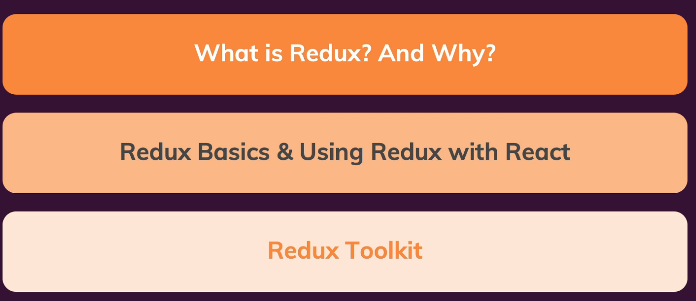
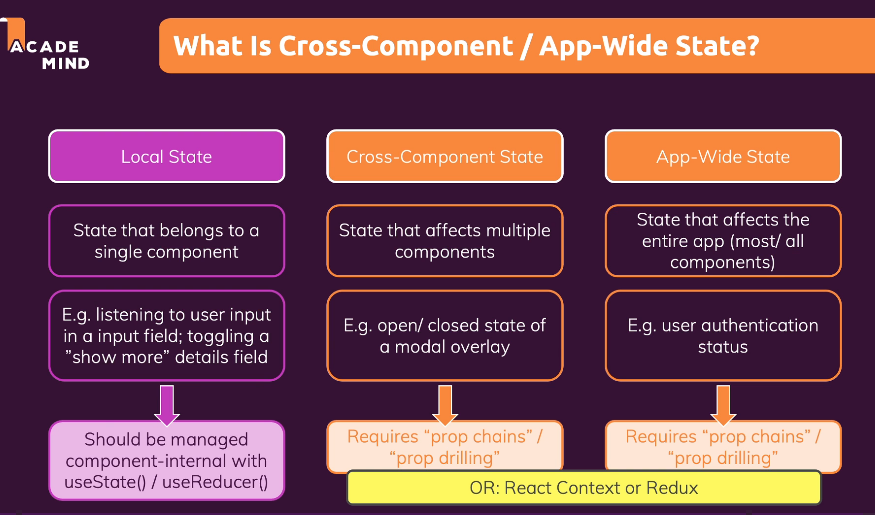
Section 18. Diving into Redux (An Alternative to the context)

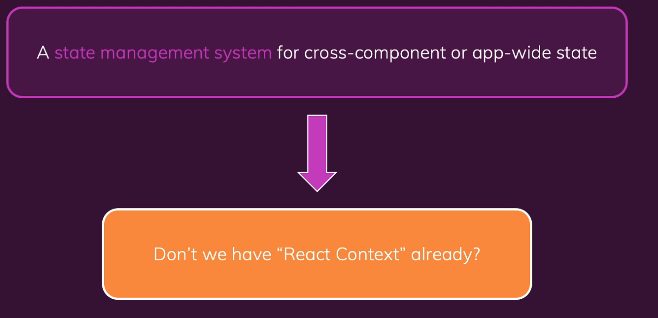


227. Another look at state in react apps



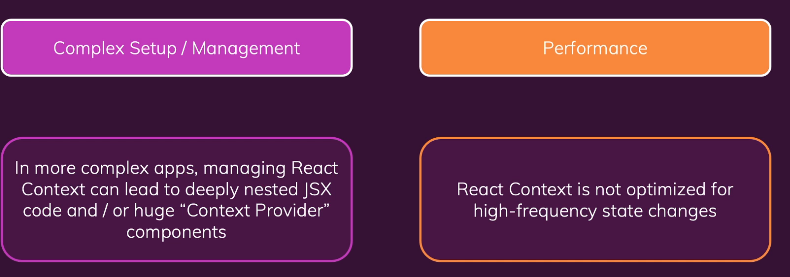
228. redux vs react Context

What is redux

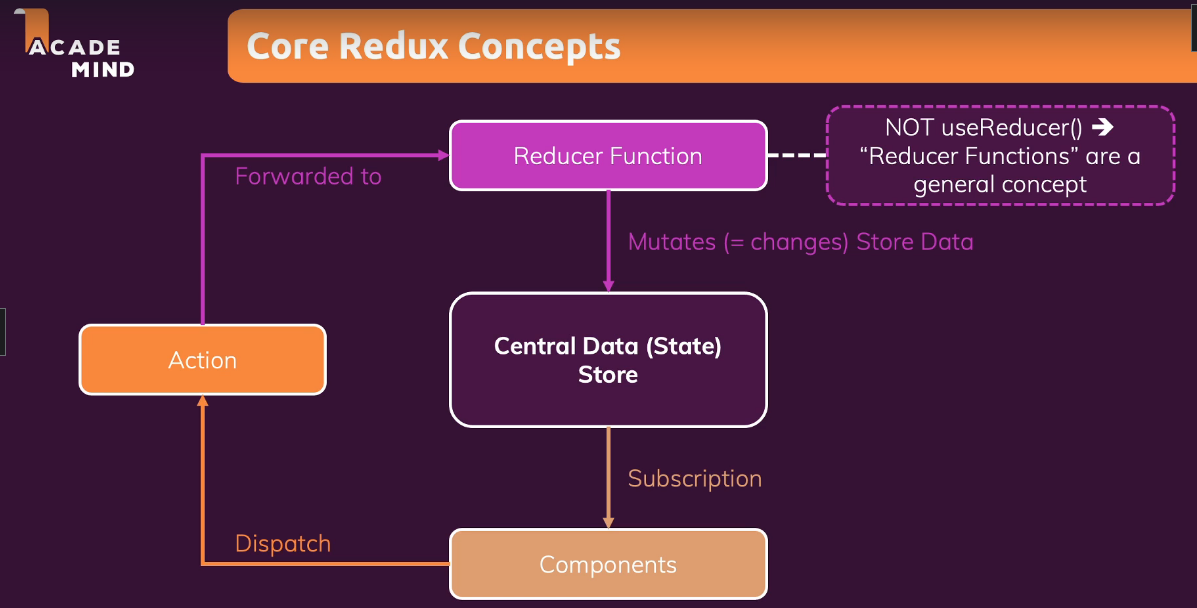


React Context – Potential Disadvantages





229. How Redux works



230. Exploring core redux

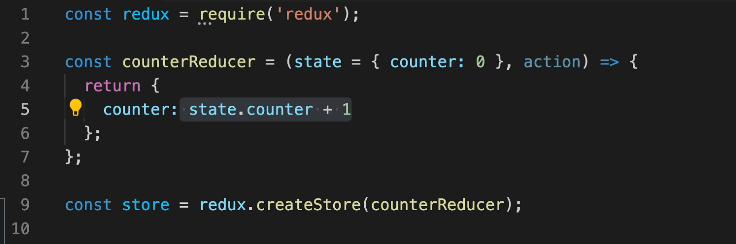
Create new folder react-redux-demo

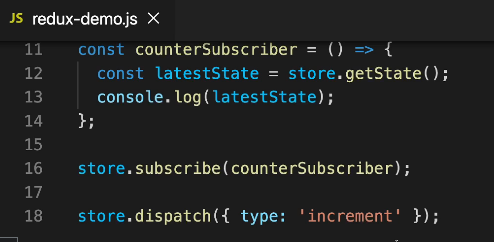
Step 1 run npm --init -y

Step 2 create new file redux-demo.js

Step 3 npm install redux

Step 4 add code in redix-remo.js

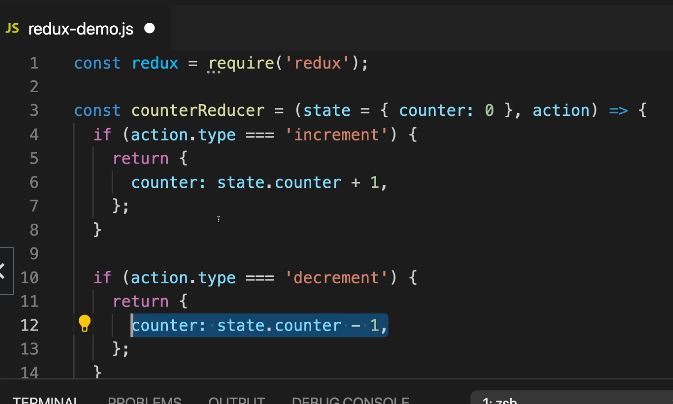




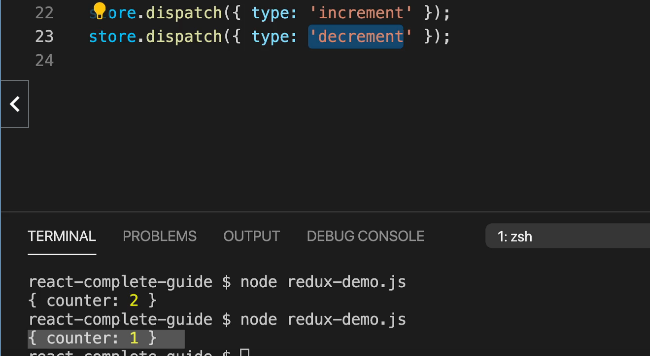
Step 5 then run node redux-demo.js

231. More Redux Basics

Step 1 change redux-demo.js



Then



232. Preparing a new Project

Checkout

<https://github.com/academind/react-complete-guide-code/tree/18-diving-into-redux>

then Clone source 02-providing-the-store

then run command npm install redux react-redux

233. Creating a redux Store for react

Step 1: Create new folder src/store/index.js

import { createStore } from 'redux';

const counterReducer = (state = { counter: 0 }, action) => {

  if (action.type === 'increment') {

    return {

      counter: state.counter + 1,

    };

  }

  if (action.type === 'decrement') {

    return {

      counter: state.counter - 1,

    };

  }

  return state;

};

const store = createStore(counterReducer);

export default store;

234. Providing the store

Step 1: in src/index.js

import React from 'react';

import ReactDOM from 'react-dom/client';

import { Provider } from 'react-redux';

import './index.css';

import App from './App';

import store from './store/index';

const root = ReactDOM.createRoot(document.getElementById('root'));

root.render(

  <Provider store={store}>

    <App />

  </Provider>

);

235. using redux data in react components

Step 1: in Counter.js

import { useSelector } from 'react-redux';

import classes from './Counter.module.css';

const Counter = () => {

  const counter = useSelector(state => state.counter);

  const toggleCounterHandler = () => {};

  return (

    <main className={classes.counter}>

      <h1>Redux Counter</h1>

      <div className={classes.value}>{counter}</div>

      <button onClick={toggleCounterHandler}>Toggle Counter</button>

    </main>

  );

};

export default Counter;

236. Dispatching actions from inside components

Step 1: in Counter.js (add Dispatch)

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

import classes from './Counter.module.css';

const Counter = () => {

  const dispatch = useDispatch();

  const counter = useSelector(state => state.counter);

  const incrementHandler = () => {

    dispatch({ type: 'increment' });

  };

  const decrementHandler = () => {

    dispatch({ type: 'decrement' });

  };

  const toggleCounterHandler = () => {};

  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;

237. redux with class-based components

Step 1: in Counter.js (another use with redux)

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

import classes from './Counter.module.css';

const Counter = () => {

  const dispatch = useDispatch();

  const counter = useSelector(state => state.counter);

  const incrementHandler = () => {

    dispatch({ type: 'increment' });

  };

  const decrementHandler = () => {

    dispatch({ type: 'decrement' });

  };

  const toggleCounterHandler = () => {};

  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;

// class Counter extends Component {

//   incrementHandler() {

//     this.props.increment();

//   }

//   decrementHandler() {

//     this.props.decrement();

//   }

//   toggleCounterHandler() {}

//   render() {

//     return (

//       <main className={classes.counter}>

//         <h1>Redux Counter</h1>

//         <div className={classes.value}>{this.props.counter}</div>

//         <div>

//           <button onClick={this.incrementHandler.bind(this)}>Increment</button>

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

//         </div>

//         <button onClick={this.toggleCounterHandler}>Toggle Counter</button>

//       </main>

//     );

//   }

// }

// const mapStateToProps = state => {

//   return {

//     counter: state.counter

//   };

// }

// const mapDispatchToProps = dispatch => {

//   return {

//     increment: () => dispatch({ type: 'increment' }),

//     decrement: () => dispatch({ type: 'decrement' }),

//   }

// };

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

238. Attaching Payloads To Actions

Step 1: in store/index.js (add increase)

import { createStore } from 'redux';

const counterReducer = (state = { counter: 0 }, action) => {

  if (action.type === 'increment') {

    return {

      counter: state.counter + 1,

    };

  }

  if (action.type === 'increase') {

    return {

      counter: state.counter + action.amount,

    };

  }

  if (action.type === 'decrement') {

    return {

      counter: state.counter - 1,

    };

  }

  return state;

};

const store = createStore(counterReducer);

export default store;

Step 2: Then in Counter.js

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

import classes from './Counter.module.css';

const Counter = () => {

  const dispatch = useDispatch();

  const counter = useSelector((state) => state.counter);

  const incrementHandler = () => {

    dispatch({ type: 'increment' });

  };

  const increaseHandler = () => {

    dispatch({ type: 'increase', amount: 10 });

  };

  const decrementHandler = () => {

    dispatch({ type: 'decrement' });

  };

  const toggleCounterHandler = () => {};

  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 10</button>

        <button onClick={decrementHandler}>Decrement</button>

      </div>

      <button onClick={toggleCounterHandler}>Toggle Counter</button>

    </main>

  );

};

export default Counter;

Step 3: check result



239. Working with Multiple State Properties

Step 1: in store/index.js

import { createStore } from 'redux';

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

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.amount,

      showCounter: state.showCounter

    };

  }

  if (action.type === 'decrement') {

    return {

      counter: state.counter - 1,

      showCounter: state.showCounter

    };

  }

  if (action.type === 'toggle') {

    return {

      showCounter: !state.showCounter,

      counter: state.counter

    };

  }

  return state;

};

const store = createStore(counterReducer);

export default store;

Step 2: in Counter.js

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

import classes from './Counter.module.css';

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 });

  };

  const decrementHandler = () => {

    dispatch({ type: 'decrement' });

  };

  const toggleCounterHandler = () => {

    dispatch({ type: 'toggle' });

  };

  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 10</button>

        <button onClick={decrementHandler}>Decrement</button>

      </div>

      <button onClick={toggleCounterHandler}>Toggle Counter</button>

    </main>

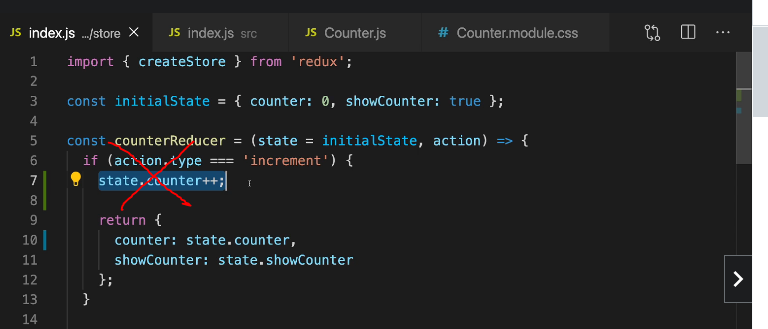
  );

};

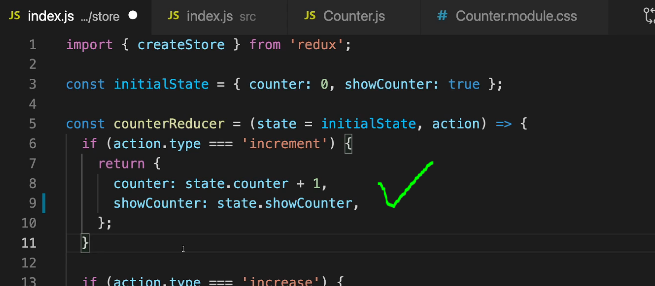
export default Counter;

240. How to Work with redux State Correctly

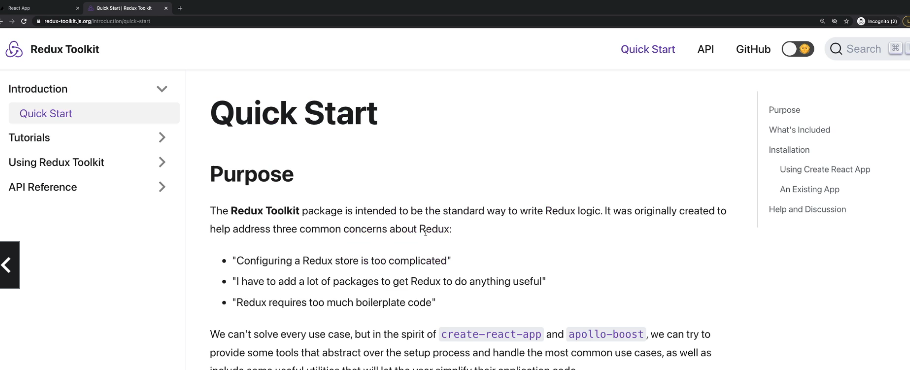
Done use redux like this



Use this



241. Redux Challenges & Introducing redux Toolkit



242 Adding State Slices + Connecting redux toolkit state + Migrating Everything to redux Toolkit

Step 1 run Command



Step 2: in store/index.js

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;

    }

  }

});

// Connecting redux toolkit State

const store = configureStore({

  reducer: counterSlice.reducer

});

//Migrating Everything To Redux Toolkit

export const counterActions = counterSlice.actions;

export default store;

Step 3: in Counter.js (use counterActions in /store/index.js)

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

import { counterActions } from '../store/index';

import classes from './Counter.module.css';

const Counter = () => {

  const dispatch = useDispatch();

  const counter = useSelector((state) => state.counter);

  const show = useSelector((state) => state.showCounter);

  const incrementHandler = () => {

    dispatch(counterActions.increment());

  };

  const increaseHandler = () => {

    dispatch(counterActions.increase(10)); // { type: SOME\_UNIQUE\_IDENTIFIER, payload: 10 }

  };

  const decrementHandler = () => {

    dispatch(counterActions.decrement());

  };

  const toggleCounterHandler = () => {

    dispatch(counterActions.toggleCounter());

  };

  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 10</button>

        <button onClick={decrementHandler}>Decrement</button>

      </div>

      <button onClick={toggleCounterHandler}>Toggle Counter</button>

    </main>

  );

};

export default Counter;

245. Working with Multiple Slices

Step 1: in store.index.js add authentication

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

const initialCounterState = { counter: 0, showCounter: true };

const counterSlice = createSlice({

  name: 'counter',

  initialState: initialCounterState,

  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 initialAuthState = {

  isAuthenticated: false,

};

const authSlice = createSlice({

  name: 'authentication',

  initialState: initialAuthState,

  reducers: {

    login(state) {

      state.isAuthenticated = true;

    },

    logout(state) {

      state.isAuthenticated = false;

    },

  },

});

const store = configureStore({

  reducer: { counter: counterSlice.reducer, auth: authSlice.reducer },

});

export const counterActions = counterSlice.actions;

export const authActions = authSlice.actions;

export default store;

Step 2: in app.js

import { Fragment } from 'react';

import { useSelector } from 'react-redux';

import Counter from './components/Counter';

import Header from './components/Header';

import Auth from './components/Auth';

import UserProfile from './components/UserProfile';

function App() {

  const isAuth = useSelector(state => state.auth.isAuthenticated);

  return (

    <Fragment>

      <Header />

      {!isAuth && <Auth />}

      {isAuth && <UserProfile />}

      <Counter />

    </Fragment>

  );

}

export default App;

Step 3: in Header.js

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

import classes from './Header.module.css';

import { authActions } from '../store/index';

const Header = () => {

  const dispatch = useDispatch();

  const isAuth = useSelector((state) => state.auth.isAuthenticated);

  const logoutHandler = () => {

    dispatch(authActions.logout());

  };

  return (

    <header className={classes.header}>

      <h1>Redux Auth</h1>

      {isAuth && (

        <nav>

          <ul>

            <li>

              <a href='/'>My Products</a>

            </li>

            <li>

              <a href='/'>My Sales</a>

            </li>

            <li>

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

            </li>

          </ul>

        </nav>

      )}

    </header>

  );

};

export default Header;

Step 4: in Auth.js

import { useDispatch } from 'react-redux';

import classes from './Auth.module.css';

import { authActions } from '../store/index';

const Auth = () => {

  const dispatch = useDispatch();

  const loginHandler = (event) => {

    event.preventDefault();

    dispatch(authActions.login());

  };

  return (

    <main className={classes.auth}>

      <section>

        <form onSubmit={loginHandler}>

          <div className={classes.control}>

            <label htmlFor='email'>Email</label>

            <input type='email' id='email' />

          </div>

          <div className={classes.control}>

            <label htmlFor='password'>Password</label>

            <input type='password' id='password' />

          </div>

          <button>Login</button>

        </form>

      </section>

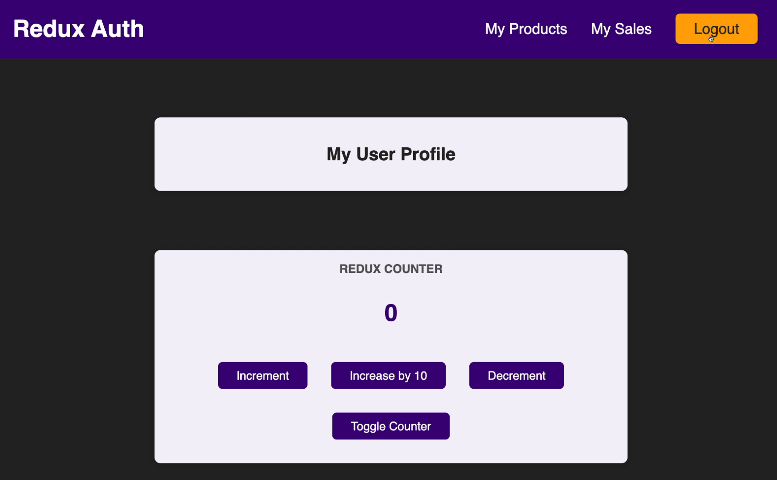
    </main>

  );

};

export default Auth;

Step 5 Check result



237. Splitting our Code

split code in store/index.js

Step 1: create store/auth.js

import { createSlice } from '@reduxjs/toolkit';

const initialAuthState = {

  isAuthenticated: false,

};

const authSlice = createSlice({

  name: 'authentication',

  initialState: initialAuthState,

  reducers: {

    login(state) {

      state.isAuthenticated = true;

    },

    logout(state) {

      state.isAuthenticated = false;

    },

  },

});

export const authActions = authSlice.actions;

export default authSlice.reducer;

Step 2: create store/counter.js

import { createSlice } from '@reduxjs/toolkit';

const initialCounterState = { counter: 0, showCounter: true };

const counterSlice = createSlice({

  name: 'counter',

  initialState: initialCounterState,

  reducers: {

    increment(state) {

      state.counter++;

    },

    decrement(state) {

      state.counter--;

    },

    increase(state, action) {

      state.counter = state.counter + action.payload;

    },

    toggleCounter(state) {

      state.showCounter = !state.showCounter;

    },

  },

});

export const counterActions = counterSlice.actions;

export default counterSlice.reducer;

Step 3: create store/index.js

import { configureStore } from '@reduxjs/toolkit';

import counterReducer from './counter';

import authReducer from './auth';

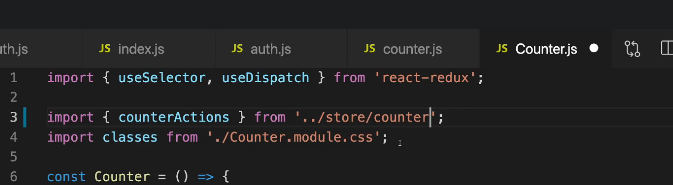
const store = configureStore({

  reducer: { counter: counterReducer, auth: authReducer },

});

export default store;

Step 4: in Counter.js (change import like this)



Also in auth.js

Also in Header.js

Step 5: check result

249 Module Resources

You may want to **compare your code to mine** (e.g. to find + fix errors).

For that, you find **multiple code snapshots** for this module here in this Github repository: <https://github.com/academind/react-complete-guide-code/tree/18-diving-into-redux>