**🔄 Why Redux?**

Redux helps manage state in large React applications. Think of it as a central store for all your app's data.

**🤔 When to Use Redux?**

* When multiple components need the same data
* When state updates are complex
* When you need to track state changes
* When different parts of your app modify the same data

**🛠️ Basic Setup**

**📦 1. Installation**

npm install redux react-redux

**📂 2. Basic Folder Structure**

src/

├── components/

├── redux/

│ ├── actions/

│ │ └── todoActions.js

│ ├── reducers/

│ │ └── todoReducer.js

│ └── store.js

└── App.js

**🏪 Creating a Store**

**📄 store.js**

import { createStore } from 'redux';

import todoReducer from './reducers/todoReducer';

const store = createStore(

todoReducer,

window.\_\_REDUX\_DEVTOOLS\_EXTENSION\_\_ && window.\_\_REDUX\_DEVTOOLS\_EXTENSION\_\_()

);

export default store;

**🔄 Creating a Reducer**

**📄 todoReducer.js**

const initialState = {

todos: []

};

function todoReducer(state = initialState, action) {

switch (action.type) {

case 'ADD\_TODO':

return {

...state,

todos: [...state.todos, action.payload]

};

case 'REMOVE\_TODO':

return {

...state,

todos: state.todos.filter(todo => todo.id !== action.payload)

};

default:

return state;

}

}

export default todoReducer;

**🎬 Creating Actions**

**📄 todoActions.js**

// Action Creators

export const addTodo = (todo) => {

return {

type: 'ADD\_TODO',

payload: todo

};

};

export const removeTodo = (id) => {

return {

type: 'REMOVE\_TODO',

payload: id

};

};

**🔗 Connecting Redux to React**

**📄 index.js**

import { Provider } from 'react-redux';

import store from './redux/store';

ReactDOM.render(

<Provider store={store}>

<App />

</Provider>,

document.getElementById('root')

);

**📝 Simple Todo App Example**

**📄 TodoList.js**

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

import { addTodo, removeTodo } from '../redux/actions/todoActions';

function TodoList() {

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

const dispatch = useDispatch();

const handleAddTodo = (text) => {

const newTodo = {

id: Date.now(),

text: text,

completed: false

};

dispatch(addTodo(newTodo));

};

return (

<div>

<button onClick={() => handleAddTodo("New Todo")}>

Add Todo

</button>

<ul>

{todos.map(todo => (

<li key={todo.id}>

{todo.text}

<button onClick={() => dispatch(removeTodo(todo.id))}>

Delete

</button>

</li>

))}

</ul>

</div>

);

}

**🪝 Using useSelector and useDispatch**

**📄 Example Component**

function Counter() {

// Get data from store

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

// Get dispatch function

const dispatch = useDispatch();

return (

<div>

<h1>Count: {count}</h1>

<button onClick={() => dispatch({ type: 'INCREMENT' })}>

Increment

</button>

</div>

);

}

**🛠️ Redux DevTools**

* Install Redux DevTools extension in your browser
* Add configuration to store:

const store = createStore(

reducer,

window.\_\_REDUX\_DEVTOOLS\_EXTENSION\_\_ && window.\_\_REDUX\_DEVTOOLS\_EXTENSION\_\_()

);

**📝 Complete Todo App Example**

**📄 redux/reducers/todoReducer.js**

const initialState = {

todos: []

};

function todoReducer(state = initialState, action) {

switch (action.type) {

case 'ADD\_TODO':

return {

...state,

todos: [...state.todos, action.payload]

};

case 'REMOVE\_TODO':

return {

...state,

todos: state.todos.filter(todo => todo.id !== action.payload)

};

default:

return state;

}

}

**📄 components/TodoApp.js**

function TodoApp() {

const [inputText, setInputText] = useState('');

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

const dispatch = useDispatch();

const handleSubmit = (e) => {

e.preventDefault();

if (inputText.trim()) {

dispatch(addTodo({

id: Date.now(),

text: inputText

}));

setInputText('');

}

};

return (

<div>

<form onSubmit={handleSubmit}>

<input

value={inputText}

onChange={(e) => setInputText(e.target.value)}

placeholder="Add new todo"

/>

<button type="submit">Add</button>

</form>

<ul>

{todos.map(todo => (

<li key={todo.id}>

{todo.text}

<button onClick={() => dispatch(removeTodo(todo.id))}>

Delete

</button>

</li>

))}

</ul>

</div>

);

}

**🗝️ Key Points to Remember**

**🔄 Redux Flow**

Action → Reducer → Store → Component

**📋 Basic Rules**

* Single source of truth (one store)
* State is read-only
* Changes made through pure functions (reducers)

**🪝 Common Hooks**

* useSelector: Get data from store
* useDispatch: Dispatch actions

**✅ Best Practices**

* Keep reducers pure
* Use action creators
* Organize files by feature

**💡 Remember:**

* Start with simple state management
* Add Redux when needed
* Use Redux DevTools for debugging
* Keep actions and reducers simple
* Follow the Redux flow

🚀 Happy coding with Redux!