

React

The Road To Enterprise

JavaScript Edition



Learn how to create Production-Ready applications with React
Best Practices, Advanced Patterns, Guides, Tricks, and more...

THOMAS FINDLAY



THE ROAD
TO ENTERPRISE

React - The Road To Enterprise

Become Production-Ready with Advanced Patterns & Best Practices for Scalable React Applications

Thomas Findlay

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 PropTypes	8
2.2.2 PostCSS	9
2.2.3 Stylelint	9
2.2.4 Configuring Sass	11
2.2.5 Tailwind	12
2.2.6 Prettier	12
2.2.7 ESLint	13
2.2.8 Vite config	14
2.2.9 Vitest, Cypress, and Testing Library	16
2.2.9.1 Cypress & Testing Library	16
2.2.9.2 Vitest & Testing Library	17
2.2.10 Formatting and Analysing Code on Commit with Husky and Lint-Staged	19
2.3 VS Code extensions	19
2.4 Summary	22
3 Scalable and Maintainable Project Architecture	25
3.1 Managing route components by feature	29
3.2 Encapsulating components and business logic	31
3.3 Summary	33
4 API Layer and Managing Async Operations	35
4.1 Implementing an API layer	36
4.2 Handling API states	43
4.2.1 How to avoid flickering loader	51
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	63
4.6	Summary	65
5	Managing APIs with API Layer and React-Query	67
5.1	How to fetch data with React-Query	68
5.2	How to update data with React-Query	70
5.3	Pagination with React-Query	76
5.4	Infinite scroll with React-Query	79
5.5	Query cancellation with the API layer and React-Query	83
5.6	Summary	88
6	State Management Patterns in React Apps	90
6.1	Sharing state between sibling components by lifting state up	90
6.2	Context State Provider	98
6.2.1	How to improve performance and avoid unnecessary re-renders with Context API .	103
6.2.1.1	Wrap JSX inside of the <i>useMemo</i> hook	105
6.2.1.2	Extract JSX into its own component wrapped with <i>memo</i>	106
6.2.1.3	Split the Context and use two separate context providers	107
6.2.1.4	Use the <i>useContextSelector</i> library.	110
6.3	Better state handling with <i>useImmer</i> and <i>useReducer</i> hooks	114
6.3.1	Immutable updates with <i>useImmer</i>	115
6.3.2	<i>useReducer</i>	120
6.3.3	Cleaner reducer with <i>useImmerReducer</i>	126
6.3.4	Reducer with Context API	127
6.4	Summary	131
7	Modern Redux - Global State Management with Redux Toolkit	133
7.1	Redux of the past	133
7.2	Modern Redux with Redux Toolkit (RTK)	135
7.2.1	API Requests with Redux Toolkit	142
7.2.2	API requests with RTK's <i>createAsyncThunk</i> and API layer	142
7.2.2.1	Fetching Users	143
7.2.2.2	Adding and Deleting Users	148
7.2.3	State Normalisation	152
7.2.3.1	Normalising State with <i>createEntityAdapter</i>	155
7.2.4	Persisting Redux Store with RTK and <i>Redux-Persist</i>	160
7.2.5	Resetting Slices State and Redux Store	164
7.2.5.1	Resetting Users Slice	165
7.2.5.2	Resetting Whole Redux Store	167
7.2.6	API management with Redux & RTK Query	171
7.2.6.1	Integrating RTK Query with API Layer	181
7.2.7	Optimistic Updates	183
7.3	Summary	185
8	Global State Management with Zustand and Jotai	187
8.1	Zustand	187
8.1.1	Events Manager with Zustand	188
8.1.2	Computing derived state in selectors	196
8.1.3	Computing derived state with the <i>useMemo</i> hook	201
8.1.4	Computing derived state with the <i>useEffect</i> hook	202
8.1.5	Computing derived state with subscriptions	203

8.1.6	Simplifying selectors with a Pick helper	206
8.1.7	Simplifying Zustand state updates with Immer	207
8.1.8	Simplifying store creation with factory helpers	208
8.1.9	Persisting Zustand store	210
8.1.10	Async operations	213
8.1.11	Zustand with React-Query	214
8.2	Jotai	222
8.2.1	Atoms	222
8.2.2	Converting Events Manager to use atoms and how to derive atom state	223
8.2.3	Jotai with Immer	230
8.2.4	Persisting Atom State with atomWithStorage	231
8.2.5	Combining atomWithStorage and Immer	231
8.2.6	Async Requests with Jotai	232
8.2.7	Jotai with React-Query	233
8.3	Summary	239
9	Advanced Component Patterns	241
9.1	Higher Order Components	241
9.2	Render Props	244
9.3	Wrapper components	246
9.4	Polymorphic components	249
9.5	Composition vs Configuration - how to build flexible, maintainable and reusable components	252
9.6	Observer Pattern - communicating between sibling components	270
9.7	Summary	273
10	Managing Application Layouts	275
10.1	Route Layouts with React Router	275
10.2	Product Grid and List layouts with the useLayout hook	285
10.3	Summary	293
11	Performance Optimisation	295
11.1	Code-Splitting and Lazy-Loading routes and components with React.Lazy and React.Suspense	295
11.2	How to optimise and prevent re-renders of React components	300
11.2.1	Optimising component re-renders with memo and useCallback	306
11.2.2	Avoiding re-renders with useMemo	308
11.2.3	Reducing the number of re-renders by moving state down (State Colocation)	309
11.2.4	Preventing re-renders by lifting components up	311
11.3	Optimising long lists by virtualising with React Virtual	313
11.4	Throttle and Debounce events to prevent frequent re-renders	316
11.4.1	Throttling mousemove events	317
11.4.2	Debouncing search requests	319
11.5	Tree-shaking	320
11.5.1	First example - Lodash	321
11.5.2	Second example - React Icons	321
11.5.3	Third example - UI frameworks	322
11.6	Choosing appropriate libraries and tree-shaking	322
11.6.1	What to look at when choosing libraries and do I even need one?	324
11.7	Reducing bundle CSS by removing unused CSS	325
11.8	Production Build	326
11.9	Summary	327

12 Application Security	329
12.1 Validate URLs	329
12.2 Rendering HTML	331
12.3 Third-party libraries	331
12.4 JSON Web Tokens (JWT)	332
12.5 Access permissions	334
12.5.1 Restricting access to specific content using the Permission component	342
12.5.2 Restricting access to private routes with the Permission component	345
12.6 Summary	348
13 React Testing - Best Practices For Writing Future-Proof Tests	350
13.1 Unit testing React components	351
13.2 How to test standalone React hooks	356
13.3 API mocking in React Unit Tests	359
13.4 End-to-end testing with Cypress	365
13.5 Useful testing tips	372
13.6 Summary	373
14 Static Site Generation (SSG), Incremental Site Regeneration (ISR) and Server Side Rendering (SSR) with Next.js using Pages and App Router	375
14.1 Pages - Routing	376
14.2 Static Site Generation (SSG) with getStaticProps and getStaticPaths	378
14.3 Pages - Incremental Site Regeneration (ISR)	384
14.4 Pages - Server Side Rendering (SSR) with getServerSideProps	386
14.5 Pages - Running code server-side with API Routes	388
14.6 Pages - Restricting Access to Specific Pages with Middleware	390
14.7 App Router - Routing	392
14.8 App Router - Static Site Generation (SSG)	394
14.9 App Router - Incremental Site Regeneration (ISR)	397
14.10 App Router - Server Side Rendering (SSR)	398
14.11 App Router - Running code server-side with API Routes	399
14.12 App Router - Restricting Access to Specific Pages with Middleware	401
14.13 Summary	403