

React

The Road To Enterprise

TypeScript 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	7
2.2 Plugins configuration	8
2.2.1 PostCSS	9
2.2.2 Stylelint	10
2.2.3 Configuring Sass	12
2.2.4 Tailwind	12
2.2.5 Prettier	13
2.2.6 ESLint	14
2.2.7 TypeScript and Vite config	15
2.2.8 Vitest, Cypress, and Testing Library	17
2.2.8.1 Cypress & Testing Library	17
2.2.8.2 Vitest & Testing Library	19
2.2.9 Formatting and Analysing Code on Commit with Husky and Lint-Staged	20
2.3 VS Code extensions	21
2.4 Summary	24
3 Scalable and Maintainable Project Architecture	27
3.1 Managing route components by feature	31
3.2 Encapsulating components and business logic	33
3.3 Summary	35
4 API Layer and Managing Async Operations	37
4.1 Implementing an API layer	38
4.2 Handling API states	46
4.2.1 How to avoid flickering loader	55
4.3 Abstracting API states and fetching with the useApi hook	56
4.4 Request cancellation	60
4.4.1 Enhancing the API layer to add abort logic	61

4.5	Error logging	67
4.6	Summary	69
5	Managing APIs with API Layer and React-Query	71
5.1	How to fetch data with React-Query	72
5.2	How to update data with React-Query	75
5.3	Pagination with React-Query	80
5.4	Infinite scroll with React-Query	83
5.5	Query cancellation with the API layer and React-Query	87
5.5.1	Cancel Function	89
5.5.2	Abort Signal	93
5.6	Summary	96
6	State Management Patterns in React Apps	98
6.1	Sharing state between sibling components by lifting state up	98
6.2	Context State Provider	106
6.2.1	How to improve performance and avoid unnecessary re-renders with Context API .	112
6.2.1.1	Wrap JSX inside of the <i>useMemo</i> hook	114
6.2.1.2	Extract JSX into its own component wrapped with <i>memo</i>	114
6.2.1.3	Split the Context and use two separate context providers	116
6.2.1.4	Use the <i>useContextSelector</i> library.	119
6.3	Better state handling with <i>useImmer</i> and <i>useReducer</i> hooks	124
6.3.1	Immutable updates with <i>useImmer</i>	125
6.3.2	<i>useReducer</i>	130
6.3.3	Cleaner reducer with <i>useImmerReducer</i>	137
6.3.4	Reducer with Context API	139
6.4	Summary	142
7	Modern Redux - Global State Management with Redux Toolkit	144
7.1	Redux of the past	144
7.2	Modern Redux with Redux Toolkit (RTK)	146
7.2.1	API Requests with Redux Toolkit	154
7.2.2	API requests with RTK's <i>createAsyncThunk</i> and API layer	154
7.2.2.1	Fetching Users	154
7.2.2.2	Adding and Deleting Users	160
7.2.3	State Normalisation	165
7.2.3.1	Normalising State with <i>createEntityAdapter</i>	168
7.2.4	Persisting Redux Store with RTK and <i>Redux-Persist</i>	174
7.2.5	Resetting Slices State and Redux Store	177
7.2.5.1	Resetting Users Slice	178
7.2.5.2	Resetting Whole Redux Store	180
7.2.6	API management with Redux & RTK Query	184
7.2.6.1	Integrating RTK Query with API Layer	195
7.2.7	Optimistic Updates	197
7.3	Summary	199
8	Global State Management with Zustand and Jotai	201
8.1	Zustand	201
8.1.1	Events Manager with Zustand	202
8.1.2	Computing derived state in selectors	212
8.1.3	Computing derived state with the <i>useMemo</i> hook	216
8.1.4	Computing derived state with the <i>useEffect</i> hook	217

8.1.5	Computing derived state with subscriptions	218
8.1.6	Simplifying selectors with a Pick helper	222
8.1.7	Simplifying Zustand state updates with Immer	223
8.1.8	Simplifying store creation with factory helpers	226
8.1.9	Persisting Zustand store	228
8.1.10	Async operations	232
8.1.11	Zustand with React-Query	232
8.2	Jotai	241
8.2.1	Atoms	242
8.2.2	Converting Events Manager to use atoms and how to derive atom state	243
8.2.3	Jotai with Immer	250
8.2.4	Persisting Atom State with atomWithStorage	251
8.2.5	Combining atomWithStorage and Immer	251
8.2.6	Async Requests with Jotai	252
8.2.7	Jotai with React-Query	253
8.3	Summary	260
9	Advanced Component Patterns	262
9.1	Higher Order Components	262
9.2	Render Props	265
9.3	Wrapper components	268
9.4	Polymorphic components	271
9.5	Composition vs Configuration - how to build flexible, maintainable and reusable components	275
9.6	Observer Pattern - communicating between sibling components	293
9.7	Summary	297
10	Managing Application Layouts	299
10.1	Route Layouts with React Router	299
10.2	Product Grid and List layouts with the useLayout hook	309
10.3	Summary	318
11	Performance Optimisation	320
11.1	Code-Splitting and Lazy-Loading routes and components with React.Lazy and React.Suspense	320
11.2	How to optimise and prevent re-renders of React components	325
11.2.1	Optimising component re-renders with memo and useCallback	331
11.2.2	Avoiding re-renders with useMemo	333
11.2.3	Reducing the number of re-renders by moving state down (state colocation)	334
11.2.4	Preventing re-renders by lifting components up	336
11.3	Optimising long lists by virtualising with React Virtual	338
11.4	Throttle and Debounce events to prevent frequent re-renders	341
11.4.1	Throttling mousemove events	342
11.4.2	Debouncing search requests	344
11.5	Tree-shaking	346
11.5.1	First example - Lodash	347
11.5.2	Second example - React Icons	347
11.5.3	Third example - UI frameworks	348
11.6	Choosing appropriate libraries and tree-shaking	348
11.6.1	What to look at when choosing libraries and do I even need one?	350
11.7	Reducing bundle CSS by removing unused CSS	351
11.8	Production Build	352
11.9	Summary	353

12 Application Security	355
12.1 Validate URLs	355
12.2 Rendering HTML	357
12.3 Third-party libraries	357
12.4 JSON Web Tokens (JWT)	358
12.5 Access permissions	360
12.5.1 Restricting access to specific content using the Permission component	369
12.5.2 Restricting access to private routes with the Permission component	373
12.6 Summary	375
13 React Testing - Best Practices For Writing Future-Proof Tests	377
13.1 Unit testing React components	378
13.2 How to test standalone React hooks	383
13.3 API mocking in React Unit Tests	386
13.4 End-to-end testing with Cypress	392
13.5 Useful testing tips	399
13.6 Summary	400
14 Static Site Generation (SSG), Incremental Site Regeneration (ISR) and Server Side Rendering (SSR) with Next.js using Pages and App Router	402
14.1 Pages - Routing	403
14.2 Static Site Generation (SSG) with getStaticProps and getStaticPaths	405
14.3 Pages - Incremental Site Regeneration (ISR)	411
14.4 Pages - Server Side Rendering (SSR) with getServerSideProps	413
14.5 Pages - Running code server-side with API Routes	415
14.6 Pages - Restricting Access to Specific Pages with Middleware	417
14.7 App Router - Routing	419
14.8 App Router - Static Site Generation (SSG)	421
14.9 App Router - Incremental Site Regeneration (ISR)	424
14.10 App Router - Server Side Rendering (SSR)	425
14.11 App Router - Running code server-side with API Routes	426
14.12 App Router - Restricting Access to Specific Pages with Middleware	428
14.13 Summary	430