404                                                                3.46 kB        67.8 kB
└ ○ /new-meetup                                                         753 B          65.1 kB
    └ css/60f9e9f292a28db77853.css                                      360 B
+ First Load JS shared by all                                          64.4 kB
  ├ chunks/94268f14e8a7805d20be5bd304f076c04a2cb3ee.49b41a.js           13.1 kB
  ├ chunks/framework.d886aa.js                                         41.8 kB
  ├ chunks/main.cfc182.js                                              6.63 kB
  ├ chunks/pages/_app.c98eca.js                                        2.13 kB
  ├ chunks/webpack.50bee0.js                                           751 B
  └ css/8b9a233bedcad23e246e.css                                       513 B

λ  (Server)  server-side renders at runtime (uses getInitialProps or getServerSideProp
s)
○  (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.

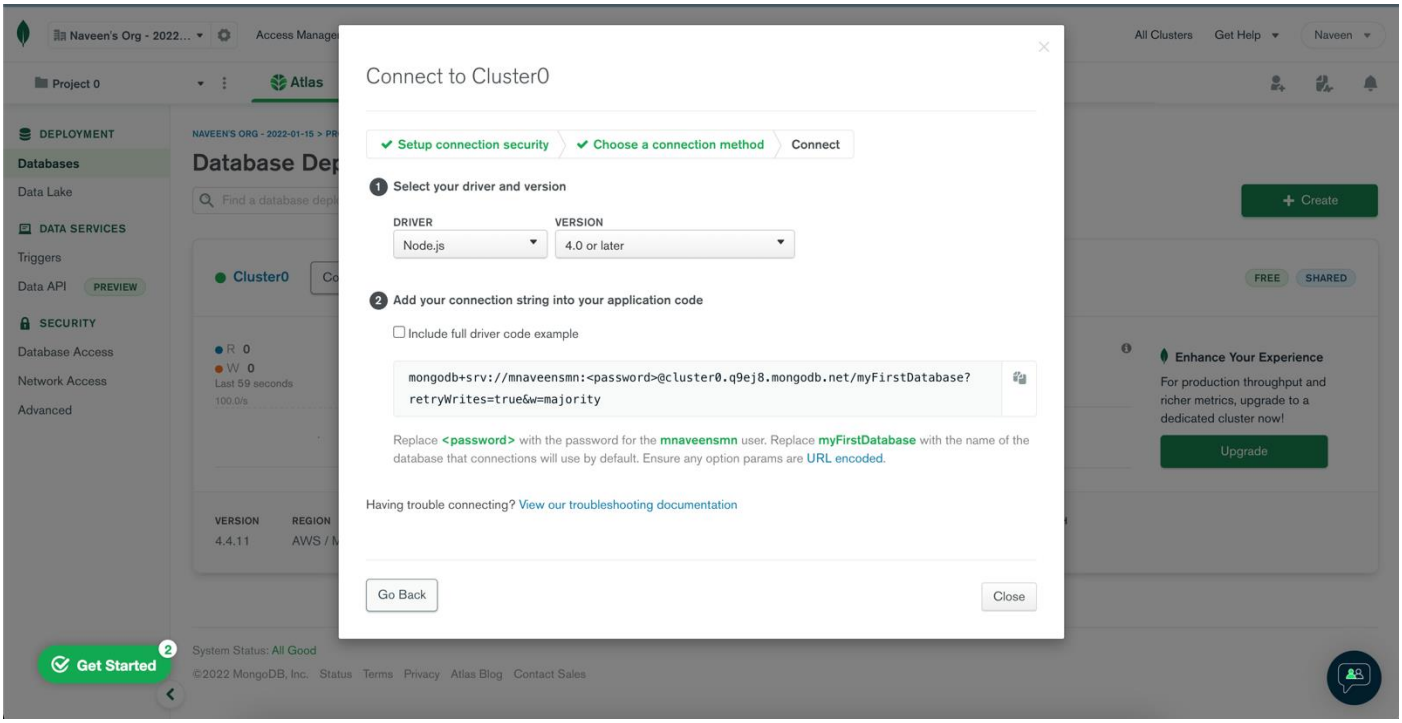
getStaticPaths()

```
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

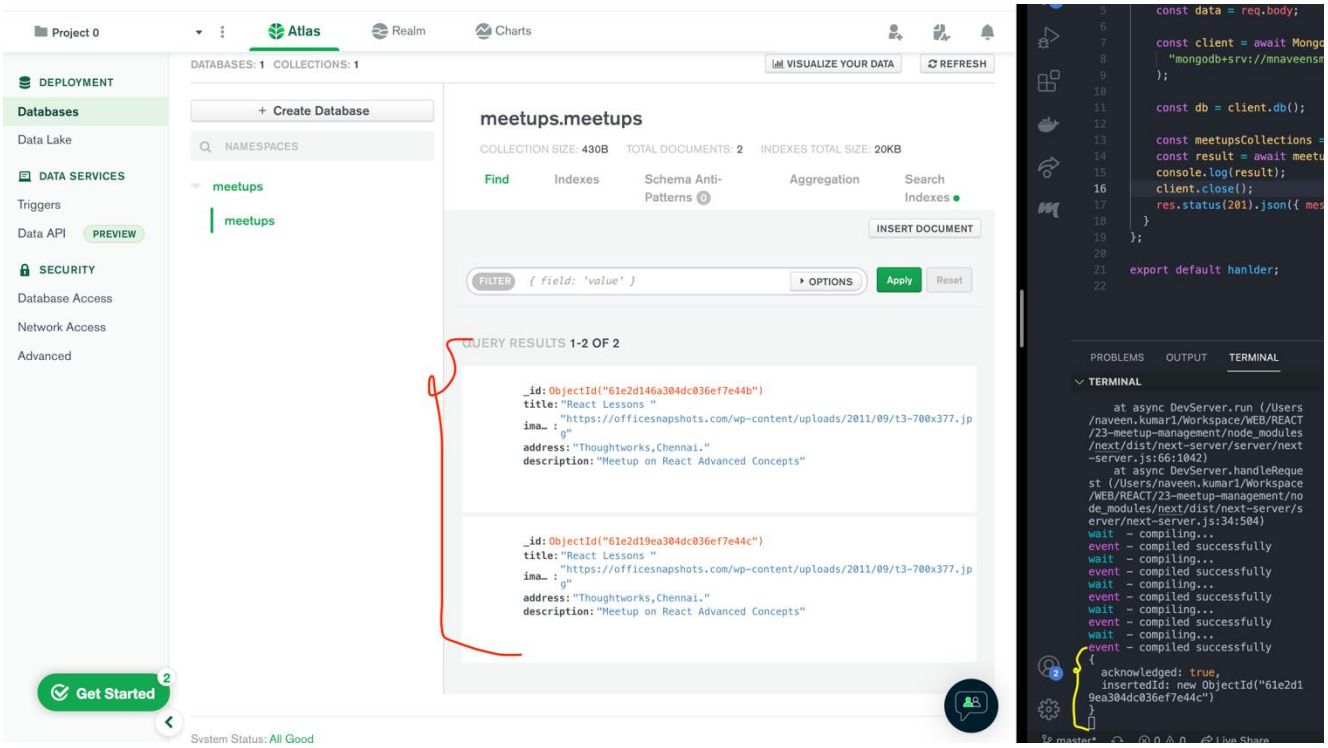


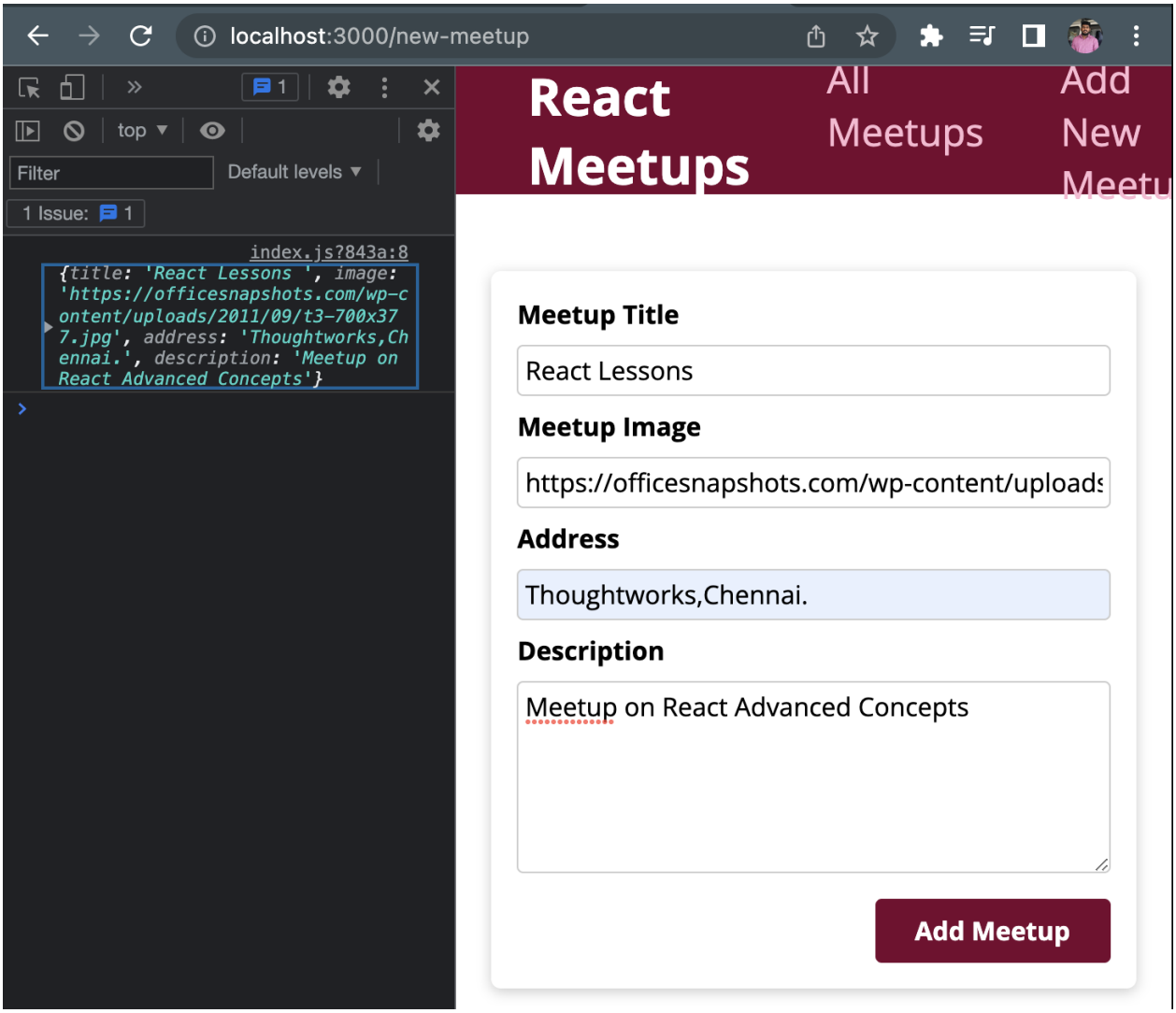
```
^C
➜  23-meetup-management git:(master) ✗ npm install mangodb

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:YO5X9Mfva6hiZ0FV@cluster0.q9ej8.mongodb.net/meet
    );

    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;
```


Project 0
Atlas
Realm
Charts
DEPLOYMENT
Databases
Data Lake
DATA SERVICES
Triggers
Data API
PREVIEW
SECURITY
Database Access
Network Access
Advanced
DATABASES: 1 COLLECTIONS: 1
VISUALIZE YOUR DATA
REFRESH
+ Create Database
NAMESPACES
meetups
meetups
meetups.meetups
COLLECTION SIZE: 430B TOTAL DOCUMENTS: 2 INDEXES TOTAL SIZE: 20KB
Find
Indexes
Schema Anti-Patterns
Aggregation
Search Indexes
INSERT DOCUMENT
FILTER { field: 'value' }
OPTIONS
Apply
Reset
QUERY RESULTS 1-2 OF 2
_id: ObjectId("61e2d146a304dc036ef7e44b")
title: "React Lessons "
address: "Thoughtworks,Chennai."
description: "Meetup on React Advanced Concepts"
_id: ObjectId("61e2d19ea304dc036ef7e44c")
title: "React Lessons "
address: "Thoughtworks,Chennai."
description: "Meetup on React Advanced Concepts"
Get Started
System Status: All Good
const data = req.body;
const client = await Mongo
"mongodb+srv://mnaveensm
);
const db = client.db();
const meetupsCollections =
const result = await meetu
console.log(result);
client.close();
res.status(201).json({ mes
};
export default handler;
PROBLEMS
OUTPUT
TERMINAL
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)
at async DevServer.handleRequest (/Users/naveen.kumar1/Workspace/WEB/REACT/23-meetup-management/node_modules/next/dist/next-server/server/next-server.js:34:504)
wait - compiling...
event - compiled successfully
wait - compiling...
event - compiled successfully
wait - compiling...
event - compiled successfully
wait - compiling...
event - compiled successfully
wait - compiling...
event - compiled successfully
{
acknowledged: true,
insertedId: new ObjectId("61e2d19ea304dc036ef7e44c")
}

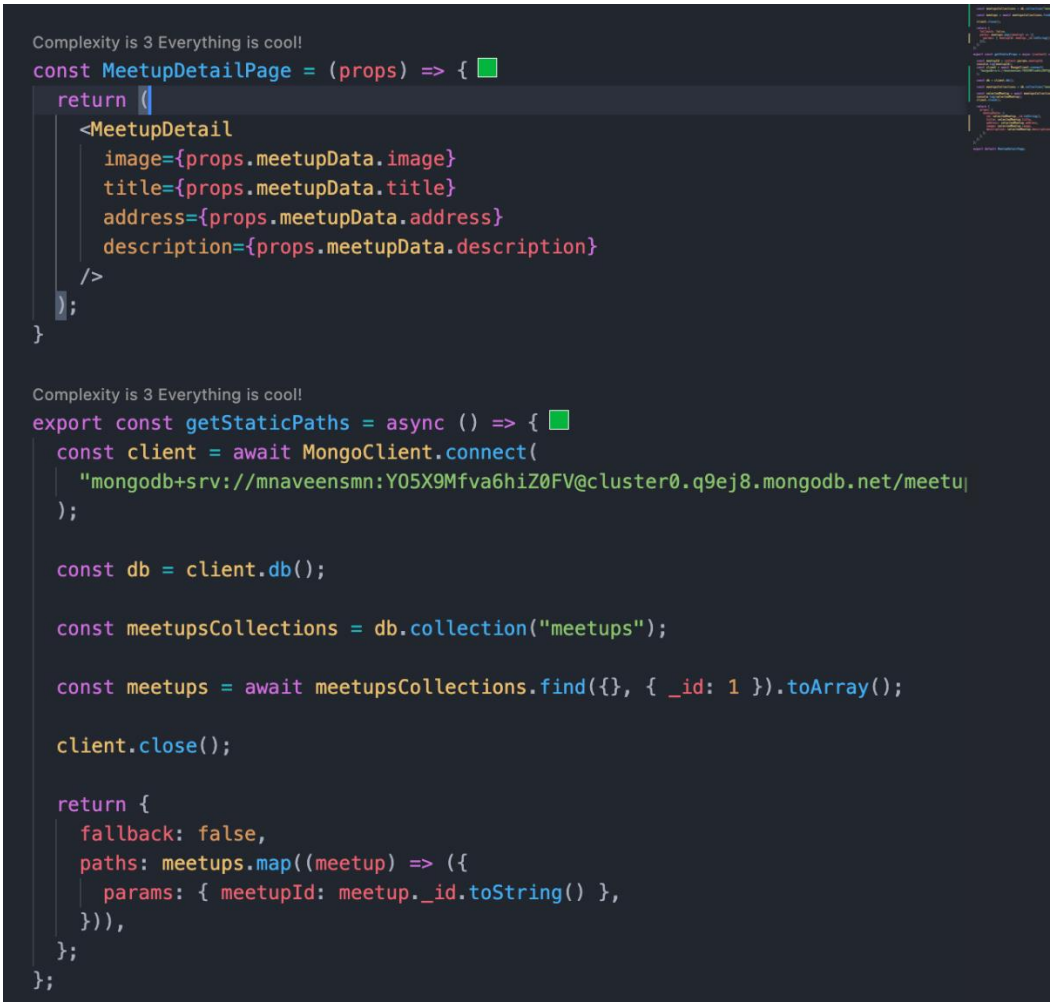

localhost:3000/new-meetup
top
Filter
Default levels
1 Issue: 1
index.js?843a:8
{title: 'React Lessons ', image: 'https://officesnapshots.com/wp-content/uploads/2011/09/t3-700x377.jpg', address: 'Thoughtworks,Chennai.', description: 'Meetup on React Advanced Concepts'}
React Meetups
All Meetups
Add New Meetu
Meetup Title
React Lessons
Meetup Image
https://officesnapshots.com/wp-content/uploads
Address
Thoughtworks,Chennai.
Description
Meetup on React Advanced Concepts
Add Meetup

Loading Data from MongoDB

```
Complexity is 3 Everything is cool!
export const getStaticProps = async () => {
  const client = await MongoClient.connect(
    "mongodb+srv://mnaveensmn:YO5X9Mfva6hiZ0FV@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,
  };
};
```

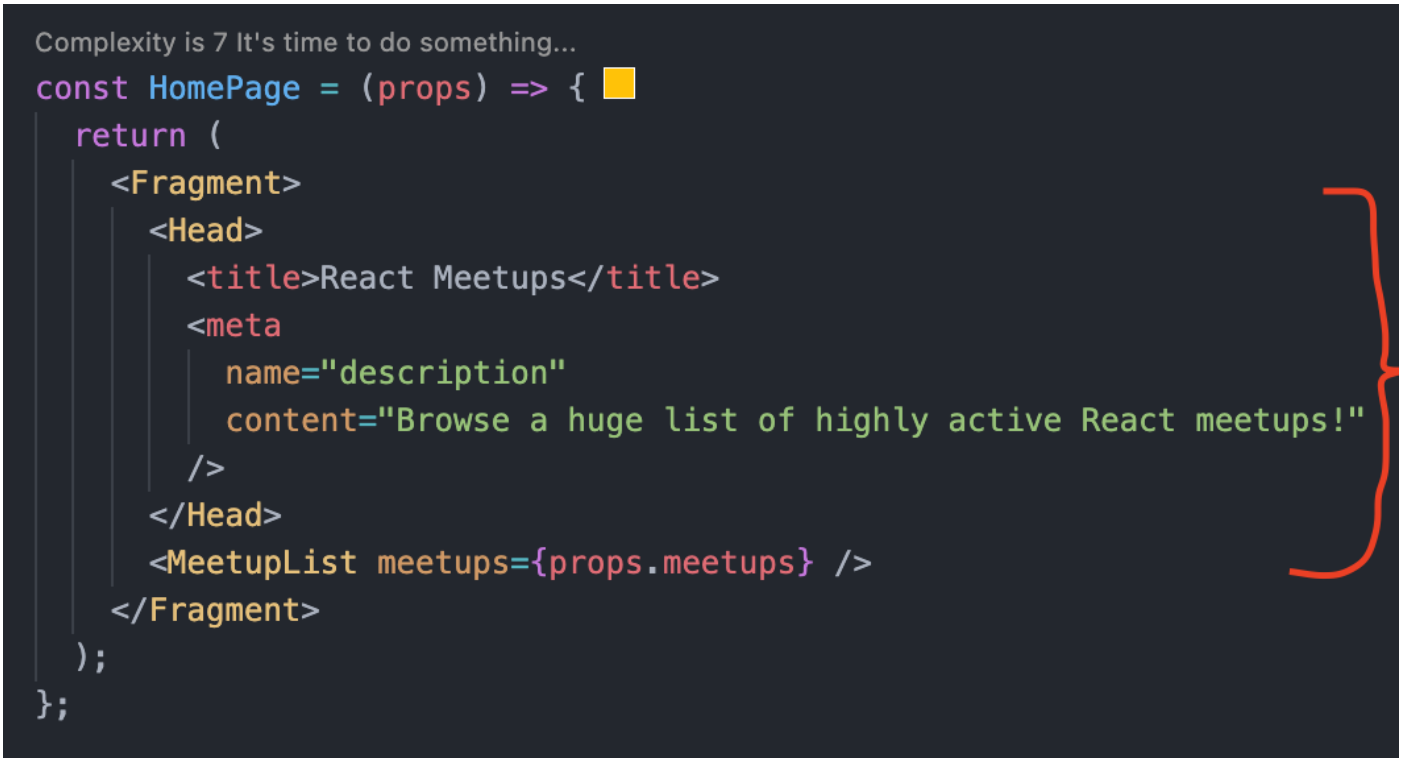
Loading a Dynamic Data in MeetupDetails page

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

  const meetupId = context.params.meetupId;
  console.log(meetupId);
  const client = await MongoClient.connect(
    "mongodb+srv://mnaveensmn:YO5X9Mfva6hiZ0FV@cluster0.q9ej8.mongodb.net/meetu
  );

  const db = client.db();

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

  const selectedMeetup = await meetupsCollections.findOne({ _id: ObjectId(meetu
  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:YO5X9Mfva6hiZ0FV@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() },
    })),
  };
};
```

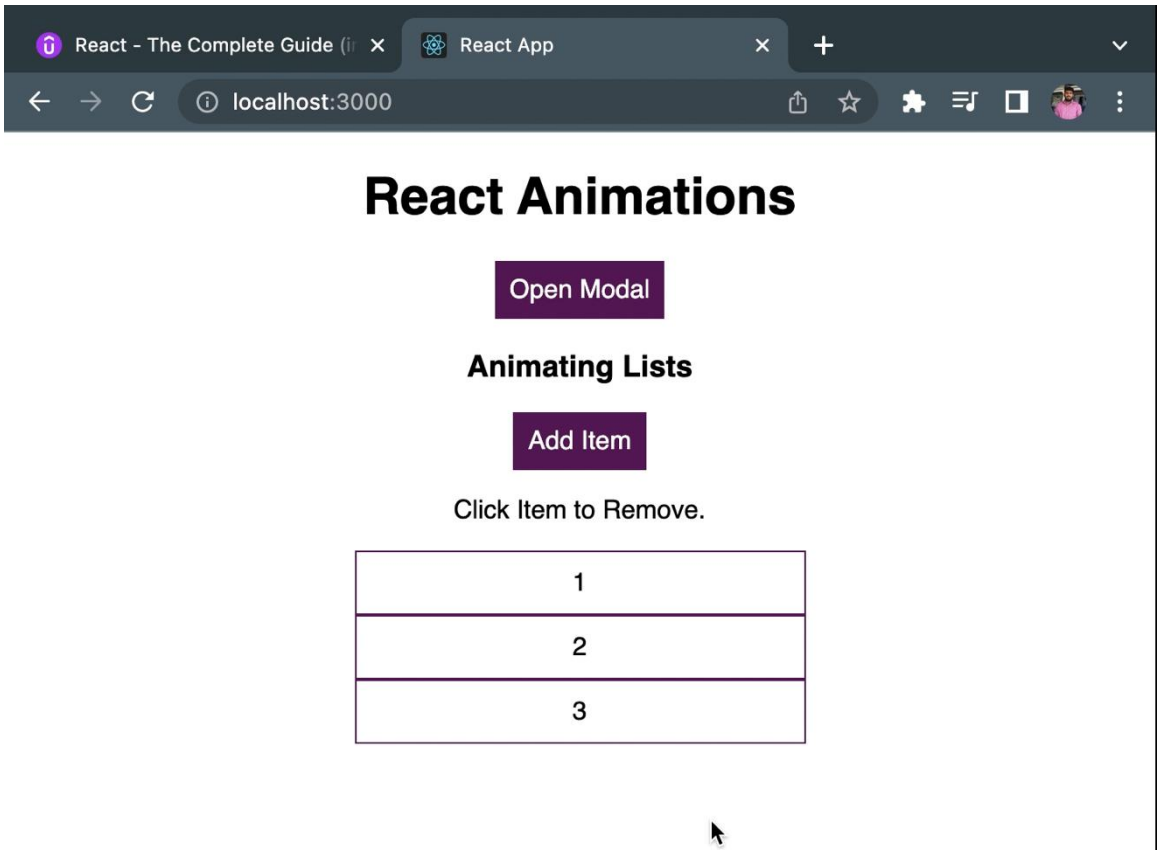
Adding a Meta Data to Page before Deployment



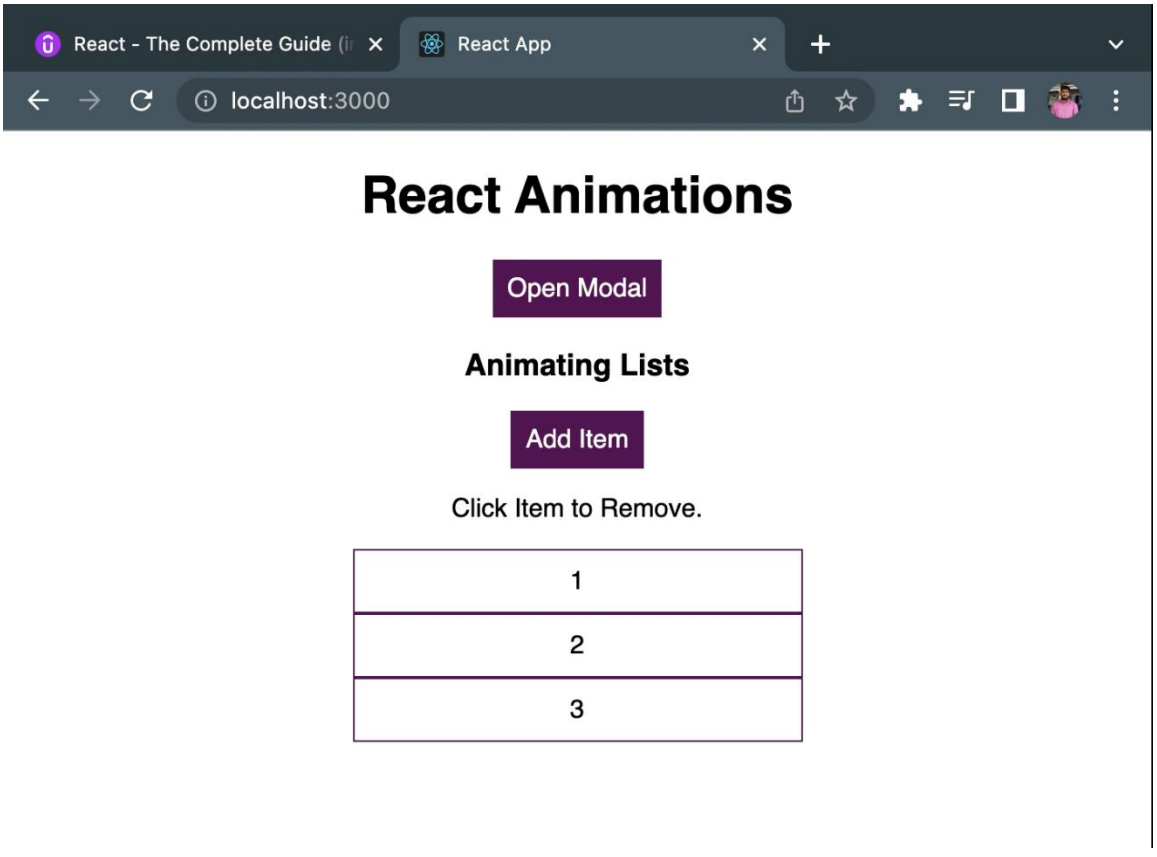
```
Complexity is 7 It's time to do something...
const HomePage = (props) => {
  return (
    <Fragment>
      <Head>
        <title>React Meetups</title>
        <meta
          name="description"
          content="Browse a huge list of highly active React meetups!"
        />
      </Head>
      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

Animation

Preparing the project for adding animation



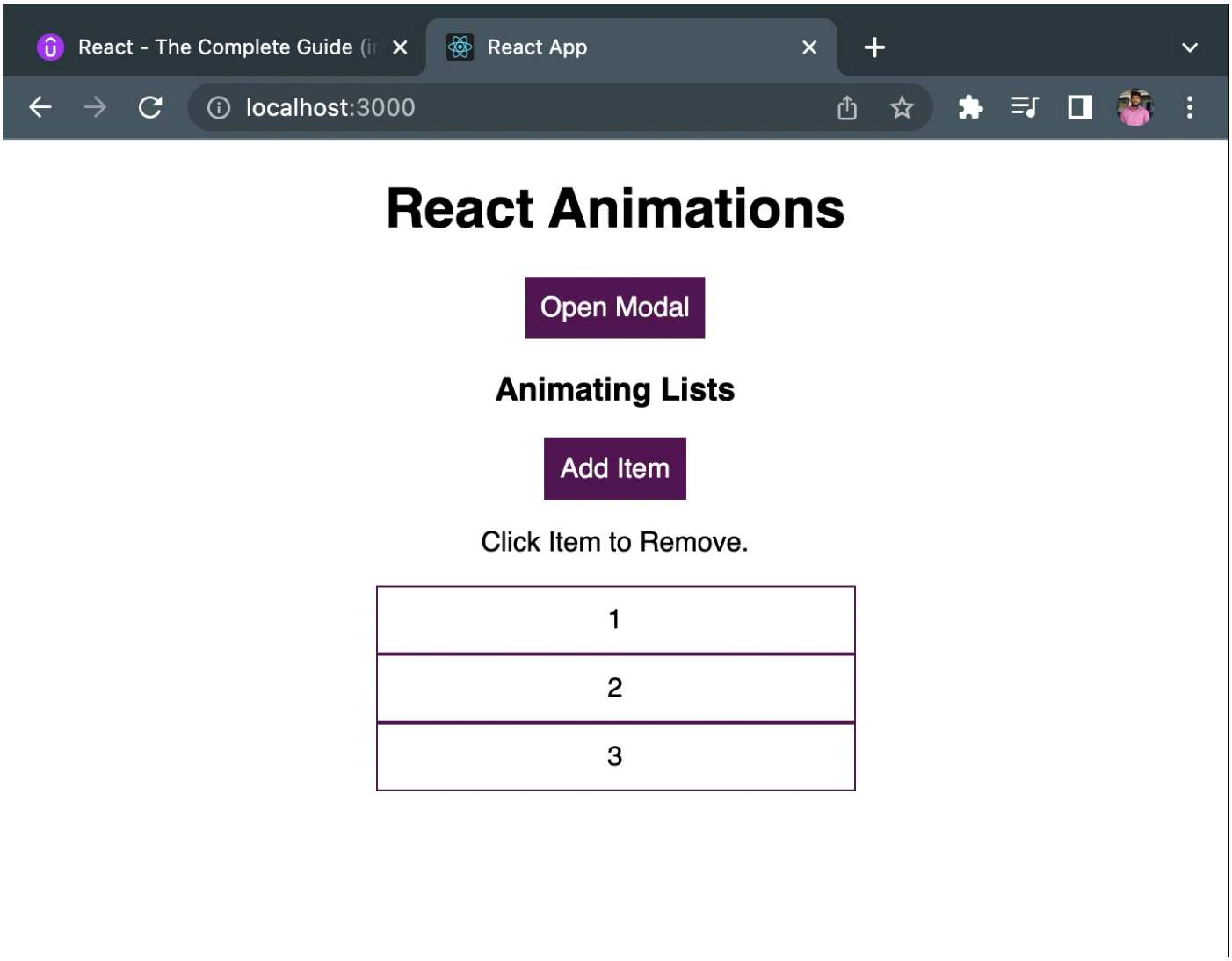
Animating the model pop up



```css
.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%);
}
```

```jsx
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
.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%);
  }

}
```

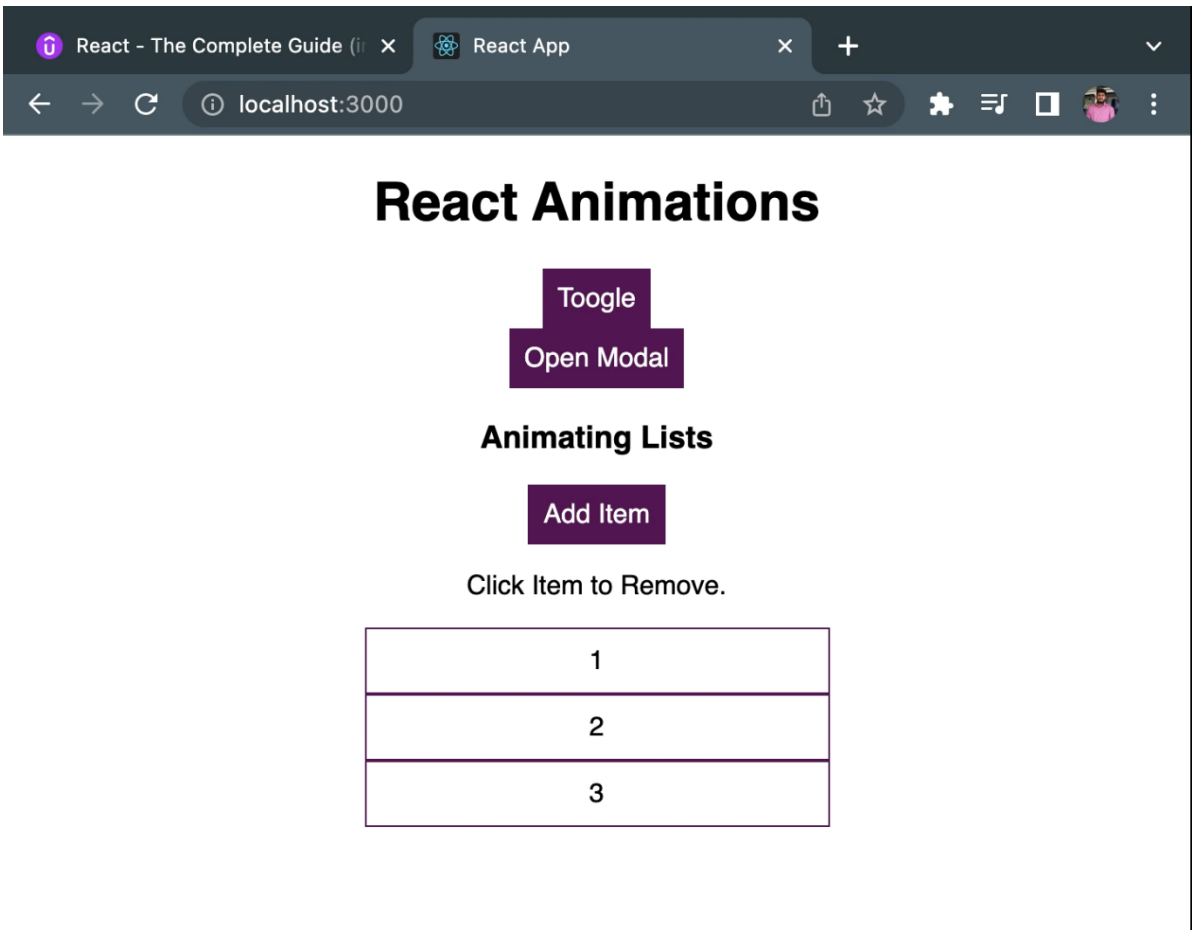


Using ReactTransition group

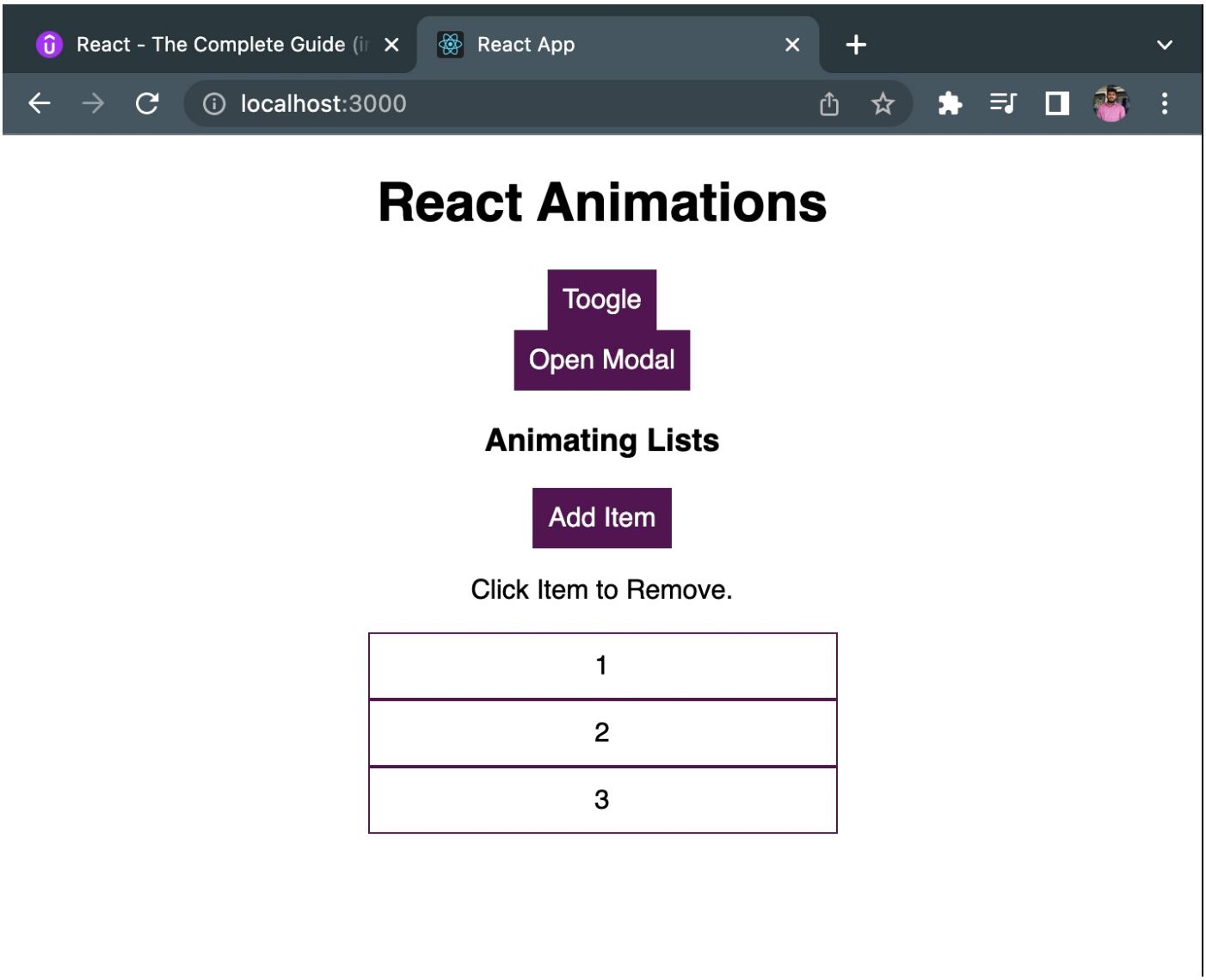
```
→  24-react-animations git:(master) ✗ npm install react-transition-group --save
npm WARN EBADENGINE Unsupported engine {
npm WARN EBADENGINE   package: 'got@5.7.1',
npm WARN EBADENGINE   required: { node: '>=0.10.0 <7' },
npm WARN EBADENGINE   current: { node: 'v14.17.5', npm: '7.24.0' }
npm WARN 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";
```

```
<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>
```



```
<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 />
```



Wrapping the Transition Component

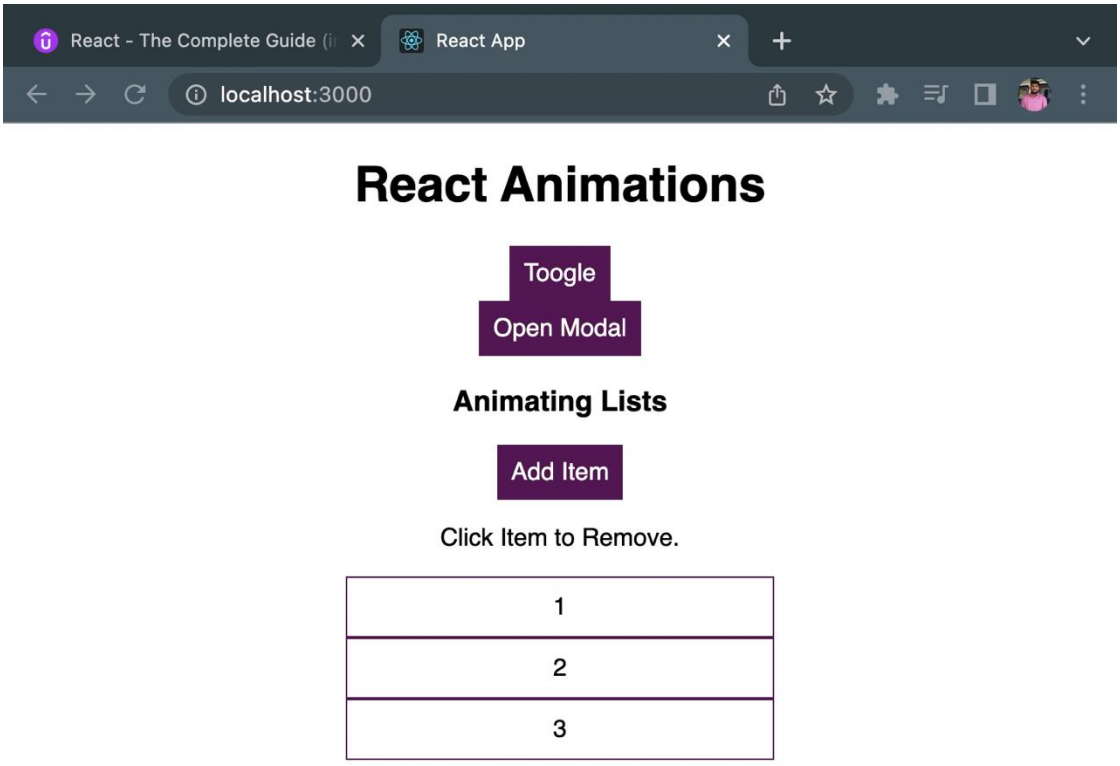
It will work the same way as the previous demo video.

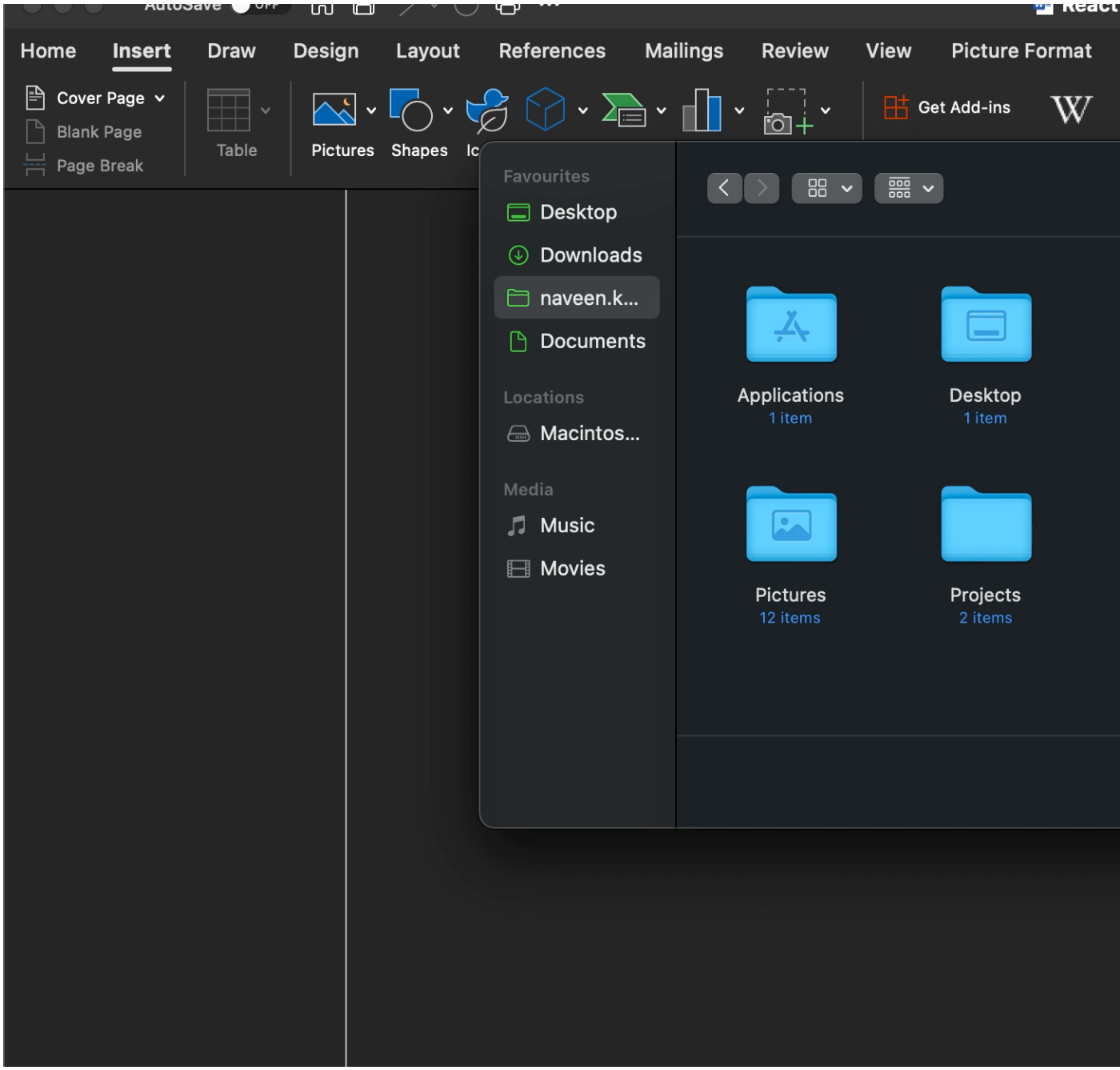
```
        </button>
        <Transition
          mountOnEnter
          unmountOnExit
          in={this.state.showBlock}
          timeout={300}
        >
          Complexity is 3 Everything is cool!
          {(state) => (
            <div
              style={{
                backgroundColor: "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>
        <h3>Animating Lists</h3>
```

```
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;
```

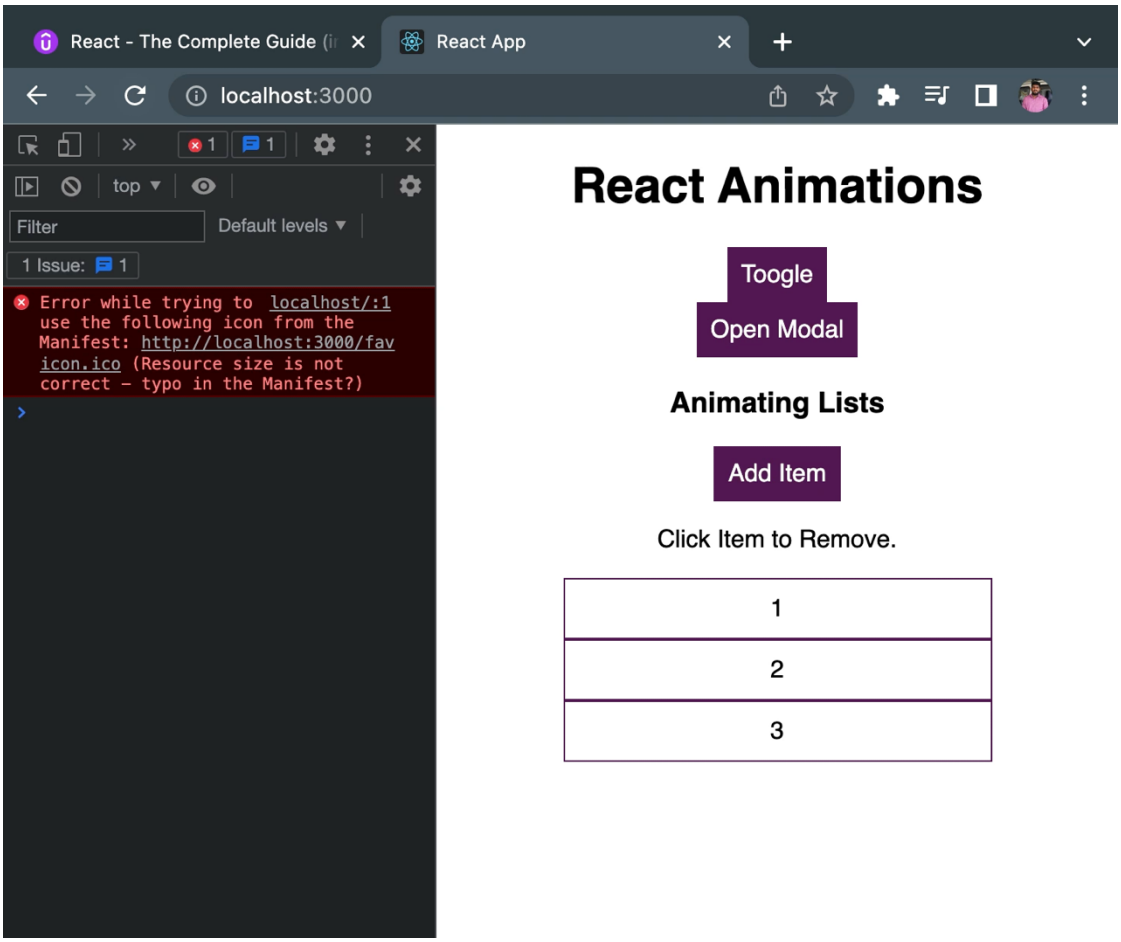
Animation Timings

```
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,
        ].
```




Home
Insert
Draw
Design
Layout
References
Mailings
Review
View
Picture Format
Cover Page
Blank Page
Page Break
Table
Pictures
Shapes
Get Add-ins
Favourites
Desktop
Downloads
naveen.k...
Documents
Locations
Macintos...
Media
Music
Movies
Applications
1 item
Desktop
1 item
Pictures
12 items
Projects
2 items

```jsx
<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.

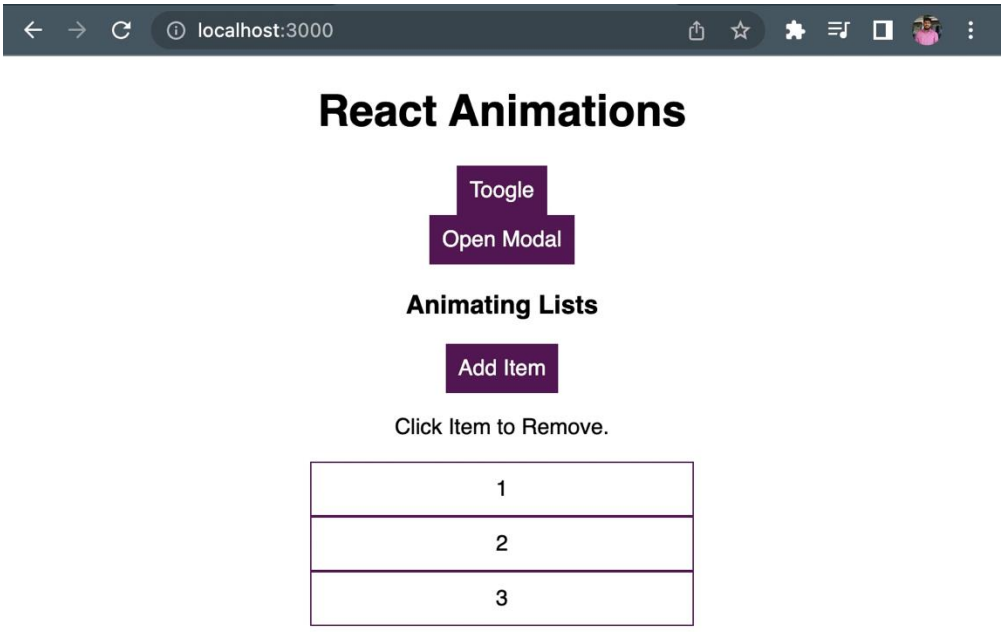
```jsx
import CSSTransition from 'react-transition-group/CSSTransition';
import './Modal.css';

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

```css
.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;
}
```

Using a classnames in CSSTransition component

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

Using TransitionGroup to manage multiple transitions

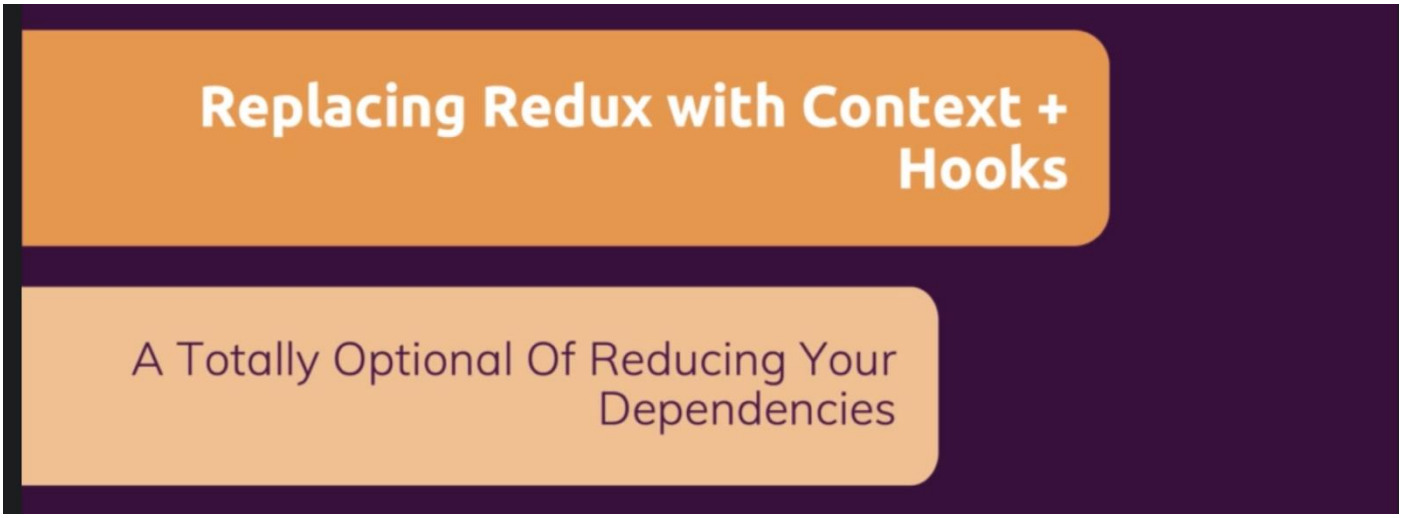


```
Complexity is 10 It's time to do something...
render () {
    Complexity is 4 Everything is cool!
    const listItems = this.state.items.map((item, index) => (
      <CSSTransition key={index} classNames="fade" timeout={300}>
        <li
          className="ListItem"
          onClick={() => this.removeItemHandler(index)}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

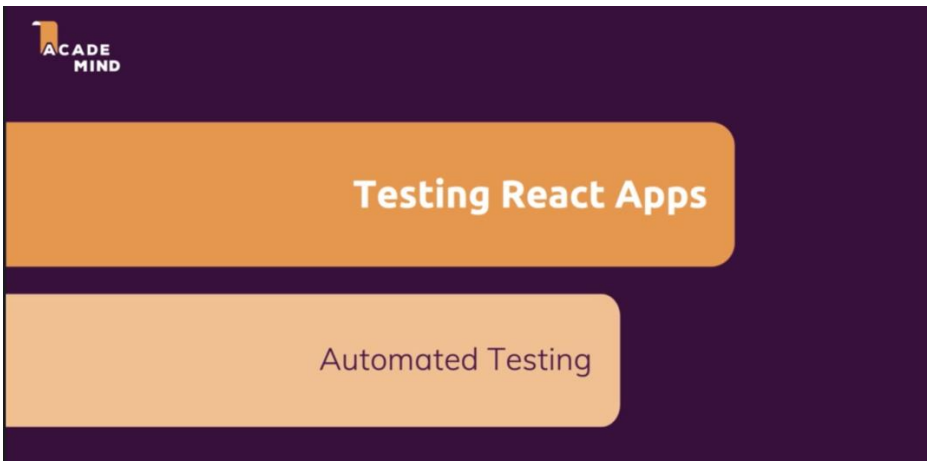
Replacing Redux with React Hooks

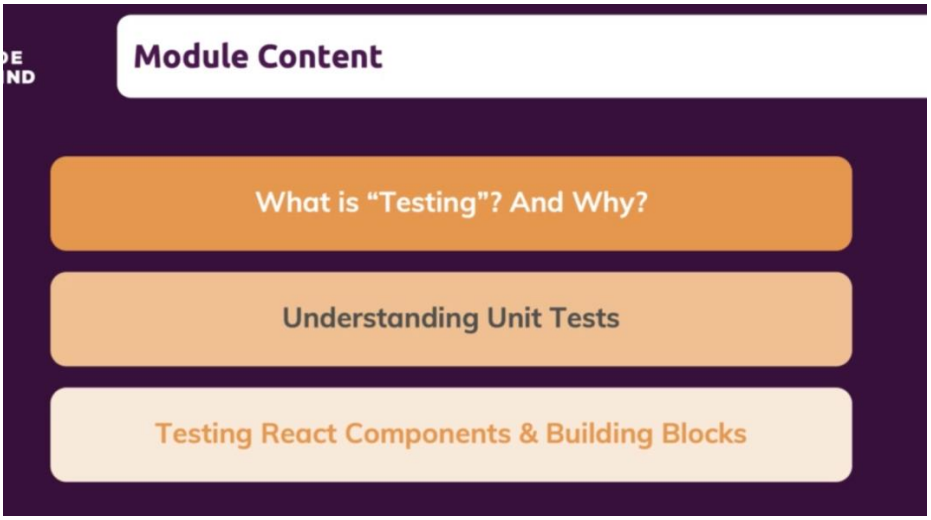


```
import { useState, useEffect } from 'react';

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 };
};
```

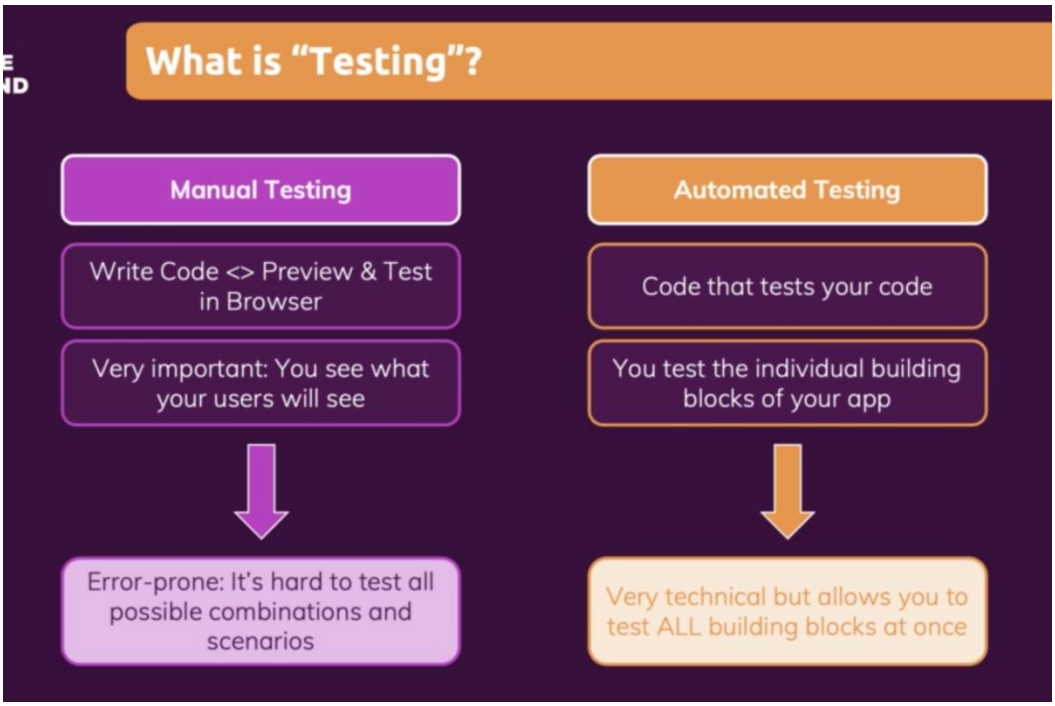
```
import { initStore } from './store';

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
```

Testing React App

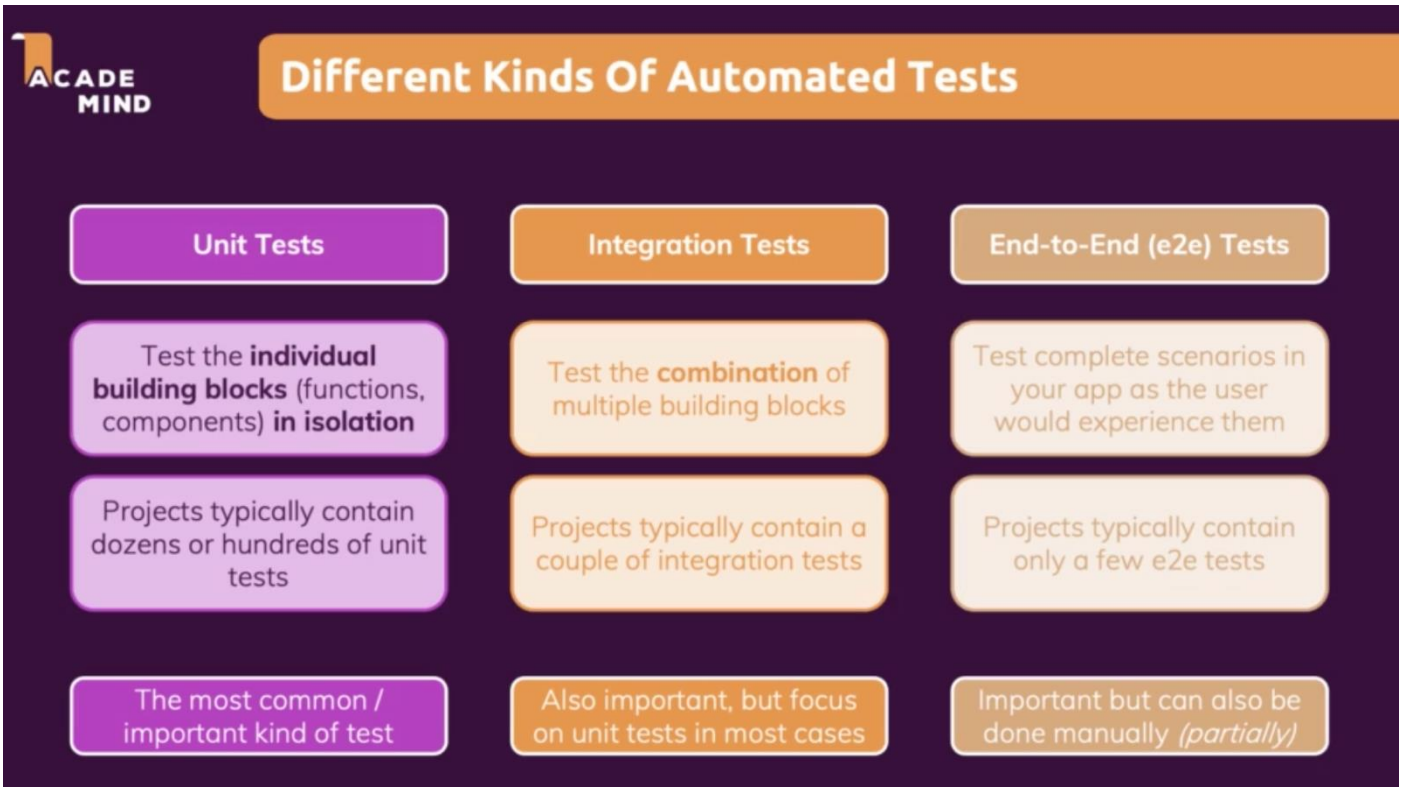




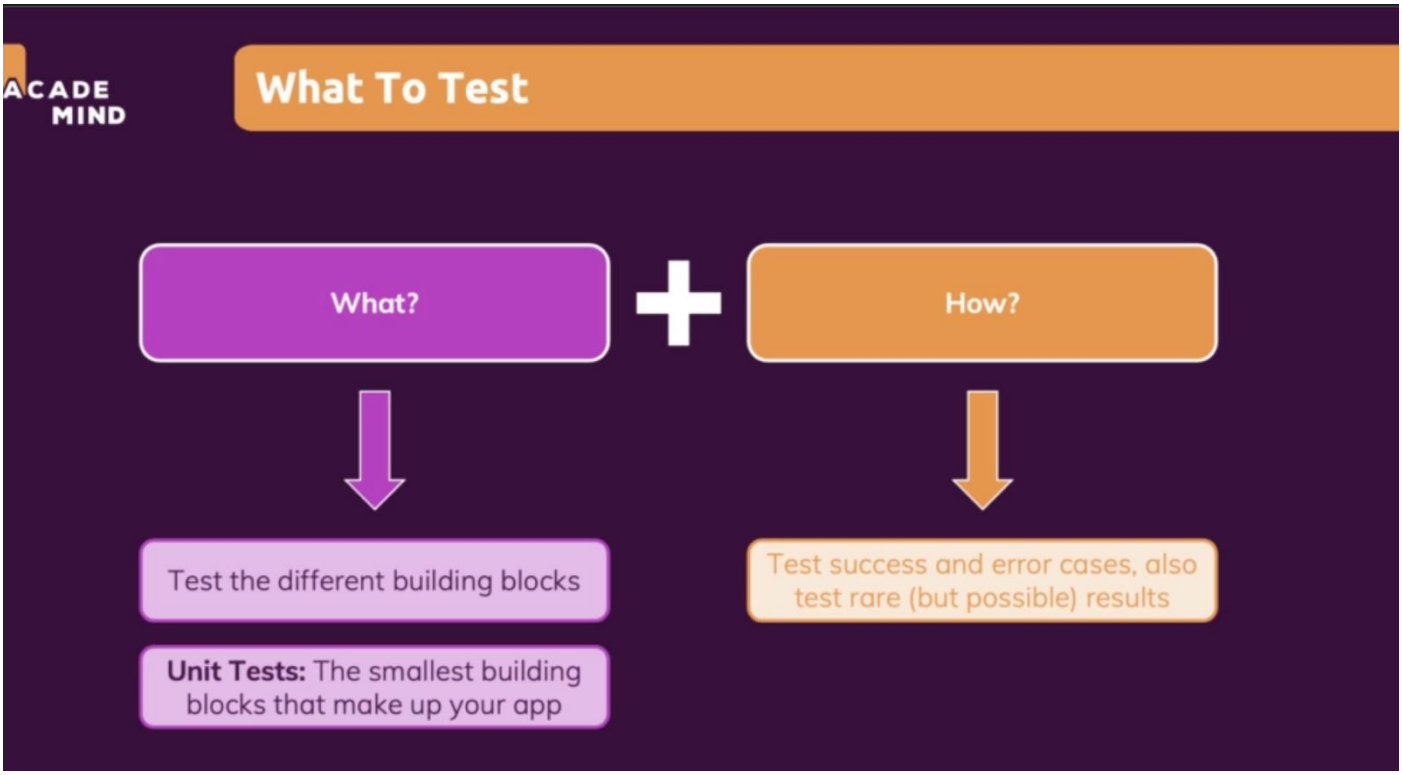
What is Testing?



Different Kind of Test

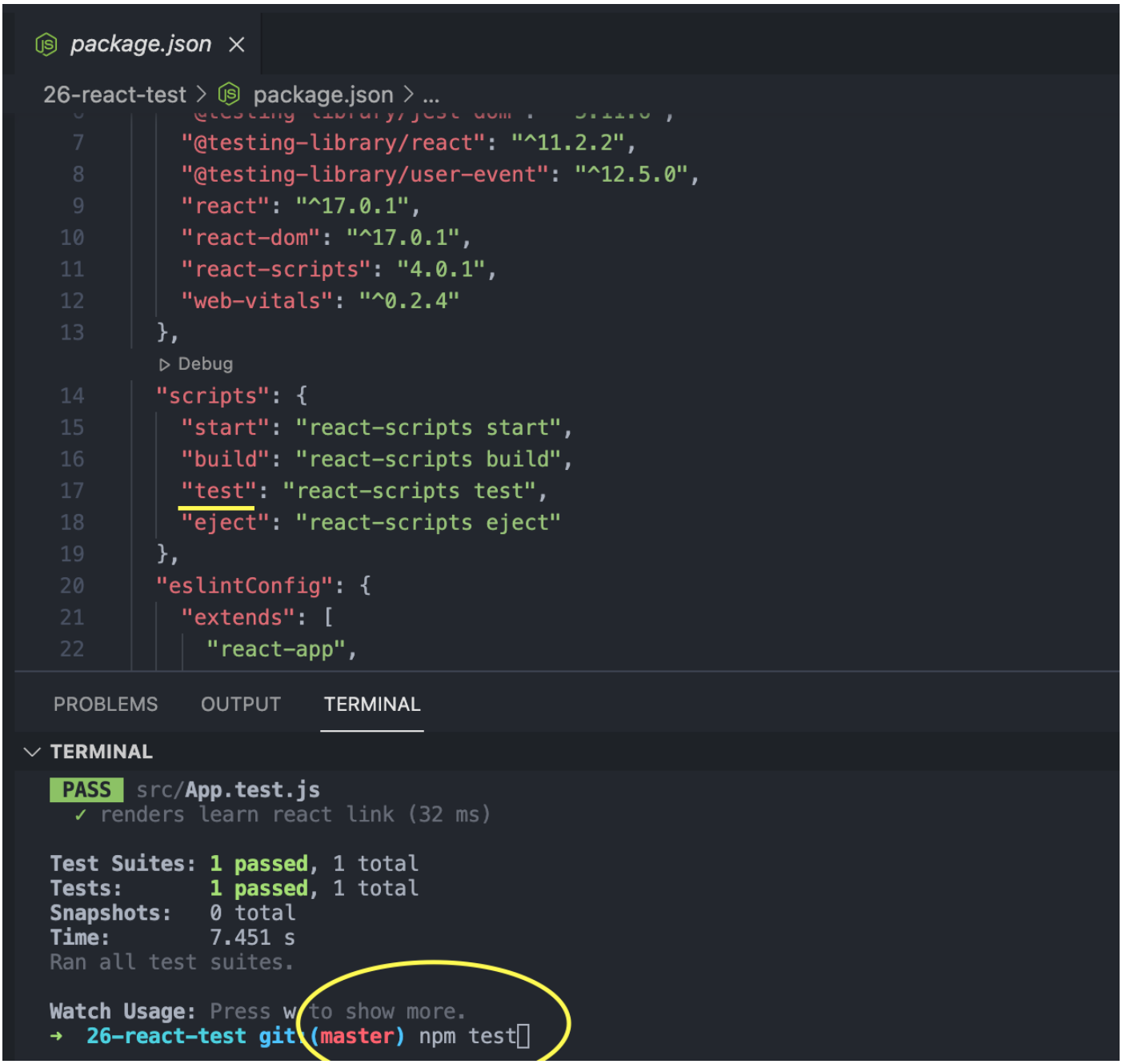


What to test ?

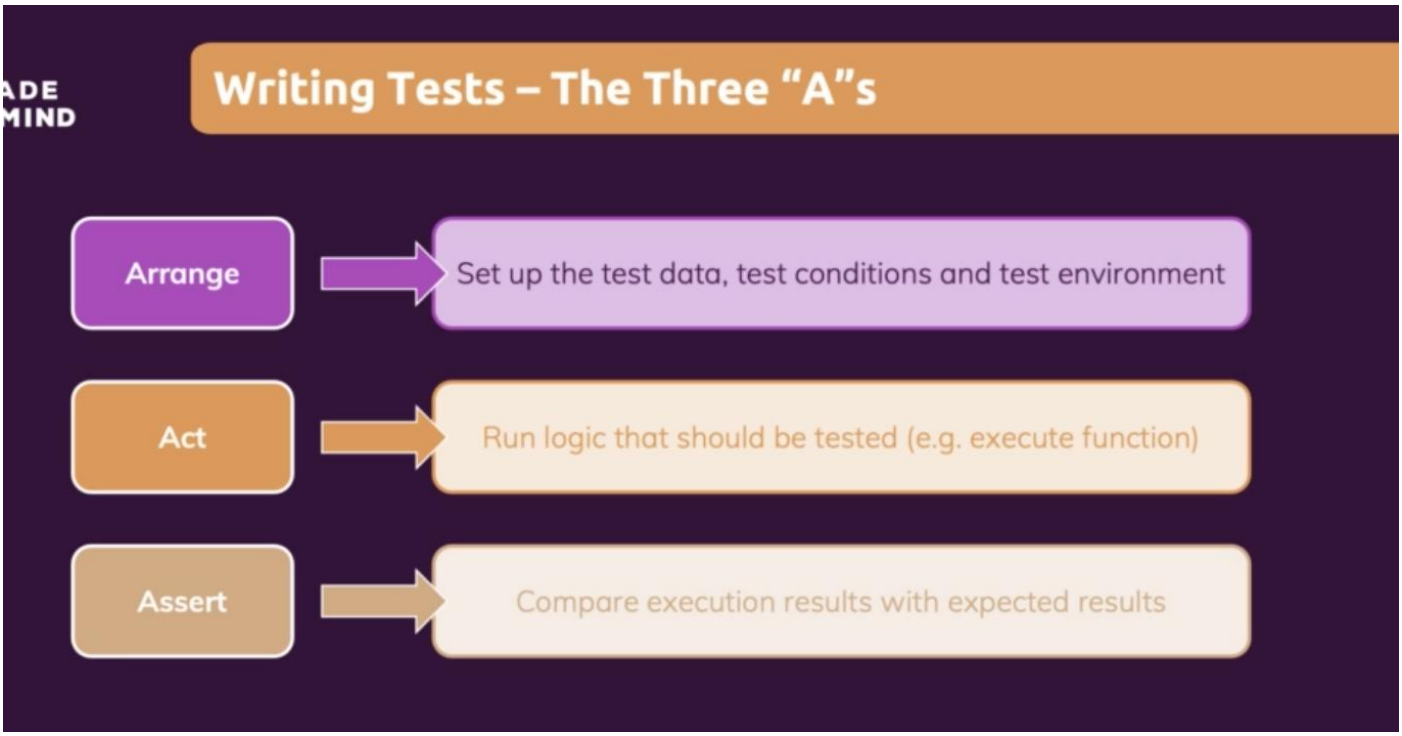


Running the test

```
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();
});
```



Writing a test




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

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

  //Act
  //... nothing

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

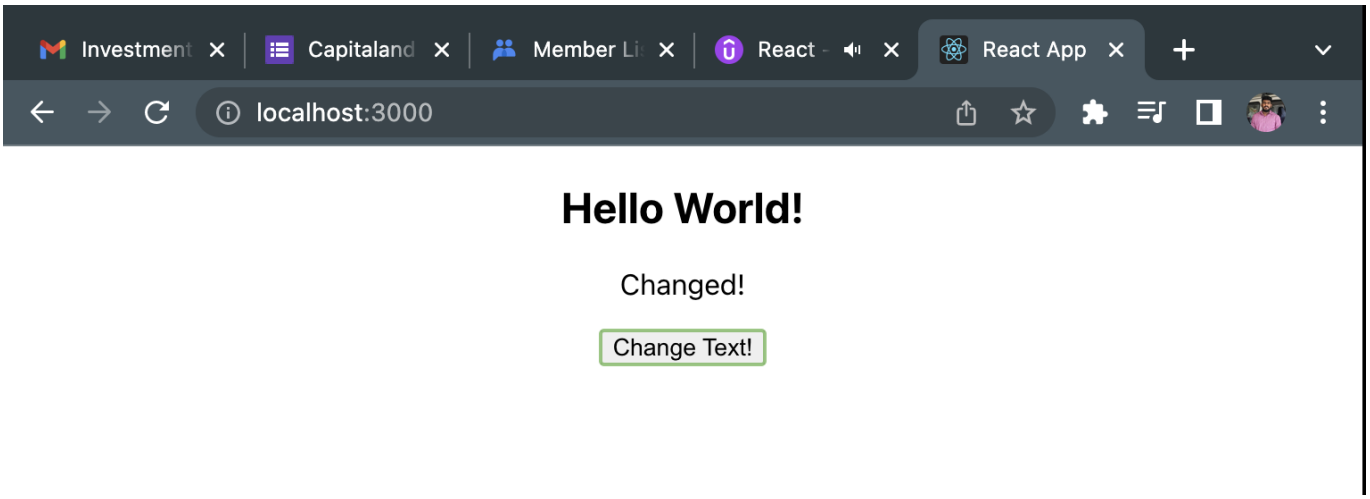
Here exact is true out of the box.

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

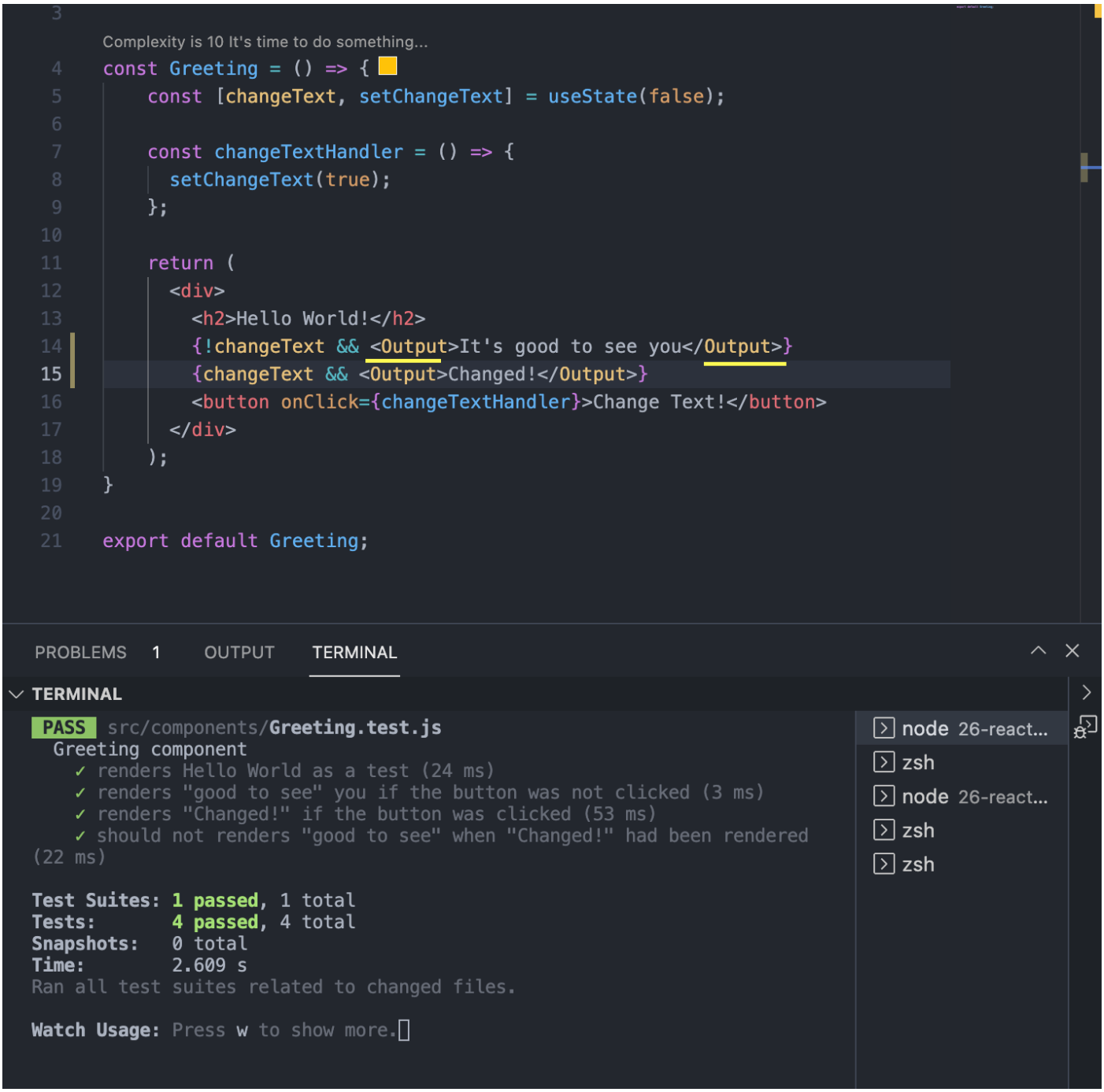


```jsx
import { useState } from "react";

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.



```jsx
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;
```

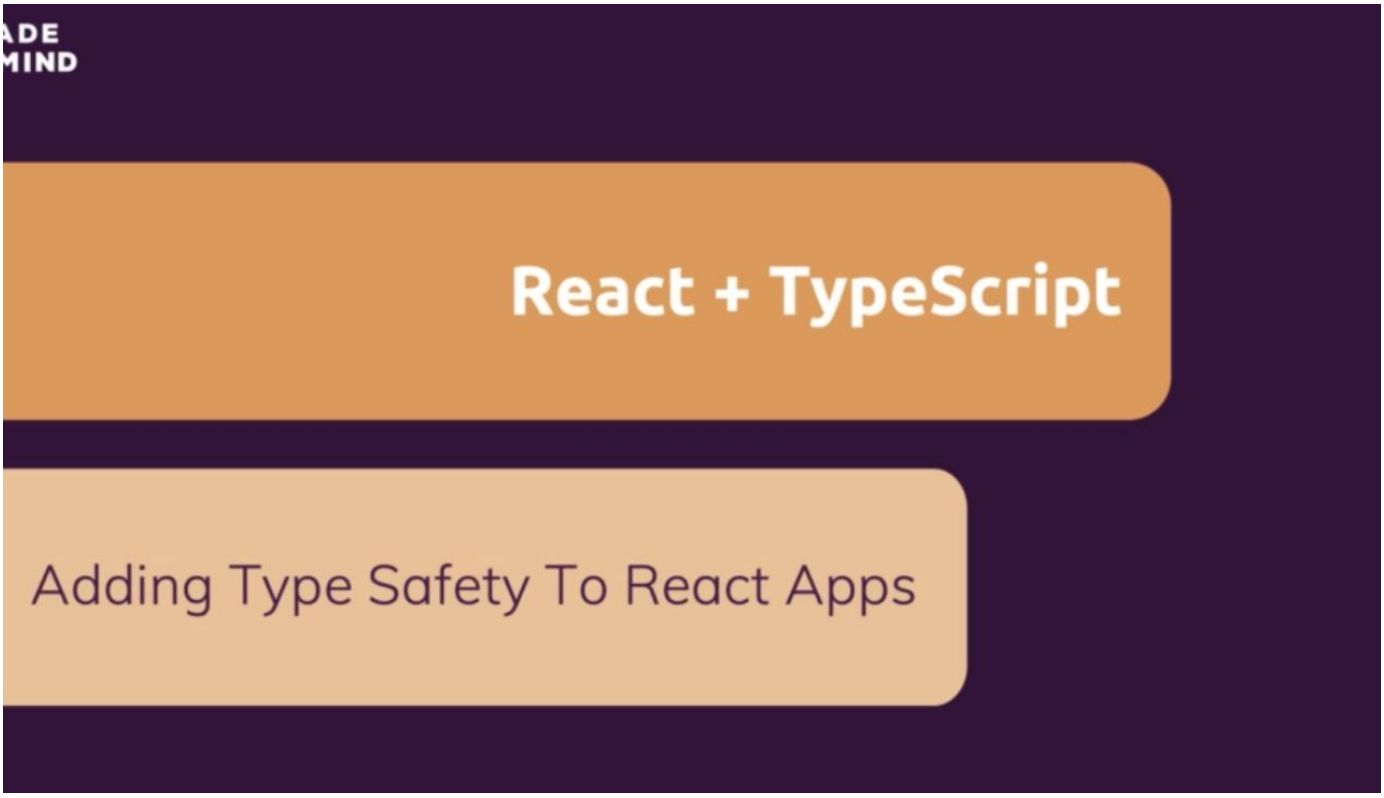
```jsx
import { render, screen } from "@testing-library/react";
import userEvent from '@testing-library/user-event';
import Async from "./Async";

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).not.toHaveLength(0);
    });
});
```

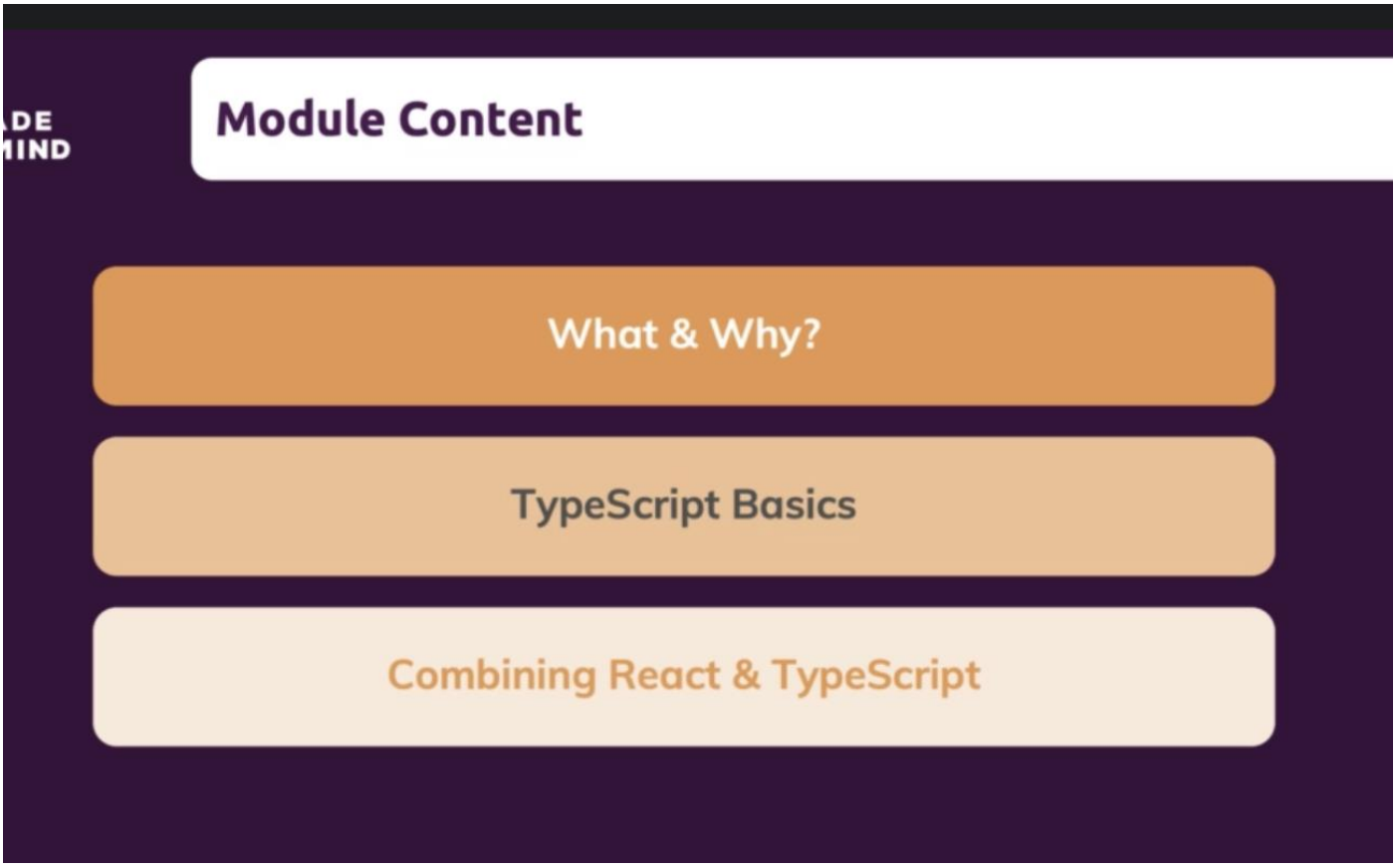
Working with Mock

```
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', ()=> {
    Complexity is 3 Everything is cool!
    test('renders posts if request succeeds', async ()=>{
        window.fetch = jest.fn();
        window.fetch.mockResolvedValueOnce({
          json: async () => [{ id: "p1", title: "First post" }],
        });
        render(<Async />);
        const listItemElements = await screen.findAllByRole("listitem");
        expect(listItemElements).not.toHaveLength(0);
    });
});
```

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

React + Typescript





What and Why?

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.

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) ✗
```

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);
```

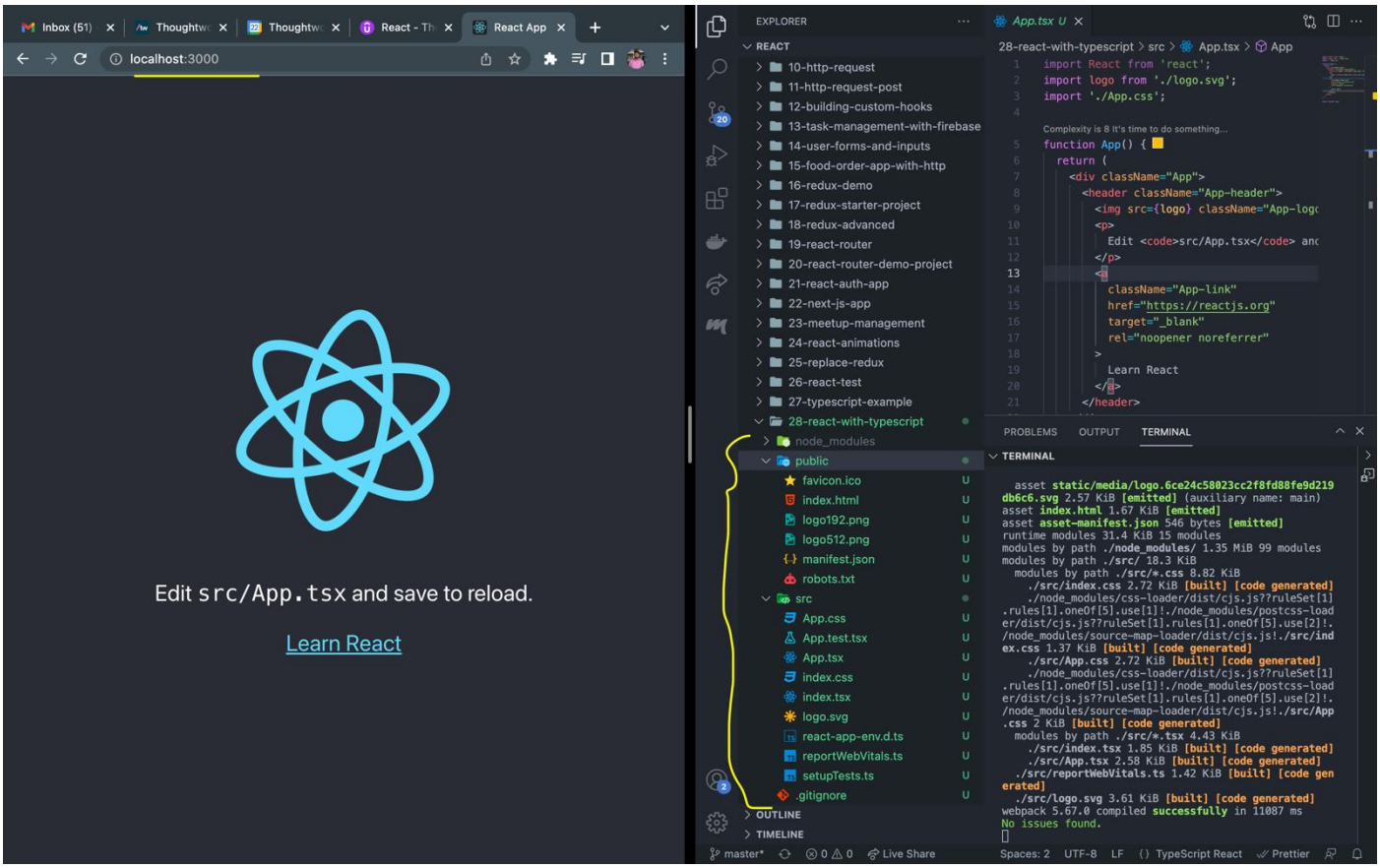
```
//Generics

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

const demoArray = [1,2,3];

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

```tsx
//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");
```

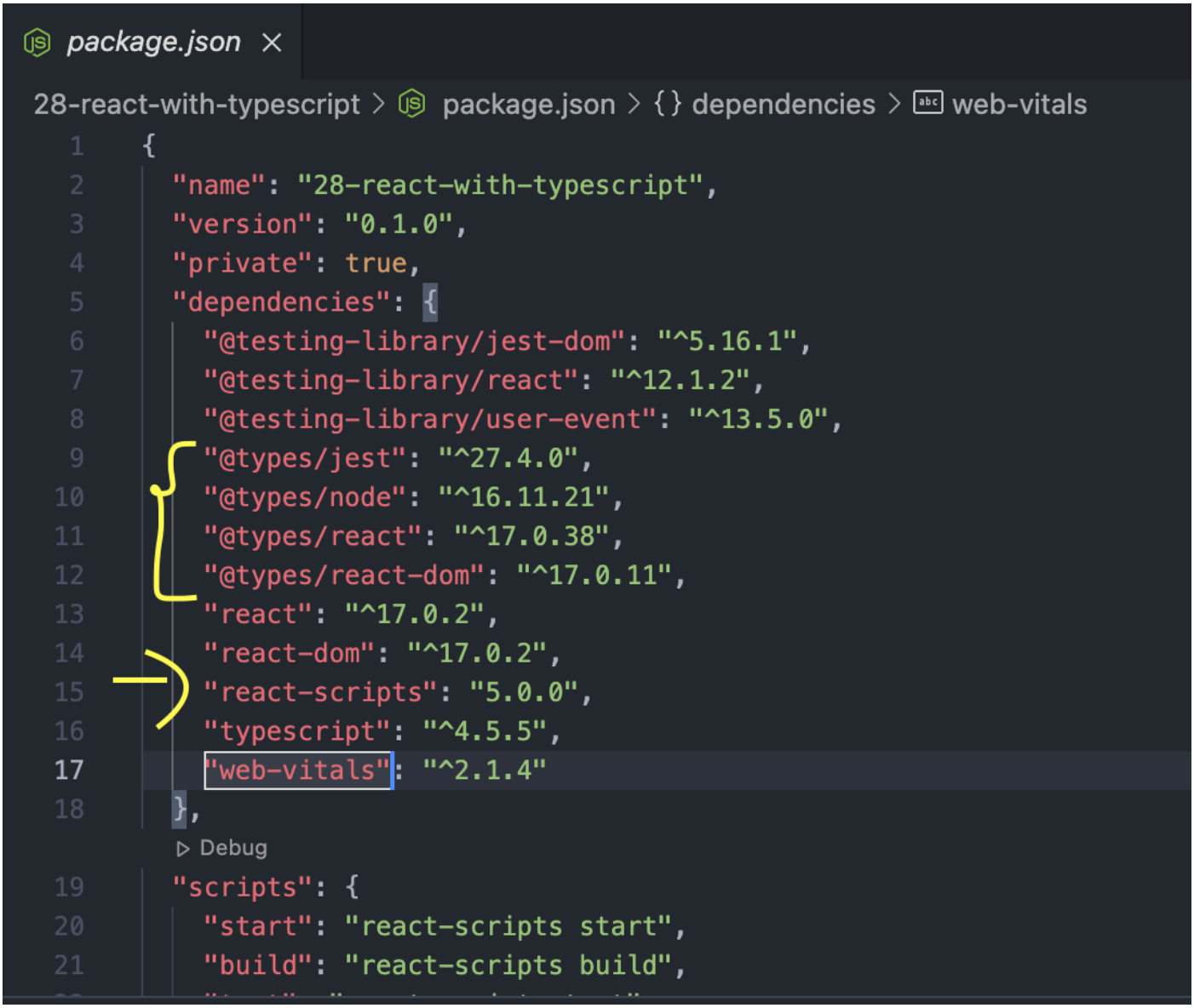
Working with React TypeScript Project

Creating a project

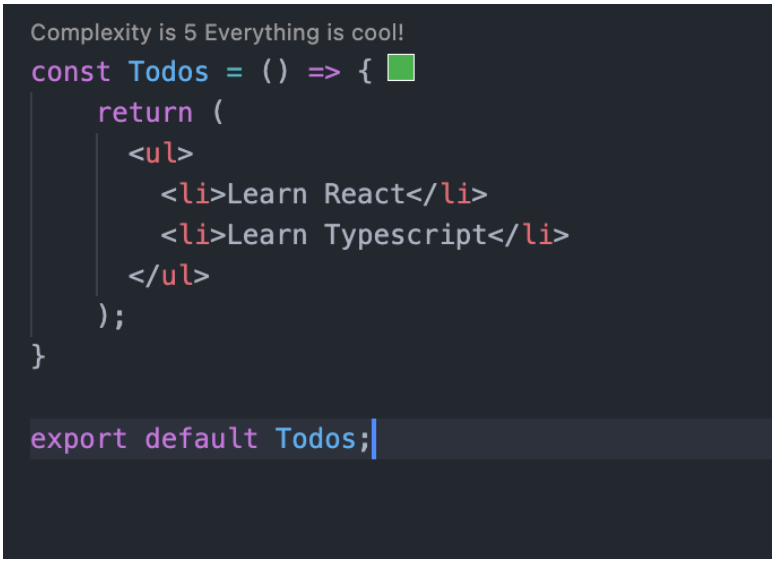
```
→  REACT git:(master) npx create-react-app 28-react-with-typescript --template typescript

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
```

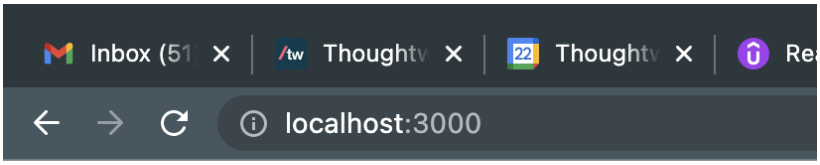


To run the typescript code, compiler internally convert the typescript code to JavaScript code. Below libraries from package.json file helps to do the conversion.



Working with component and TypeScript





- Learn React
- Learn Typescript

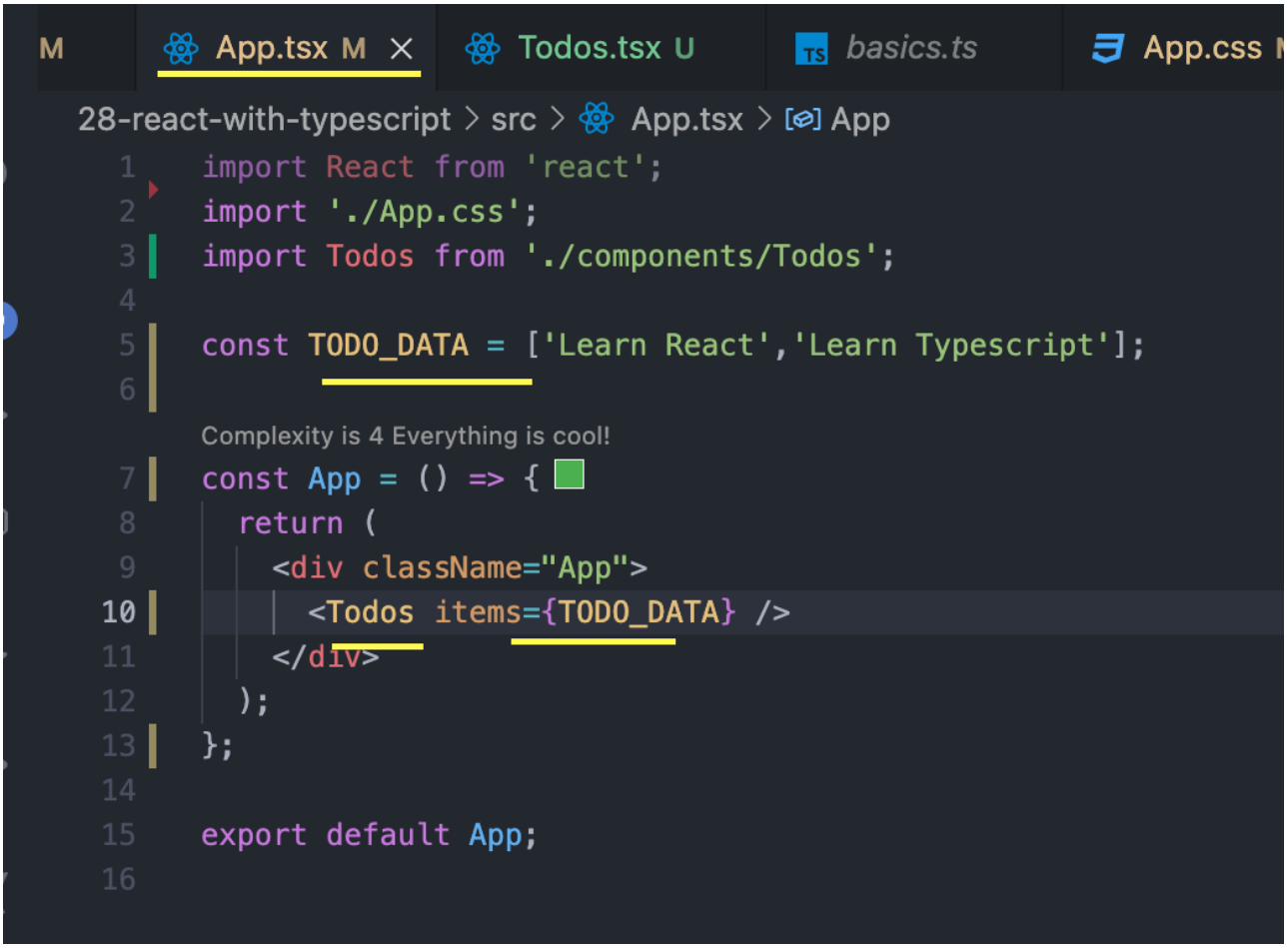
Working with props and Typescript

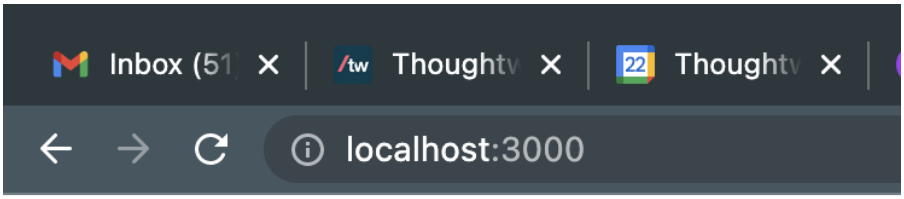
```
Complexity is 3 Everything is cool!
const Todos: React.FC = (props) => {
  return (
    <ul>
      {props.children}
    </ul>
  );
};

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.

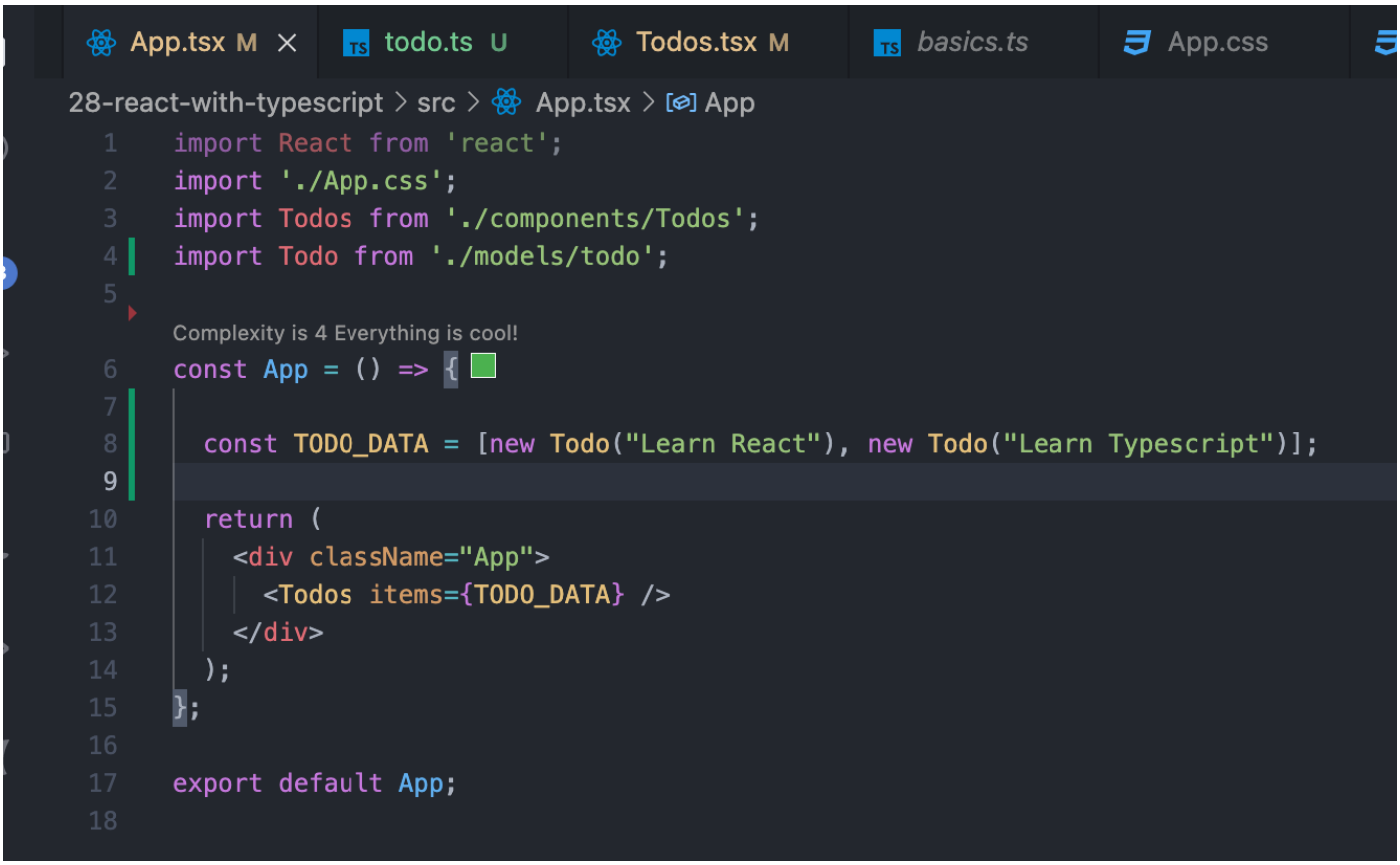


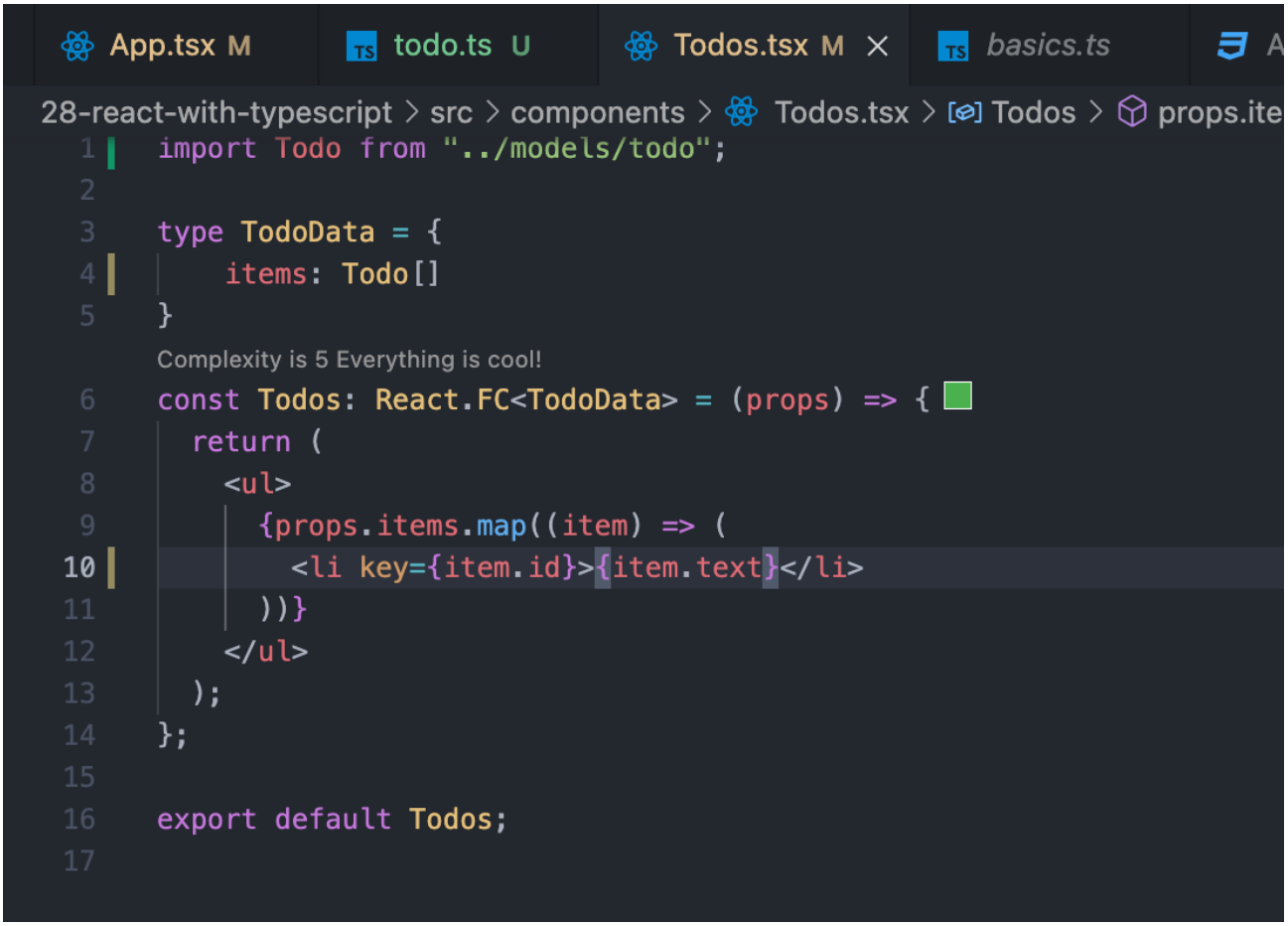


- Learn React
- Learn Typescript

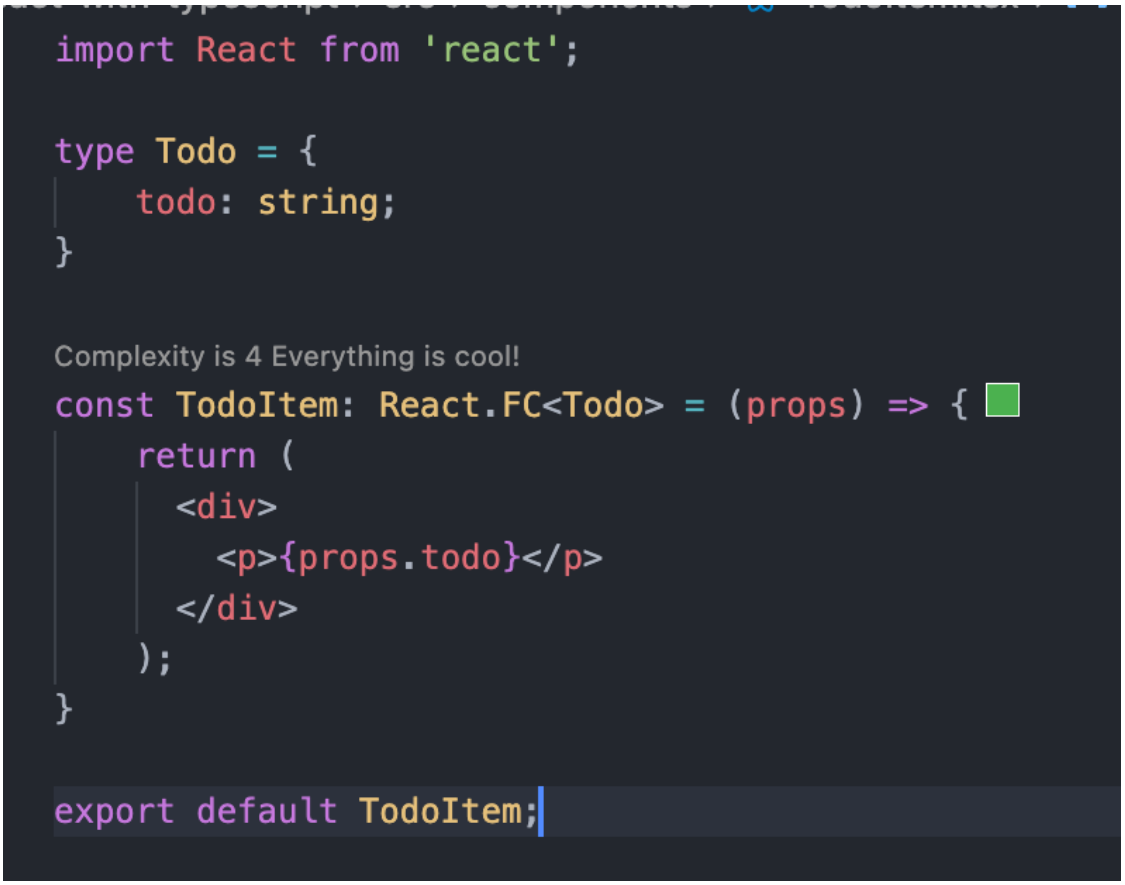
Adding a Data Model

Adding a data model to represent a Todo data.





Sperate out by Adding a TodoItem component



```tsx
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;
```

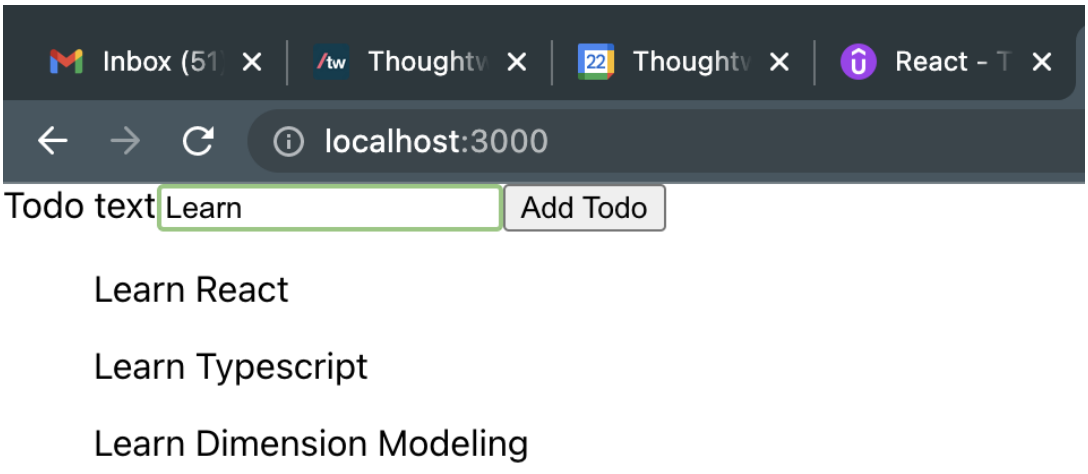
As we can in TodoItem props, key can be added even though it not specified in Todo type. “key” props is default to React.FC type annotation. So, it is merging it.

Working with ref, useRef and Function Props

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

```tsx
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;
```

```
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

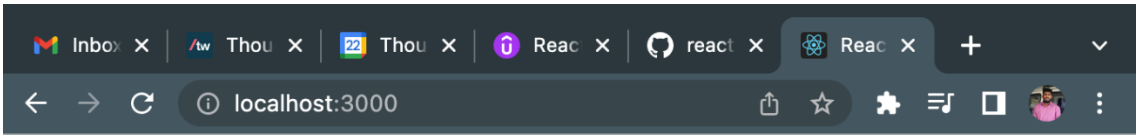
```
import React, { useEffect, useState } from 'react';
import './App.css';
import NewTodo from './components/NewTodo';
import Todos from './components/Todos';
import Todo from './models/todo';

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 probs.
- So, this help us to provide the future parameter reference to a function.

```tsx
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.

```
eact-with-typescript > src > components > TodoItem.tsx > 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

TodoItem.tsx | todo-context.tsx M | App.tsx M | Todos.tsx M | NewTodo.tsx M

28-react-with-typescript > src > store > todo-context.tsx > TodosContextProvider

```tsx
import React, { useState } from 'react';
import Todo from '../models/todo';

type TodoContextType = {
    items: Todo[];
    addTodo: (text: string)=>void;
    removeTodo : (id: string)=>void;
}

export const TodoContext = React.createContext<TodoContextType>({
  items: [],
  addTodo: () => {},
  removeTodo: (id: string) => {},
});

Complexity is 10 It's time to do something...
const TodosContextProvider : React.FC = (props)=>{
    const [todos, setTodos] = useState<Todo[]>([]);

    Complexity is 4 Everything is cool!
    const removeTodoHandler = (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);
      });
    };

    const contextValue: TodoContextType = {
      items: todos,
      addTodo: addTodoHandler,
      removeTodo: removeTodoHandler,
    };

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

export default TodosContextProvider;
```

```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;
```

```tsx
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;
```

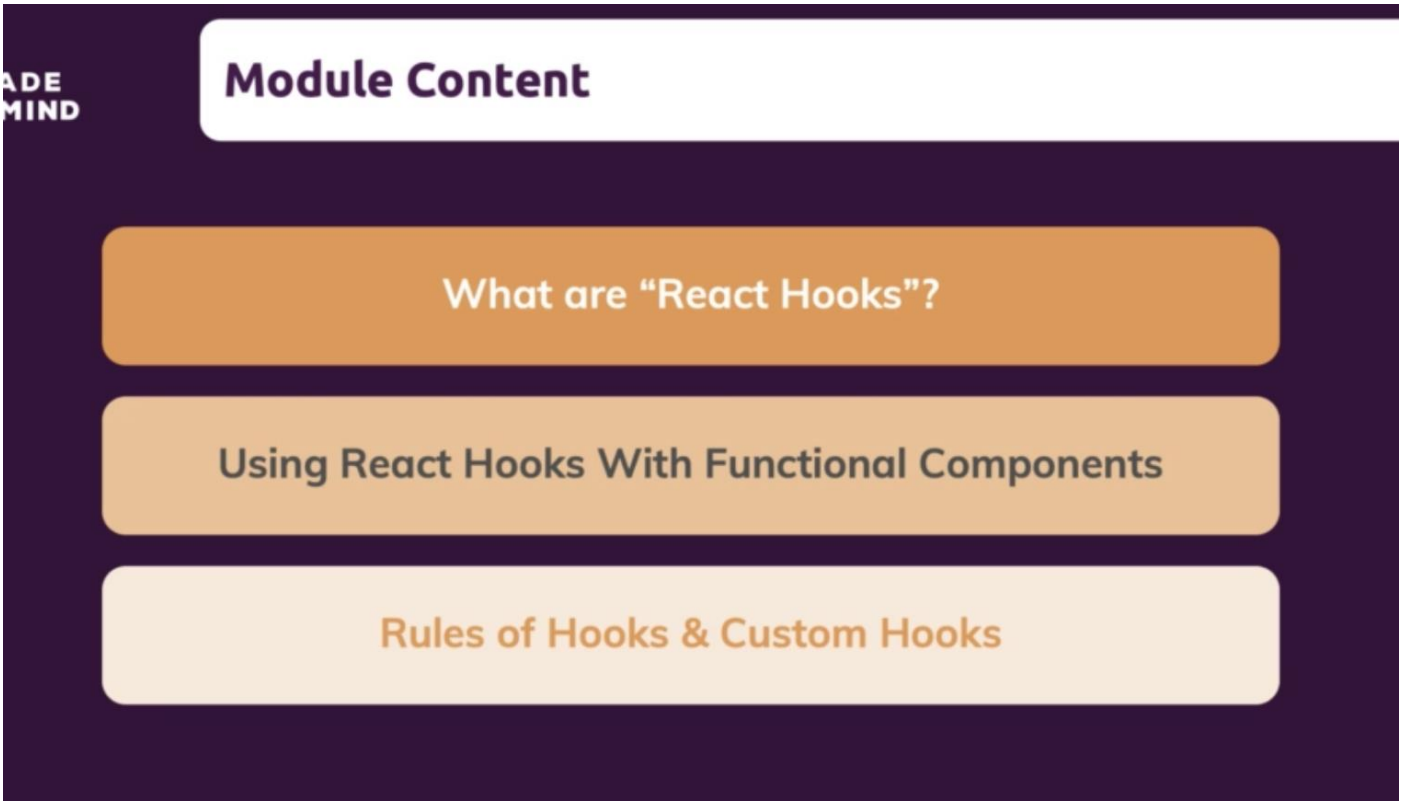
```
import TodoItem from "./TodoItem";
import classes from "./Todos.module.css";
import { TodoContext } from "../store/todo-context";
import { useContext } from "react";

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

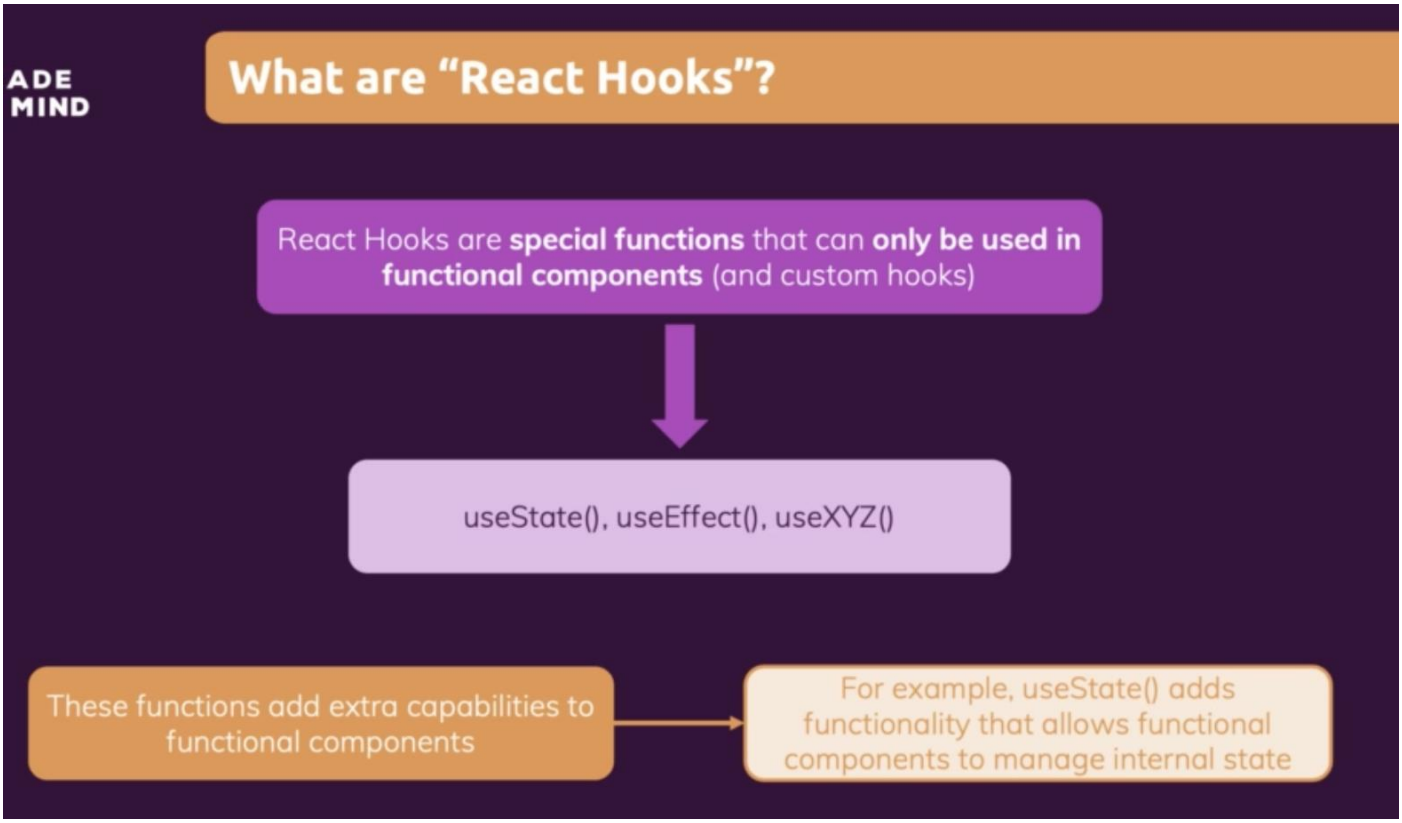
This file take care of the configuration for typescript. Like libraries to manage etc.


eact-with-typescript > tsconfig.json > ...
{
"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"
]
}

React Hooks



What are React Hooks ?

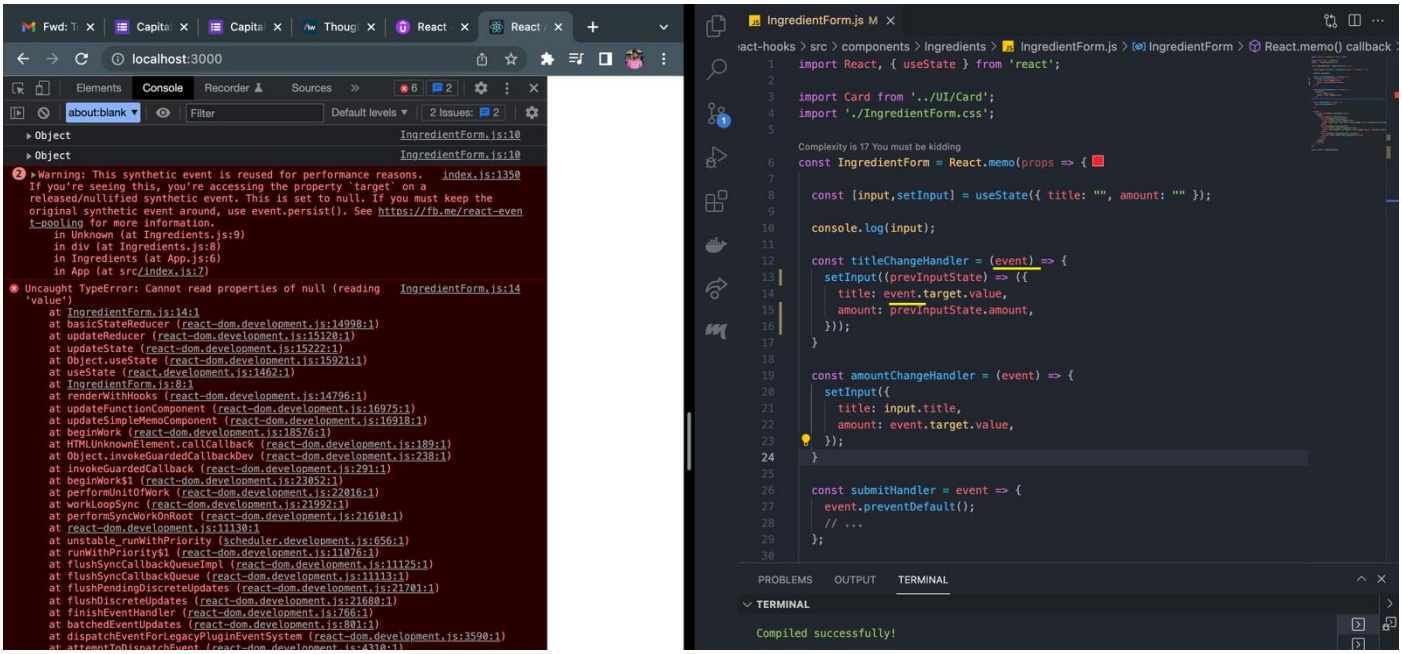


For class-based component, state is always an object.

useState Hook

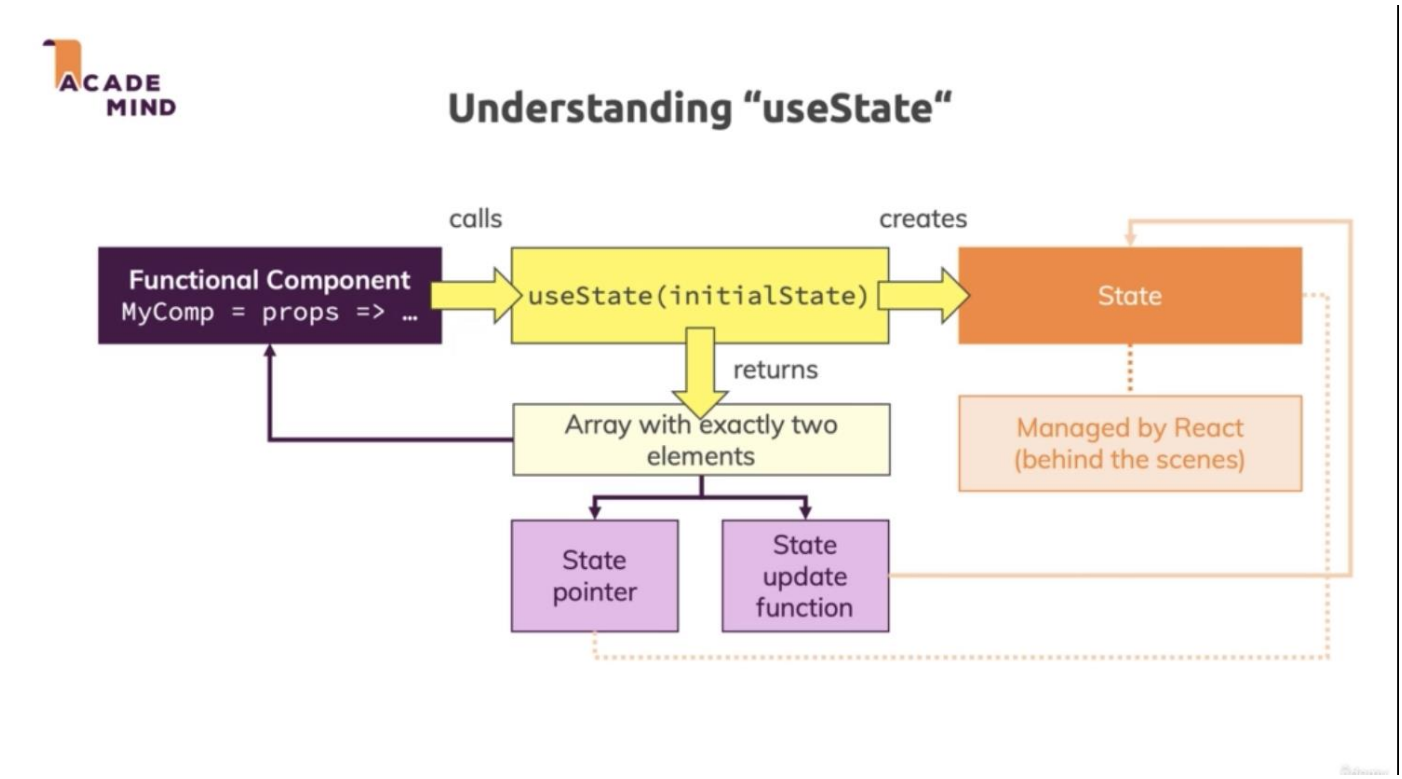
SetStateFunction with Previous state

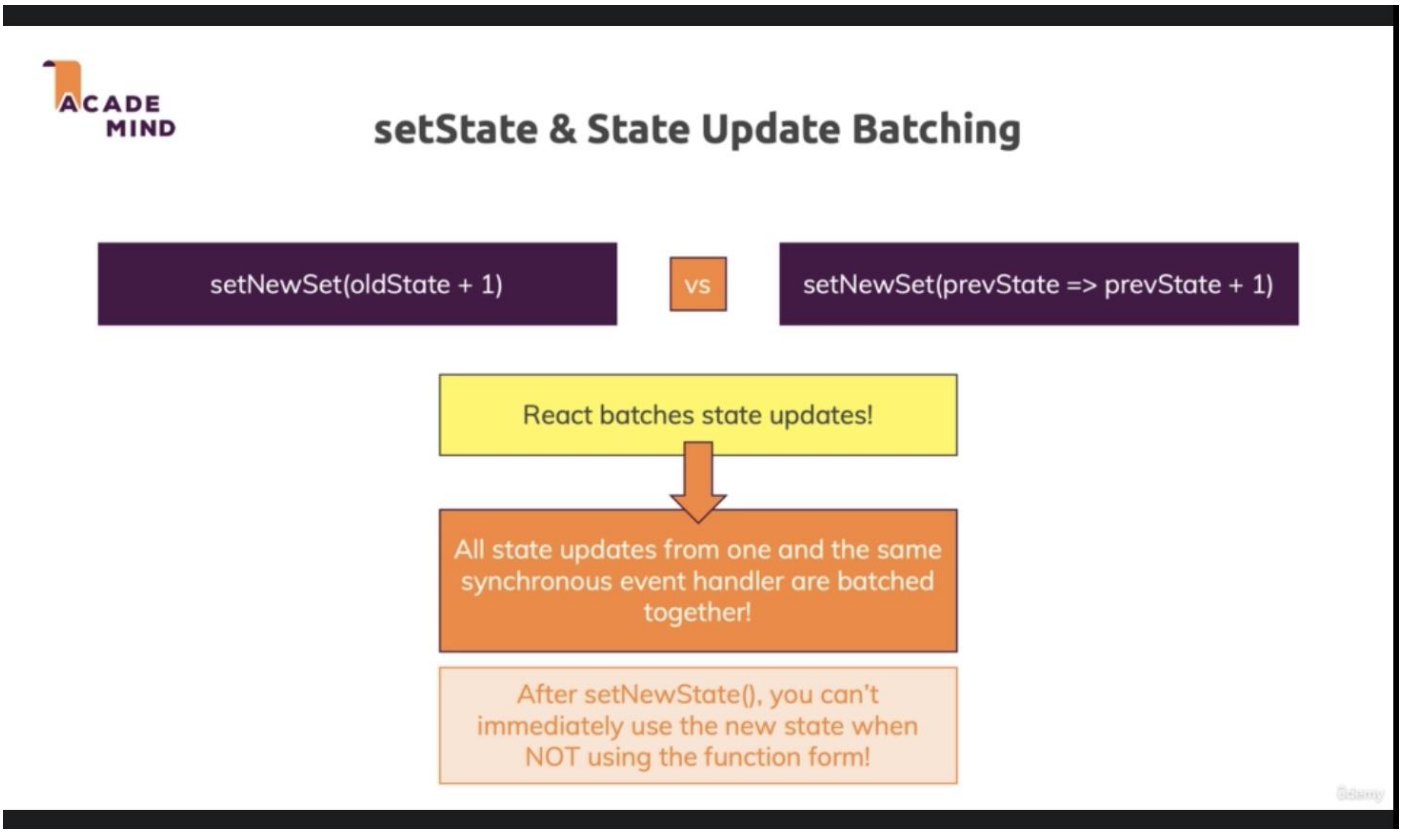
Here, we are getting the error because we are using the event object inside the anonymous function. Because of this reason, evet object will be locked inside the set state function. The subsequent keystroke followed by first keystroke, try to reuse the first event object which has been locked inside. This in turn cause the error.



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,
  }));
}
```


ACADEMIND
Understanding "useState"
calls
creates
Functional Component
MyComp = props => ...
useState(initialState)
State
returns
Array with exactly two elements
Managed by React (behind the scenes)
State pointer
State update function


ACADEMIND
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

```
setName('Max');
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:

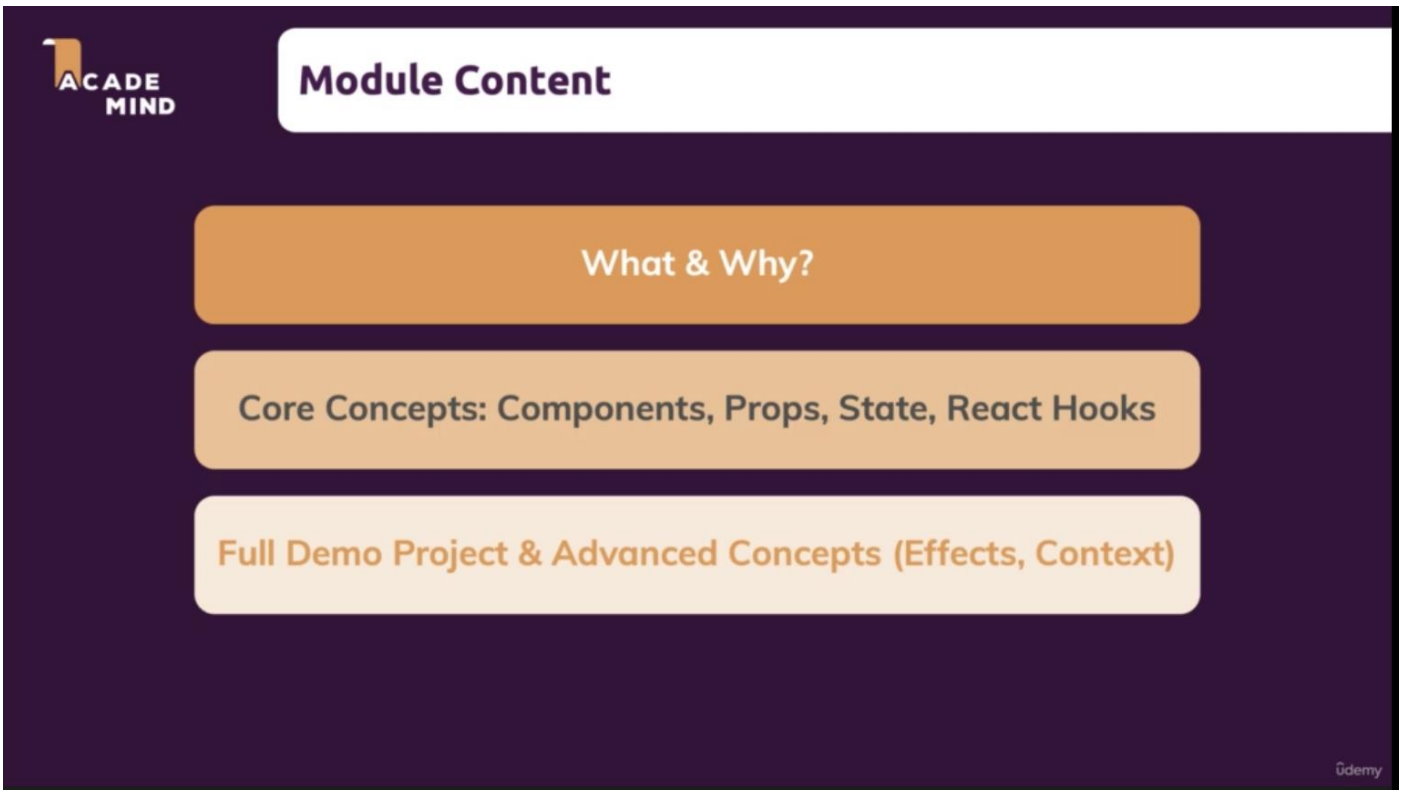
```
console.log(name); // prints name state, e.g. 'Manu'
setName('Max');
console.log(name); // ??? what gets printed? 'Max'?
```

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()`!

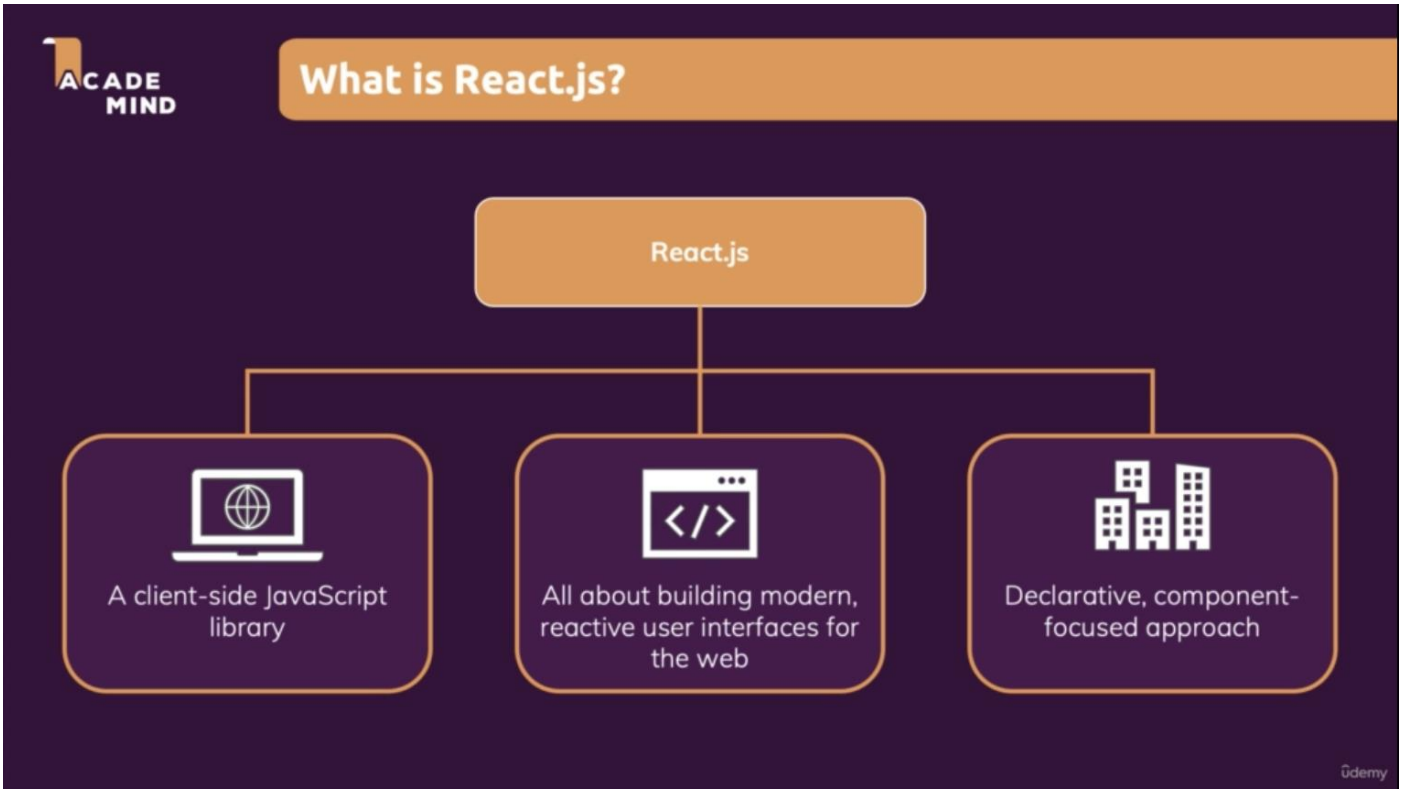
```
const clearError = useCallback(() => {
  dispatchHttp({ type: "CLEAR" });
}, []);

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>
);
};
```

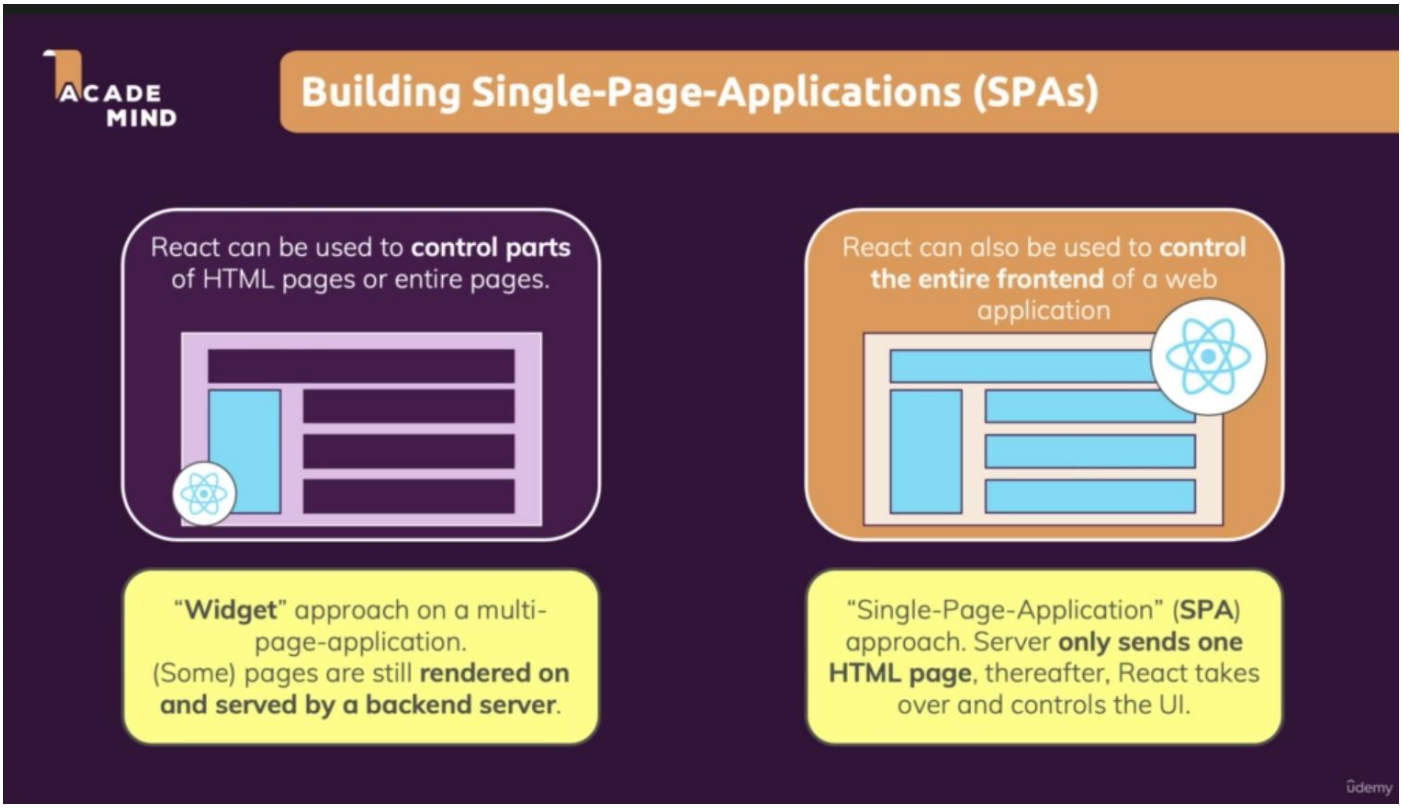
React Summary



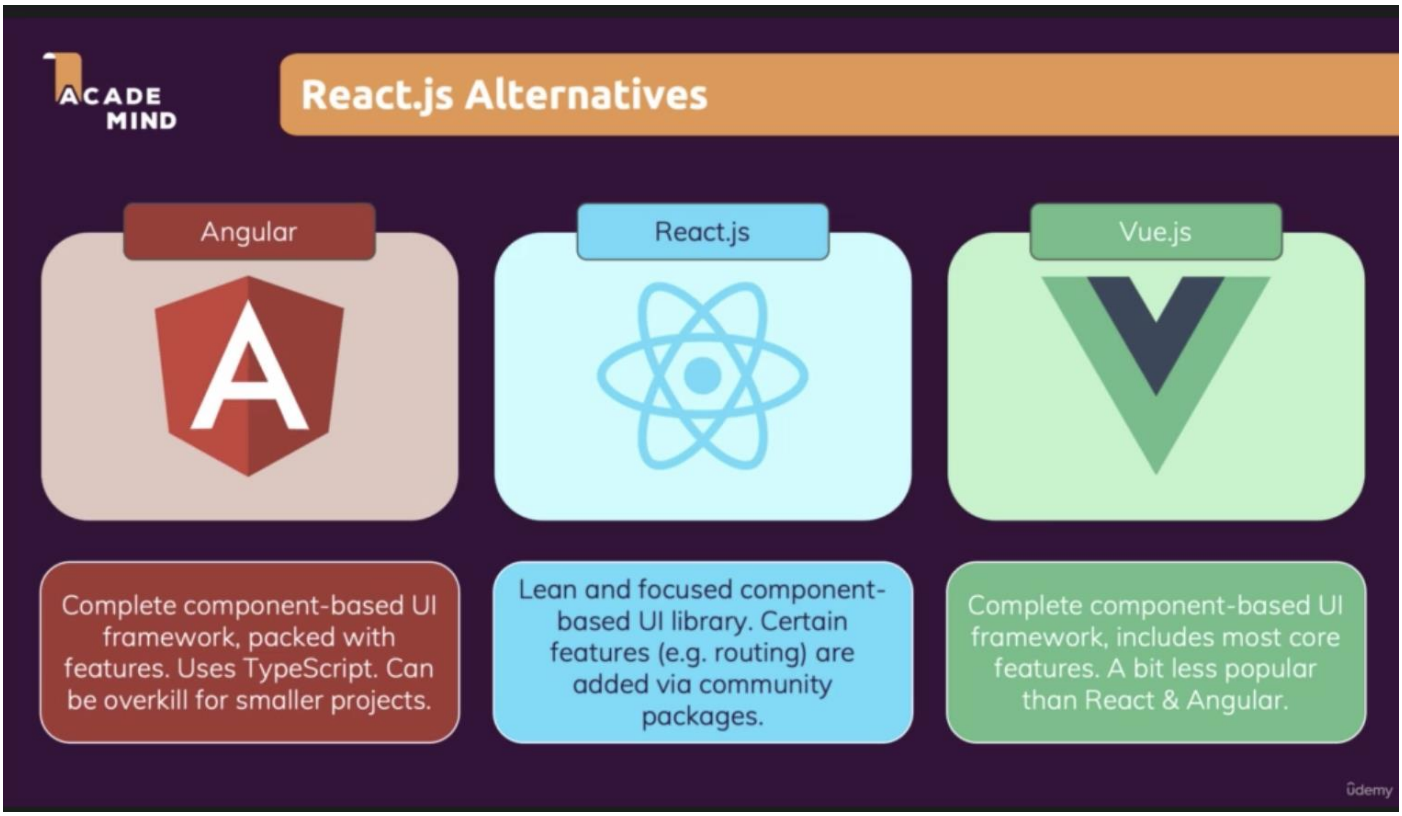
What is React ?

Java script library for building user interface.



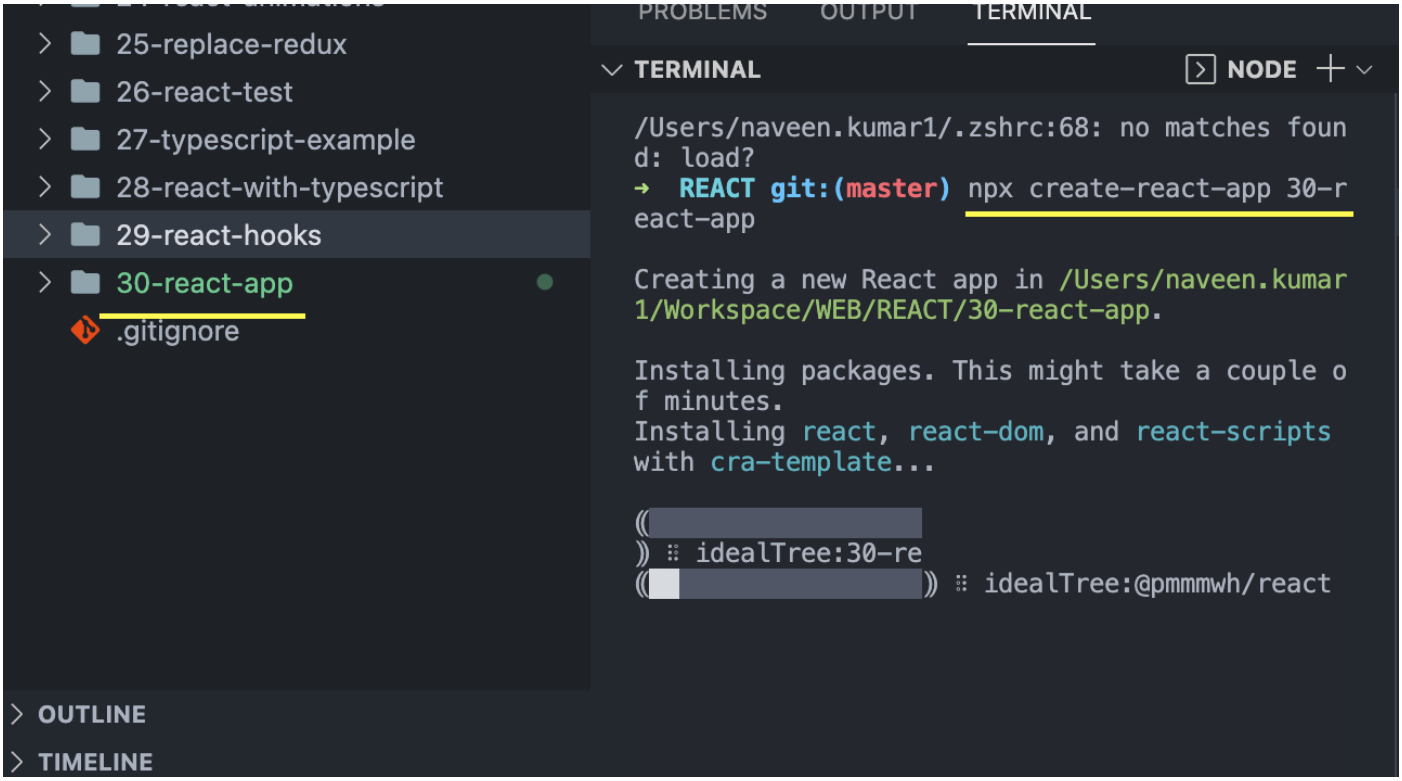


Reactive Alternative



Creating a React project

npx create-react-app my-app



React Eco System