

Table Of Contents

- Foreword** **ii**
- 0.1 Acknowledgements ii
- 0.2 About the author iv
- 0.3 Contact iv

- 1 Introduction** **1**
- 1.1 About this book 1
- 1.2 Who is this book for? 2
- 1.3 How to follow this book 3
 - 1.3.1 Code examples 4
- 1.4 Pre-requisites 4

- 2 Project Configuration and Useful Extensions** **6**
- 2.1 Setting up a React project 6
- 2.2 Plugins configuration 7
 - 2.2.1 PostCSS 9
 - 2.2.2 Stylelint 10
 - 2.2.3 Configuring Sass 11
 - 2.2.4 Tailwind 11
 - 2.2.5 Prettier 12
 - 2.2.6 TypeScript 13
 - 2.2.7 Jest, Cypress, and Testing Library 15
 - 2.2.7.1 Cypress & Testing Library 15
 - 2.2.7.2 Jest & Testing Library 17
 - 2.2.8 Formatting Code Automatically on Commit 17
 - 2.2.9 What about Vite? 18
- 2.3 VS Code extensions 18
- 2.4 Summary 22

- 3 Scalable and Maintainable Project Architecture** **24**
- 3.1 Managing route components by feature 28
- 3.2 Encapsulating components and business logic 30
- 3.3 Summary 32

- 4 API Layer and Managing Async Operations** **34**
- 4.1 Implementing an API layer 35
- 4.2 Handling API states 43
 - 4.2.1 How to avoid flickering loader 52
- 4.3 Abstracting API states and fetching with the useApi hook 53
- 4.4 Request cancellation 57
 - 4.4.1 Enhancing the API layer to add abort logic 58

4.5	Error logging	64
4.6	Summary	66
5	Managing APIs with API Layer and React-Query	68
5.1	How to fetch data with React-Query	69
5.2	How to update data with React-Query	72
5.3	Pagination with React-Query	77
5.4	Infinite scroll with React-Query	80
5.5	Query cancellation with the API layer and React-Query	84
5.5.1	Cancel Function	86
5.5.2	Abort Signal	90
5.6	Summary	93
6	State Management Patterns in React Apps	95
6.1	Sharing state between sibling components by lifting state up	95
6.2	Context State Provider	103
6.2.1	How to improve performance and avoid unnecessary re-renders with Context API .	109
6.2.1.1	Wrap JSX inside of the <i>useMemo</i> hook	111
6.2.1.2	Extract JSX into its own component wrapped with <i>memo</i>	111
6.2.1.3	Split the Context and use two separate context providers	112
6.2.1.4	Use the <i>useContextSelector</i> library.	116
6.3	Better state handling with <i>useImmer</i> and <i>useReducer</i> hooks	120
6.3.1	Immutable updates with <i>useImmer</i>	121
6.3.2	<i>useReducer</i>	126
6.3.3	Cleaner reducer with <i>useImmerReducer</i>	133
6.3.4	Reducer with Context API	135
6.4	Summary	139
7	Modern Redux - Global State Management with Redux Toolkit	141
7.1	Redux of the past	141
7.2	Modern Redux with Redux Toolkit (RTK)	143
7.2.1	API Requests with Redux Toolkit	151
7.2.2	API requests with RTK's <i>createAsyncThunk</i> and API layer	151
7.2.2.1	Fetching Users	151
7.2.2.2	Adding and Deleting Users	157
7.2.3	State Normalisation	162
7.2.3.1	Normalising State with <i>createEntityAdapter</i>	165
7.2.4	Persisting Redux Store with RTK and <i>Redux-Persist</i>	171
7.2.5	Resetting Slices State and Redux Store	174
7.2.5.1	Resetting Users Slice	175
7.2.5.2	Resetting Whole Redux Store	177
7.2.6	API management with Redux & RTK Query	181
7.2.6.1	Integrating RTK Query with API Layer	192
7.2.7	Optimistic Updates	194
7.3	Summary	196
8	Global State Management with Zustand and Jotai	198
8.1	Zustand	198
8.1.1	Events Manager with Zustand	199
8.1.2	Computing derived state in selectors	209
8.1.3	Computing derived state with the <i>useMemo</i> hook	213
8.1.4	Computing derived state with the <i>useEffect</i> hook	214

8.1.5	Computing derived state with subscriptions	215
8.1.6	Simplifying selectors with a Pick helper	219
8.1.7	Simplifying Zustand state updates with Immer	220
8.1.8	Simplifying store creation with factory helpers	223
8.1.9	Persisting Zustand store	225
8.1.10	Async operations	228
8.1.11	Zustand with React-Query	229
8.2	Jotai	238
8.2.1	Atoms	239
8.2.2	Converting Events Manager to use atoms and how to derive atom state	240
8.2.3	Jotai with Immer	247
8.2.4	Persisting Atom State with atomWithStorage	248
8.2.5	Combining atomWithStorage and Immer	248
8.2.6	Async Requests with Jotai	249
8.2.7	Jotai with React-Query	250
8.3	Summary	257
9	Advanced Component Patterns	259
9.1	Higher Order Components	259
9.2	Render Props	262
9.3	Wrapper components	265
9.4	Polymorphic components	268
9.5	Composition vs Configuration - how to build flexible, maintainable and reusable components	272
9.6	Observer Pattern - communicating between sibling components	290
9.7	Summary	294
10	Managing Application Layouts	296
10.1	Route Layouts with React Router	296
10.2	Product Grid and List layouts with the useLayout hook	306
10.3	Summary	315
11	Performance Optimisation	317
11.1	Code-Splitting and Lazy-Loading routes and components with React.Lazy and React.Suspense	317
11.2	How to optimise and prevent re-renders of React components	322
11.2.1	Optimising component re-renders with memo and useCallback	328
11.2.2	Avoiding re-renders with useMemo	330
11.2.3	Reducing the number of re-renders by moving state down (state colocation)	331
11.2.4	Preventing re-renders by lifting components up	333
11.3	Optimising long lists by virtualising with React Virtual	335
11.4	Throttle and Debounce events to prevent frequent re-renders	338
11.4.1	Throttling mousemove events	339
11.4.2	Debouncing search requests	341
11.5	Tree-shaking	343
11.5.1	First example - Lodash	344
11.5.2	Second example - React Icons	344
11.5.3	Third example - UI frameworks	345
11.6	Choosing appropriate libraries and tree-shaking	345
11.6.1	What to look at when choosing libraries and do I even need one?	347
11.7	Reducing bundle CSS by removing unused CSS	348
11.8	Production Build	349
11.9	Summary	350

12 Application Security	352
12.1 Validate URLs	352
12.2 Rendering HTML	354
12.3 Third-party libraries	354
12.4 JSON Web Tokens (JWT)	355
12.5 Access permissions	357
12.5.1 Restricting access to specific content using the Permission component	366
12.5.2 Restricting access to private routes with the Permission component	370
12.6 Summary	372
13 React Testing - Best Practices For Writing Future-Proof Tests	374
13.1 Unit testing React components	375
13.2 How to test standalone React hooks	381
13.3 API mocking in React Unit Tests	384
13.4 End-to-end testing with Cypress	390
13.5 Useful testing tips	398
13.6 Summary	399
14 Static Site Generation (SSG), Incremental Site Regeneration (ISR) and Server Side Rendering (SSR) with Next.js	401
14.1 Next.js Project Setup	402
14.2 Pages and Routing in Next.js apps	403
14.3 Static Site Generation (SSG) with <code>getStaticProps</code> and <code>getStaticPaths</code>	406
14.4 Incremental Site Regeneration (ISR)	414
14.5 Server Side Rendering (SSR) with <code>getServerSideProps</code>	417
14.6 Running code server-side with API Routes	419
14.7 Restricting Access to Specific Pages with Middleware	421
14.8 Summary	424