

Contents

Book Revision	1
Updated January 13th, 2020	1
Join our Official Community Discord	1
Bug Reports	1
Be notified of updates via Twitter	1
We'd love to hear from you!	1
Foreword	2
How to Get the Most Out of This Book	1
Overview	1
Running Code Examples	2
Project setups	3
Code Blocks and Context	3
Code Block Numbering	4
Getting Help	5
Emailing Us	5
Get excited	6
Part I	7
Your first React Web Application	8
Building Product Hunt	8
Setting up your development environment	9
Code editor	9
Node.js and npm	10
Install Git	10

CONTENTS

Browser	10
Special instruction for Windows users	11
Ensure IIS is installed	11
JavaScript ES6/ES7	11
Getting started	12
Sample Code	12
Previewing the application	12
Prepare the app	15
What's a component?	19
Our first component	20
JSX	22
The developer console	24
Babel	26
ReactDOM.render()	28
Building Product	30
Making Product data-driven	33
The data model	33
Using props	34
Rendering multiple products	39
React the vote (your app's first interaction)	44
Propagating the event	45
Binding custom component methods	48
Using state	52
Setting state with this.setState()	54
Updating state and immutability	56
Refactoring with the Babel plugin transform-class-properties	63
Babel plugins and presets	63
Property initializers	65
Refactoring Product	65
Refactoring ProductList	67
Congratulations!	69
Components	70
A time-logging app	70
Getting started	71
Previewing the app	71

CONTENTS

Prepare the app	72
Breaking the app into components	77
The steps for building React apps from scratch	85
Step 2: Build a static version of the app	87
TimersDashboard	87
EditableTimer	89
TimerForm	90
ToggleableTimerForm	92
Timer	92
Render the app	94
Try it out	95
Step 3: Determine what should be stateful	96
State criteria	96
Applying the criteria	97
Step 4: Determine in which component each piece of state should live . .	98
The list of timers and properties of each timer	99
Whether or not the edit form of a timer is open	99
Visibility of the create form	100
Step 5: Hard-code initial states	100
Adding state to TimersDashboard	100
Receiving props in EditableTimerList	102
Props vs. state	103
Adding state to EditableTimer	104
Timer remains stateless	105
Adding state to ToggleableTimerForm	105
Adding state to TimerForm	107
Step 6: Add inverse data flow	111
TimerForm	111
ToggleableTimerForm	113
TimersDashboard	115
Updating timers	117
Adding editability to Timer	118
Updating EditableTimer	118
Updating EditableTimerList	120
Defining onEditFormSubmit() in TimersDashboard	121
Deleting timers	124

CONTENTS

Adding the event handler to <code>Timer</code>	124
Routing through <code>EditableTimer</code>	125
Routing through <code>EditableTimerList</code>	126
Implementing the delete function in <code>TimersDashboard</code>	127
Adding timing functionality	128
Adding a <code>forceUpdate()</code> interval to <code>Timer</code>	129
Try it out	131
Add start and stop functionality	131
Add timer action events to <code>Timer</code>	131
Create <code>TimerActionButton</code>	133
Run the events through <code>EditableTimer</code> and <code>EditableTimerList</code> . . .	134
Try it out	137
Methodology review	138
Components & Servers	140
Introduction	140
Preparation	140
<code>server.js</code>	141
The Server API	141
text/html endpoint	143
JSON endpoints	143
Playing with the API	144
Loading state from the server	148
Try it out	151
<code>client</code>	152
Fetch	152
Sending starts and stops to the server	156
Sending creates, updates, and deletes to the server	159
Give it a spin	161
Next up	161
JSX and the Virtual DOM	162
React Uses a Virtual DOM	162
Why Not Modify the Actual DOM?	162
What is a Virtual DOM?	163
Virtual DOM Pieces	163

CONTENTS

ReactElement	164
Experimenting with ReactElement	165
Rendering Our ReactElement	167
Adding Text (with children)	169
ReactDOM.render()	171
JSX	172
JSX Creates Elements	172
JSX Attribute Expressions	174
JSX Conditional Child Expressions	175
JSX Boolean Attributes	175
JSX Comments	176
JSX Spread Syntax	176
JSX Gotchas	177
JSX Summary	181
References	182
Advanced Component Configuration with props, state, and children . . .	183
Intro	183
How to use this chapter	184
Components	185
Creating Components - ES6 Classes or Functional Components	185
render() Returns a ReactElement Tree	186
Getting Data into render()	187
props are the parameters	188
PropTypes	189
Default props with getDefaultProps()	191
Context	191
Default value	195
Multiple contexts	196
state	197
Using state: Building a Custom Radio Button	197
Stateful components	203
State updates that depend on the current state	206
Thinking About State	207
Stateless Components	209
Switching to Stateless	210

CONTENTS

Stateless Encourages Reuse	213
Talking to Children Components with <code>props.children</code>	213
<code>React.Children.map()</code> & <code>React.Children.forEach()</code>	216
<code>React.Children.toArray()</code>	217
Summary	218
References	219
Forms	220
Forms 101	220
Preparation	221
The Basic Button	222
Events and Event Handlers	224
Back to the Button	225
Text Input	227
Accessing User Input With <code>refs</code>	228
Using User Input	230
Uncontrolled vs. Controlled Components	234
Accessing User Input With <code>state</code>	235
Multiple Fields	238
On Validation	243
Adding Validation to Our App	244
Creating the Field Component	249
Using our new Field Component	253
Remote Data	259
Building the Custom Component	261
Adding <code>CourseSelect</code>	267
Separation of View and State	271
Async Persistence	271
Redux	280
Form Component	287
Connect the Store	292
Form Modules	294
<code>formsy-react</code>	294
<code>react-input-enhancements</code>	295
<code>tcomb-form</code>	295
<code>winterfell</code>	295

CONTENTS

react-redux-form	296
Using Webpack with Create React App	297
JavaScript modules	298
Create React App	300
Exploring Create React App	301
public/index.html	302
package.json	303
src/	306
index.js	308
Bootling the app	310
Webpack basics	312
Making modifications to the sample app	320
Hot reloading	320
Auto-reloading	322
Creating a production build	323
Ejecting	327
Buckle up	328
Using Create React App with an API server	330
The completed app	330
How the app is organized	335
The server	336
Client	338
Concurrently	338
Using the Webpack development proxy	343
Webpack at large	345
When to use Webpack/Create React App	345
Unit Testing	347
Writing tests without a framework	347
Preparing Modash	348
Writing the first spec	352
The <code>assertEqual()</code> function	354
What is Jest?	359
Using Jest	360
<code>expect()</code>	360

CONTENTS

The first Jest test for Modash	363
The other truncate() spec	365
The rest of the specs	366
Testing strategies for React applications	368
Integration vs Unit Testing	368
Shallow rendering	369
Enzyme	370
Testing a basic React component with Enzyme	371
Setup	371
The App component	372
The first spec for App	376
More assertions for App	383
Using beforeEach	388
Simulating a change	392
Clearing the input field	396
Simulating a form submission	398
Writing tests for the food lookup app	407
FoodSearch	410
Exploring FoodSearch	412
Writing FoodSearch.test.js	419
In initial state	420
A user has typed a value into the search field	423
Mocking with Jest	428
Mocking Client	431
The API returns results	437
The user clicks on a food item	443
The API returns empty result set	449
Further reading	454
Routing	457
What's in a URL?	457
React Router's core components	460
Building the components of react-router	461
The completed app	461
Building Route	463
Building Link	471

CONTENTS

Building Router	477
Building Redirect	483
Using react-router	488
More Route	489
Using Switch	495
Dynamic routing with React Router	498
The completed app	498
The server's API	502
Starting point of the app	505
Using URL params	511
Propagating pathnames as props	519
Dynamic menu items with NavLink	525
Supporting authenticated routes	529
The Client library	530
Implementing login	532
PrivateRoute, a higher-order component	539
Redirect state	544
Recap	546
Further Reading	546

Part II 547

Intro to Flux and Redux	548
Why Flux?	548
Flux is a Design Pattern	549
Flux overview	549
Flux implementations	550
Redux	551
Redux's key ideas	551
Building a counter	552
Preparation	552
Overview	553
The counter's actions	554
Incrementing the counter	555
Decrementing the counter	556

CONTENTS

Supporting additional parameters on actions	558
Building the store	560
Try it out	564
The core of Redux	565
Next up	566
The beginnings of a chat app	566
Previewing	566
State	569
Actions	569
Building the reducer()	570
Initializing state	570
Handling the ADD_MESSAGE action	571
Handling the DELETE_MESSAGE action	575
Subscribing to the store	577
createStore() in full	580
Connecting Redux to React	582
Using store.getState()	583
Using store.subscribe()	583
Using store.dispatch()	584
The app's components	585
Preparing App.js	585
The App component	586
The MessageInput component	588
The MessageView component	590
Next up	592
Intermediate Redux	594
Preparation	594
Using createStore() from the redux library	596
Try it out	596
Representing messages as objects in state	597
Updating ADD_MESSAGE	598
Updating DELETE_MESSAGE	601
Updating the React components	602
Introducing threads	604
Supporting threads in initialState	606

CONTENTS

Supporting threads in the React components	608
Modifying App	609
Turning <code>MessageView</code> into <code>Thread</code>	610
Try it out	611
Adding the <code>ThreadTabs</code> component	612
Updating App	612
Creating <code>ThreadTabs</code>	614
Try it out	614
Supporting threads in the reducer	615
Updating <code>ADD_MESSAGE</code> in the reducer	615
Updating the <code>MessageInput</code> component	622
Try it out	624
Updating <code>DELETE_MESSAGE</code> in the reducer	624
Try it out	628
Adding the action <code>OPEN_THREAD</code>	628
The action object	628
Modifying the reducer	629
Dispatching from <code>ThreadTabs</code>	630
Try it out	631
Breaking up the reducer function	632
A new <code>reducer()</code>	632
Updating <code>threadsReducer()</code>	635
Try it out	640
Adding <code>messagesReducer()</code>	640
Modifying the <code>ADD_MESSAGE</code> action handler	641
Creating <code>messagesReducer()</code>	642
Modifying the <code>DELETE_MESSAGE</code> action handler	643
Adding <code>DELETE_MESSAGE</code> to <code>messagesReducer()</code>	647
Defining the initial state in the reducers	649
Initial state in <code>reducer()</code>	650
Adding initial state to <code>activeThreadIdReducer()</code>	650
Adding initial state to <code>threadsReducer()</code>	651
Try it out	653
Using <code>combineReducers()</code> from <code>redux</code>	653
Next up	654

CONTENTS

Using Presentational and Container Components with Redux	656
Presentational and container components	656
Splitting up ThreadTabs	659
Splitting up Thread	665
Removing store from App	672
Try it out	673
Generating containers with react-redux	674
The Provider component	674
Wrapping App in Provider	675
Using connect() to generate ThreadTabs	676
Using connect() to generate ThreadDisplay	680
Action creators	687
Conclusion	691
Asynchronicity and server communication	692
Using GraphQL	693
Your First GraphQL Query	693
GraphQL Benefits	696
GraphQL vs. REST	697
GraphQL vs. SQL	698
Relay and GraphQL Frameworks	699
Chapter Preview	700
Consuming GraphQL	701
Exploring With GraphQL	701
GraphQL Syntax 101	706
Complex Types	711
Unions	712
Fragments	713
Interfaces	714
Exploring a Graph	715
Graph Nodes	718
Viewer	720
Graph Connections and Edges	722
Mutations	726
Subscriptions	727
GraphQL With JavaScript	729

CONTENTS

GraphQL With React	730
Wrapping Up	732
GraphQL Server	733
Writing a GraphQL Server	733
Special setup for Windows users	733
Game Plan	735
Express HTTP Server	735
Adding First GraphQL Types	738
Adding GraphQL	741
Introspection	743
Mutation	745
Rich Schemas and SQL	748
Setting Up The Database	749
Schema Design	754
Object and Scalar Types	756
Lists	763
Performance: Look-Ahead Optimizations	766
Lists Continued	769
Connections	772
Authentication	782
Authorization	784
Rich Mutations	789
Relay and GraphQL	793
Performance: N+1 Queries	795
Summary	800
Relay Classic	801
Introduction	801
What We're Going to Cover	802
What We're Building	803
Guide to the Code Structure	807
Relay is a Data Architecture	809
Relay GraphQL Conventions	810
Exploring Relay Conventions in GraphQL	811
Fetching Objects By ID	811

CONTENTS

Walking Connections	816
Changing Data with Mutations	821
Relay GraphQL Queries Summary	823
Adding Relay to Our App	823
Quick Look at the Goal	823
A Preview of the Author Page	826
Containers, Queries, and Fragments	827
Validating Our Relay Queries at Compile Time	828
Setting Up Routing	835
Adding Relay to Our Routes	837
App Component	838
AuthorQueries Component	840
AuthorPage Component	841
Try It Out	842
AuthorPage with Styles	845
BooksPage	847
BooksPage Route	848
BooksPage Component	849
BooksPage render()	852
BookItem	853
BookItem Fragment	855
Fragment Value Masking	855
Improving the AuthorPage	858
Changing Data With Mutations	861
Building a Book's Page	862
Book Page Editing	866
Mutations	869
Defining a Mutation Object	870
Inline Editing	875
Conclusion	877
Where to Go From Here	878
React Native	879
Init	880
Routing	882
<Navigator />	886

CONTENTS

renderScene()	887
configureScene()	890
Web components vs. Native components	893
<View />	894
<Text />	894
<Image />	895
<TextInput />	895
<TouchableHighlight />, <TouchableOpacity />, and <Touchable-	
WithoutFeedback />	895
<ActivityIndicator />	896
<WebView />	896
<ScrollView />	896
<ListView />	897
Styles	907
StyleSheet	908
Flexbox	910
HTTP requests	932
What is a promise	932
Enter Promises	935
Single-use guarantee	937
Creating a promise	937
Debugging with React Native	939
Where to go from here	941
Appendix A: PropTypes	943
Validators	945
string	945
number	946
boolean	947
function	948
object	949
object shape	950
multiple types	951
instanceOf	952
array	953
array of type	954

CONTENTS

node	955
element	956
any type	958
Optional & required props	958
custom validator	959
Appendix B: ES6	962
Prefer const and let over var	962
Arrow functions	963
Modules	966
Object.assign()	969
Template literals	970
The spread operator (...)	971
Enhanced object literals	971
Default arguments	972
Destructuring assignments	973
Appendix C: React Hooks	976
Motivation behind Hooks	976
How Hooks Map to Component Classes	977
Using Hooks Requires react "next"	977
useState() Hook Example	977
Our Component is a Function	981
Reading and Writing State	981
React Tracks the State	982
Multiple States	982
useEffect() Hook Example	983
Fetch Data and Update State	984
Performance Concerns When Using Effects	985
useContext() Hook Example	986
The Point of Context	986
useContext() makes context easier to use	986
Getting a Reference to the Context in a Larger App	987
useRef() Hook Example	988
useRef() and forms with input	988
Using Custom Hooks	990

CONTENTS

Writing Tests for React Hooks	992
Writing tests for useState() Hook	993
Writing tests for useEffect() Hook	994
Writing tests for useRef() Hook	996
Community Reaction to Hooks	997
References to the Different types of Hooks	998
Future of Hooks	999
More Resources	1000
Changelog	1001
Revision 41 - 2020-07-06	1001
Revision 40 - 2020-01-13	1001
Revision 39 - 2019-01-10	1001
Revision 38 - 2018-12-20	1001
Revision 37 - 2018-12-19	1001
Revision 36 - 2018-10-01	1002
Revision 35 - 2018-04-02	1002
Revision 34 - 2017-10-17	1002
Revision 33 - 2017-08-31	1002
Revision 32 - 2017-06-14	1002
Revision 31 - 2017-05-18	1002
Revision 30 - 2017-04-20	1003
Revision 29 - 2017-04-13	1003
Revision 28 - 2017-04-10	1003
Revision 27 - 2017-03-16	1003
Revision 26 - 2017-02-22	1003
Revision 25 - 2017-02-17	1003
Revision 24 - 2017-02-08	1004
Revision 23 - 2017-02-06	1004
Revision 22 - 2017-02-01	1004
Revision 21 - 2017-01-27	1004
Revision 20 - 2017-01-10	1004
Revision 19 - 2016-12-20	1004
Revision 18 - 2016-11-22	1005
Revision 17 - 2016-11-04	1005
Revision 16 - 2016-10-12	1005