

# Contents

Book Revision . . . . .	1
Bug Reports . . . . .	1
Chat With The Community! . . . . .	1
Be notified of updates via Twitter . . . . .	1
We'd love to hear from you! . . . . .	1
<b>Foreword . . . . .</b>	<b>2</b>
<b>How to Get the Most Out of This Book . . . . .</b>	<b>1</b>
Overview . . . . .	1
Running Code Examples . . . . .	2
Project setups . . . . .	2
Code Blocks and Context . . . . .	3
Code Block Numbering . . . . .	3
Getting Help . . . . .	3
Emailing Us . . . . .	4
Technical Support Response Time . . . . .	5
Get excited . . . . .	5
<b>Part I . . . . .</b>	<b>6</b>
<b>Your first React Web Application . . . . .</b>	<b>7</b>
Building Product Hunt . . . . .	7
Setting up your development environment . . . . .	7
Code editor . . . . .	7
Node.js and npm . . . . .	7
Install Git . . . . .	8
Browser . . . . .	8
Special instruction for Windows users . . . . .	8
Ensure IIS is installed . . . . .	8
JavaScript ES6/ES7 . . . . .	8
Getting started . . . . .	9
Sample Code . . . . .	9

## CONTENTS

Previewing the application . . . . .	9
Prepare the app . . . . .	11
What's a component? . . . . .	15
Our first component . . . . .	16
JSX . . . . .	18
The developer console . . . . .	19
Babel . . . . .	21
ReactDOM.render() . . . . .	23
Building Product . . . . .	25
Making Product data-driven . . . . .	27
The data model . . . . .	28
Using props . . . . .	28
Rendering multiple products . . . . .	32
React the vote (your app's first interaction) . . . . .	37
Propagating the event . . . . .	37
Binding custom component methods . . . . .	39
Using state . . . . .	42
Setting state with this.setState() . . . . .	44
Updating state and immutability . . . . .	46
Refactoring with the Babel plugin transform-class-properties . . . . .	51
Babel plugins and presets . . . . .	51
Property initializers . . . . .	52
Refactoring Product . . . . .	53
Refactoring ProductList . . . . .	54
Congratulations! . . . . .	56
<b>Components . . . . .</b>	<b>57</b>
A time-logging app . . . . .	57
Getting started . . . . .	58
Previewing the app . . . . .	58
Prepare the app . . . . .	58
Breaking the app into components . . . . .	62
The steps for building React apps from scratch . . . . .	69
Step 2: Build a static version of the app . . . . .	71
TimersDashboard . . . . .	71
EditableTimer . . . . .	73
TimerForm . . . . .	74
ToggleableTimerForm . . . . .	75
Timer . . . . .	76
Render the app . . . . .	77
Try it out . . . . .	78
Step 3: Determine what should be stateful . . . . .	79
State criteria . . . . .	79

## CONTENTS

Applying the criteria . . . . .	80
Step 4: Determine in which component each piece of state should live . . . . .	81
The list of timers and properties of each timer . . . . .	82
Whether or not the edit form of a timer is open . . . . .	82
Visibility of the create form . . . . .	82
Step 5: Hard-code initial states . . . . .	83
Adding state to TimersDashboard . . . . .	83
Receiving props in EditableTimerList . . . . .	84
Props vs. state . . . . .	85
Adding state to EditableTimer . . . . .	85
Timer remains stateless . . . . .	86
Adding state to ToggleableTimerForm . . . . .	86
Adding state to TimerForm . . . . .	88
Step 6: Add inverse data flow . . . . .	91
TimerForm . . . . .	92
ToggleableTimerForm . . . . .	93
TimersDashboard . . . . .	95
Updating timers . . . . .	96
Adding editability to Timer . . . . .	97
Updating EditableTimer . . . . .	97
Updating EditableTimerList . . . . .	99
Defining onEditFormSubmit() in TimersDashboard . . . . .	99
Deleting timers . . . . .	102
Adding the event handler to Timer . . . . .	102
Routing through EditableTimer . . . . .	103
Routing through EditableTimerList . . . . .	103
Implementing the delete function in TimersDashboard . . . . .	104
Adding timing functionality . . . . .	105
Adding a forceUpdate() interval to Timer . . . . .	106
Try it out . . . . .	107
Add start and stop functionality . . . . .	107
Add timer action events to Timer . . . . .	107
Create TimerActionButton . . . . .	108
Run the events through EditableTimer and EditableTimerList . . . . .	109
Try it out . . . . .	112
Methodology review . . . . .	113
<b>Components &amp; Servers . . . . .</b>	<b>115</b>
Introduction . . . . .	115
Preparation . . . . .	115
server.js . . . . .	115
The Server API . . . . .	116
text/html endpoint . . . . .	117

## CONTENTS

JSON endpoints . . . . .	117
Playing with the API . . . . .	118
Loading state from the server . . . . .	121
Try it out . . . . .	124
client . . . . .	124
Fetch . . . . .	125
Sending starts and stops to the server . . . . .	128
Sending creates, updates, and deletes to the server . . . . .	130
Give it a spin . . . . .	132
Next up . . . . .	132
<b>JSX and the Virtual DOM . . . . .</b>	<b>133</b>
React Uses a Virtual DOM . . . . .	133
Why Not Modify the Actual DOM? . . . . .	133
What is a Virtual DOM? . . . . .	133
Virtual DOM Pieces . . . . .	134
ReactDOM . . . . .	135
Experimenting with ReactDOM . . . . .	135
Rendering Our ReactDOM . . . . .	137
Adding Text (with children) . . . . .	139
ReactDOM.render() . . . . .	140
JSX . . . . .	141
JSX Creates Elements . . . . .	141
JSX Attribute Expressions . . . . .	143
JSX Conditional Child Expressions . . . . .	143
JSX Boolean Attributes . . . . .	144
JSX Comments . . . . .	144
JSX Spread Syntax . . . . .	144
JSX Gotchas . . . . .	145
JSX Summary . . . . .	148
References . . . . .	149
<b>Advanced Component Configuration with props, state, and children . . . . .</b>	<b>150</b>
Intro . . . . .	150
How to use this chapter . . . . .	151
ReactDOM . . . . .	151
Creating ReactDOM - createClass or ES6 Classes . . . . .	151
render() Returns a ReactDOM Tree . . . . .	152
Getting Data into render() . . . . .	153
props are the parameters . . . . .	154
PropTypes . . . . .	155
Default props with getDefaultProps() . . . . .	156
context . . . . .	157

## CONTENTS

state . . . . .	162
Using state: Building a Custom Radio Button . . . . .	163
Stateful components . . . . .	168
State updates that depend on the current state . . . . .	170
Thinking About State . . . . .	172
Stateless Components . . . . .	173
Switching to Stateless . . . . .	174
Stateless Encourages Reuse . . . . .	176
Talking to Children Components with <code>props.children</code> . . . . .	176
<code>React.Children.map()</code> & <code>React.Children.forEach()</code> . . . . .	178
<code>React.Children.toArray()</code> . . . . .	180
Summary . . . . .	181
References . . . . .	181
<b>Forms . . . . .</b>	<b>182</b>
Forms 101 . . . . .	182
Preparation . . . . .	182
The Basic Button . . . . .	183
Events and Event Handlers . . . . .	185
Back to the Button . . . . .	186
Text Input . . . . .	188
Accessing User Input With <code>refs</code> . . . . .	188
Using User Input . . . . .	190
Uncontrolled vs. Controlled Components . . . . .	193
Accessing User Input With <code>state</code> . . . . .	194
Multiple Fields . . . . .	196
On Validation . . . . .	201
Adding Validation to Our App . . . . .	201
Creating the Field Component . . . . .	205
Using our new Field Component . . . . .	209
Remote Data . . . . .	214
Building the Custom Component . . . . .	214
Adding <code>CourseSelect</code> . . . . .	220
Separation of View and State . . . . .	223
Async Persistence . . . . .	223
Redux . . . . .	230
Form Component . . . . .	235
Connect the Store . . . . .	239
Form Modules . . . . .	241
<code>formsy-react</code> . . . . .	241
<code>react-input-enhancements</code> . . . . .	241
<code>tcomb-form</code> . . . . .	242
<code>winterfell</code> . . . . .	242

## CONTENTS

react-redux-form . . . . .	242
<b>Using Webpack with Create React App . . . . .</b>	<b>243</b>
JavaScript modules . . . . .	243
Create React App . . . . .	245
Exploring Create React App . . . . .	246
public/index.html . . . . .	247
package.json . . . . .	248
src/ . . . . .	250
index.js . . . . .	252
Booting the app . . . . .	254
Webpack basics . . . . .	255
Making modifications to the sample app . . . . .	261
Hot reloading . . . . .	261
Auto-reloading . . . . .	262
Creating a production build . . . . .	263
Ejecting . . . . .	266
Buckle up . . . . .	267
Using Create React App with an API server . . . . .	269
The completed app . . . . .	269
How the app is organized . . . . .	273
The server . . . . .	274
Client . . . . .	275
Concurrently . . . . .	276
Using the Webpack development proxy . . . . .	279
Webpack at large . . . . .	281
When to use Webpack/Create React App . . . . .	282
<b>Unit Testing . . . . .</b>	<b>283</b>
Writing tests without a framework . . . . .	283
Preparing Modash . . . . .	284
Writing the first spec . . . . .	287
The <code>assertEqual()</code> function . . . . .	289
What is Jest? . . . . .	293
Using Jest . . . . .	293
<code>expect()</code> . . . . .	294
The first Jest test for Modash . . . . .	296
The other <code>truncate()</code> spec . . . . .	298
The rest of the specs . . . . .	299
Testing strategies for React applications . . . . .	301
Integration vs Unit Testing . . . . .	301
Shallow rendering . . . . .	302
Enzyme . . . . .	302

## CONTENTS

Testing a basic React component with Enzyme . . . . .	303
Setup . . . . .	303
The App component . . . . .	304
The first spec for App . . . . .	308
More assertions for App . . . . .	312
Using beforeEach . . . . .	315
Simulating a change . . . . .	318
Clearing the input field . . . . .	322
Simulating a form submission . . . . .	324
Writing tests for the food lookup app . . . . .	331
FoodSearch . . . . .	333
Exploring FoodSearch . . . . .	335
Writing FoodSearch.test.js . . . . .	339
In initial state . . . . .	341
A user has typed a value into the search field . . . . .	343
Mocking with Jest . . . . .	347
Mocking Client . . . . .	350
The API returns results . . . . .	356
The user clicks on a food item . . . . .	361
The API returns empty result set . . . . .	365
Further reading . . . . .	369
<b>Routing . . . . .</b>	<b>372</b>
What's in a URL? . . . . .	372
React Router's core components . . . . .	374
Building the components of react-router . . . . .	375
The completed app . . . . .	375
Building Route . . . . .	377
Building Link . . . . .	384
Building Router . . . . .	389
Building Redirect . . . . .	394
Using react-router . . . . .	399
More Route . . . . .	400
Using Switch . . . . .	405
Dynamic routing with React Router . . . . .	407
The completed app . . . . .	408
The server's API . . . . .	411
Starting point of the app . . . . .	413
Using URL params . . . . .	419
Propagating pathnames as props . . . . .	426
Dynamic menu items with NavLink . . . . .	431
Supporting authenticated routes . . . . .	434
The Client library . . . . .	435

## CONTENTS

Implementing login . . . . .	436
PrivateRoute, a higher-order component . . . . .	442
Redirect state . . . . .	446
Recap . . . . .	448
Further Reading . . . . .	448

## **Part II . . . . . 449**

<b>Intro to Flux and Redux . . . . .</b>	<b>450</b>
Why Flux? . . . . .	450
Flux is a Design Pattern . . . . .	450
Flux overview . . . . .	451
Flux implementations . . . . .	452
Redux . . . . .	452
Redux's key ideas . . . . .	452
Building a counter . . . . .	453
Preparation . . . . .	453
Overview . . . . .	454
The counter's actions . . . . .	455
Incrementing the counter . . . . .	455
Decrementing the counter . . . . .	457
Supporting additional parameters on actions . . . . .	458
Building the store . . . . .	460
Try it out . . . . .	463
The core of Redux . . . . .	464
Next up . . . . .	465
The beginnings of a chat app . . . . .	466
Previewing . . . . .	466
State . . . . .	468
Actions . . . . .	468
Building the reducer() . . . . .	469
Initializing state . . . . .	469
Handling the ADD_MESSAGE action . . . . .	469
Handling the DELETE_MESSAGE action . . . . .	472
Subscribing to the store . . . . .	475
createStore() in full . . . . .	477
Connecting Redux to React . . . . .	479
Using store.getState() . . . . .	479
Using store.subscribe() . . . . .	480
Using store.dispatch() . . . . .	480
The app's components . . . . .	481



## CONTENTS

Preparing App.js . . . . .	482
The App component . . . . .	482
The MessageInput component . . . . .	484
The MessageView component . . . . .	486
Next up . . . . .	487
<b>Intermediate Redux . . . . .</b>	<b>489</b>
Preparation . . . . .	489
Using createStore() from the redux library . . . . .	490
Try it out . . . . .	491
Representing messages as objects in state . . . . .	491
Updating ADD_MESSAGE . . . . .	492
Updating DELETE_MESSAGE . . . . .	494
Updating the React components . . . . .	495
Introducing threads . . . . .	497
Supporting threads in initialState . . . . .	498
Supporting threads in the React components . . . . .	500
Modifying App . . . . .	501
Turning MessageView into Thread . . . . .	502
Try it out . . . . .	503
Adding the ThreadTabs component . . . . .	503
Updating App . . . . .	504
Creating ThreadTabs . . . . .	505
Try it out . . . . .	505
Supporting threads in the reducer . . . . .	506
Updating ADD_MESSAGE in the reducer . . . . .	506
Updating the MessageInput component . . . . .	512
Try it out . . . . .	513
Updating DELETE_MESSAGE in the reducer . . . . .	514
Try it out . . . . .	516
Adding the action OPEN_THREAD . . . . .	517
The action object . . . . .	517
Modifying the reducer . . . . .	517
Dispatching from ThreadTabs . . . . .	518
Try it out . . . . .	519
Breaking up the reducer function . . . . .	520
A new reducer() . . . . .	521
Updating threadsReducer() . . . . .	523
Try it out . . . . .	526
Adding messagesReducer() . . . . .	527
Modifying the ADD_MESSAGE action handler . . . . .	527
Creating messagesReducer() . . . . .	528
Modifying the DELETE_MESSAGE action handler . . . . .	529

## CONTENTS

Adding DELETE_MESSAGE to messagesReducer()	532
Defining the initial state in the reducers	533
Initial state in reducer()	534
Adding initial state to activeThreadIdReducer()	534
Adding initial state to threadsReducer()	535
Try it out	536
Using combineReducers() from redux	536
Next up	537
<b>Using Presentational and Container Components with Redux</b>	<b>539</b>
Presentational and container components	539
Splitting up ThreadTabs	541
Splitting up Thread	546
Removing store from App	552
Try it out	553
Generating containers with react-redux	553
The Provider component	554
Wrapping App in Provider	554
Using connect() to generate ThreadTabs	555
Using connect() to generate ThreadDisplay	559
Action creators	564
Conclusion	568
Asynchronicity and server communication	568
<b>Using GraphQL</b>	<b>570</b>
Your First GraphQL Query	570
GraphQL Benefits	572
GraphQL vs. REST	573
GraphQL vs. SQL	574
Relay and GraphQL Frameworks	574
Chapter Preview	576
Consuming GraphQL	576
Exploring With GraphQL	576
GraphQL Syntax 101	580
Complex Types	584
Unions	584
Fragments	585
Interfaces	586
Exploring a Graph	587
Graph Nodes	590
Viewer	591
Graph Connections and Edges	592
Mutations	596

## CONTENTS

Subscriptions . . . . .	597
GraphQL With JavaScript . . . . .	598
GraphQL With React . . . . .	599
Wrapping Up . . . . .	601
<b>GraphQL Server . . . . .</b>	<b>602</b>
Writing a GraphQL Server . . . . .	602
Special setup for Windows users . . . . .	602
Game Plan . . . . .	603
Express HTTP Server . . . . .	603
Adding First GraphQL Types . . . . .	606
Adding GraphiQL . . . . .	608
Introspection . . . . .	610
Mutation . . . . .	611
Rich Schemas and SQL . . . . .	614
Setting Up The Database . . . . .	615
Schema Design . . . . .	619
Object and Scalar Types . . . . .	621
Lists . . . . .	626
Performance: Look-Ahead Optimizations . . . . .	628
Lists Continued . . . . .	631
Connections . . . . .	634
Authentication . . . . .	641
Authorization . . . . .	643
Rich Mutations . . . . .	647
Relay and GraphQL . . . . .	650
Performance: N+1 Queries . . . . .	651
Summary . . . . .	655
<b>Relay . . . . .</b>	<b>657</b>
Introduction . . . . .	657
What We're Going to Cover . . . . .	658
What We're Building . . . . .	658
Guide to the Code Structure . . . . .	662
Relay is a Data Architecture . . . . .	663
Relay GraphQL Conventions . . . . .	664
Exploring Relay Conventions in GraphQL . . . . .	665
Fetching Objects By ID . . . . .	665
Walking Connections . . . . .	669
Changing Data with Mutations . . . . .	674
Relay GraphQL Queries Summary . . . . .	675
Adding Relay to Our App . . . . .	675
Quick Look at the Goal . . . . .	675

## CONTENTS

A Preview of the Author Page . . . . .	678
Containers, Queries, and Fragments . . . . .	679
Validating Our Relay Queries at Compile Time . . . . .	679
Setting Up Routing . . . . .	685
Adding Relay to Our Routes . . . . .	687
App Component . . . . .	688
AuthorQueries Component . . . . .	689
AuthorPage Component . . . . .	689
Try It Out . . . . .	691
AuthorPage with Styles . . . . .	693
BooksPage . . . . .	695
BooksPage Route . . . . .	695
BooksPage Component . . . . .	696
BooksPage render() . . . . .	698
BookItem . . . . .	699
BookItem Fragment . . . . .	701
Fragment Value Masking . . . . .	701
Improving the AuthorPage . . . . .	703
Changing Data With Mutations . . . . .	706
Building a Book's Page . . . . .	707
Book Page Editing . . . . .	710
Mutations . . . . .	713
Defining a Mutation Object . . . . .	714
Inline Editing . . . . .	718
Conclusion . . . . .	720
Where to Go From Here . . . . .	721
<b>React Native . . . . .</b>	<b>722</b>
Init . . . . .	723
Routing . . . . .	724
<Navigator /> . . . . .	727
renderScene() . . . . .	729
configureScene() . . . . .	731
Web components vs. Native components . . . . .	734
<View /> . . . . .	735
<Text /> . . . . .	735
<Image /> . . . . .	735
<TextInput /> . . . . .	735
<TouchableHighlight />, <TouchableOpacity />, and <TouchableWithoutFeedback /> . . . . .	735
<ActivityIndicator /> . . . . .	736
<WebView /> . . . . .	736
<ScrollView /> . . . . .	736

## CONTENTS

<ListView /> . . . . .	737
Styles . . . . .	745
StyleSheet . . . . .	746
Flexbox . . . . .	747
HTTP requests . . . . .	765
What is a promise . . . . .	766
Enter Promises . . . . .	768
Single-use guarantee . . . . .	770
Creating a promise . . . . .	770
Debugging with React Native . . . . .	772
Where to go from here . . . . .	773
<b>Appendix A: PropTypes . . . . .</b>	<b>775</b>
Validators . . . . .	776
string . . . . .	777
number . . . . .	777
boolean . . . . .	778
function . . . . .	779
object . . . . .	780
object shape . . . . .	780
multiple types . . . . .	781
instanceOf . . . . .	782
array . . . . .	783
array of type . . . . .	784
node . . . . .	785
element . . . . .	786
any type . . . . .	787
Optional & required props . . . . .	787
custom validator . . . . .	788
<b>Appendix B: ES6 . . . . .</b>	<b>790</b>
Prefer const and let over var . . . . .	790
Arrow functions . . . . .	790
Modules . . . . .	793
Object.assign() . . . . .	796
Template literals . . . . .	797
The spread operator (...) . . . . .	797
Enhanced object literals . . . . .	798
Default arguments . . . . .	798
Destructuring assignments . . . . .	799
<b>Changelog . . . . .</b>	<b>802</b>
Revision 33 - 2017-08-31 . . . . .	802

CONTENTS

Revision 32 - 2017-06-14 . . . . .	802
Revision 31 - 2017-05-18 . . . . .	802
Revision 30 - 2017-04-20 . . . . .	802
Revision 29 - 2017-04-13 . . . . .	802
Revision 28 - 2017-04-10 . . . . .	802
Revision 27 - 2017-03-16 . . . . .	803
Revision 26 - 2017-02-22 . . . . .	803
Revision 25 - 2017-02-17 . . . . .	803
Revision 24 - 2017-02-08 . . . . .	803
Revision 23 - 2017-02-06 . . . . .	803
Revision 22 - 2017-02-01 . . . . .	803
Revision 21 - 2017-01-27 . . . . .	804
Revision 20 - 2017-01-10 . . . . .	804
Revision 19 - 2016-12-20 . . . . .	804
Revision 18 - 2016-11-22 . . . . .	804
Revision 17 - 2016-11-04 . . . . .	804
Revision 16 - 2016-10-12 . . . . .	804
Revision 15 - 2016-10-05 . . . . .	804
Revision 14 - 2016-08-26 . . . . .	804
Revision 13 - 2016-08-02 . . . . .	805
Revision 12 - 2016-07-26 . . . . .	805
Revision 11 - 2016-07-08 . . . . .	805
Revision 10 - 2016-06-24 . . . . .	805
Revision 9 - 2016-06-21 . . . . .	805
Revision 8 - 2016-06-02 . . . . .	805
Revision 7 - 2016-05-13 . . . . .	805
Revision 6 - 2016-05-13 . . . . .	805
Revision 5 - 2016-04-25 . . . . .	805
Revision 4 - 2016-04-22 . . . . .	806
Revision 3 - 2016-04-08 . . . . .	806
Revision 2 - 2016-03-16 . . . . .	806
Revision 1 - 2016-02-14 . . . . .	806

## Book Revision

Revision 33 - Supports React 15.5.4 (2017-08-31)

## Bug Reports

If you'd like to report any bugs, typos, or suggestions just email us at: [react@fullstack.io](mailto:react@fullstack.io)<sup>1</sup>.

For further help dealing with issues, refer to "How to Get the Most Out of This Book."

## Chat With The Community!

There's an unofficial community chat room for this book using Gitter. If you'd like to hang out with other people learning React, come [join us on Gitter](https://gitter.im/fullstackreact/fullstackreact)<sup>2</sup>!

## Be notified of updates via Twitter

If you'd like to be notified of updates to the book on Twitter, [follow @fullstackio](https://twitter.com/fullstackio)<sup>3</sup>

## We'd love to hear from you!

Did you like the book? Did you find it helpful? We'd love to add your face to our list of testimonials on the website! Email us at: [react@fullstack.io](mailto:react@fullstack.io)<sup>4</sup>.

---

<sup>1</sup><mailto:react@fullstack.io?Subject=Fullstack%20React%20book%20feedback>

<sup>2</sup><https://gitter.im/fullstackreact/fullstackreact>

<sup>3</sup><https://twitter.com/fullstackio>

<sup>4</sup><mailto:react@fullstack.io?Subject=react%20%20testimonial>

# Foreword

Web development is often seen as a crazy world where the way you develop software is by throwing hacks on top of hacks. I believe that React breaks from this pattern and instead has been designed from first principle which gives you a solid foundation to build on.

A major source of bugs for front-end applications was around synchronizing the data model with the DOM. It is very hard to make sure that whenever data changes, everything in the UI is updated with it.

React's first innovation was to introduce a pure-JavaScript representation of the DOM and implement diffing in userland and then use events which send simple commands: create, update, delete.

With React, by conceptually **re-rendering everything whenever anything changes**, not only do you have code that is **safe by default**, it is also **less work** as you only need to write the creation path: updates are taken care of for you.

Browsers have, for a long time, been incompatible in various ways due to the large API surface area of what they have to support to make the DOM work. Not only does React provide a great way to solve browser differences, but it enables use cases that were never before possible for a front-end library, such as server-side rendering and the ability to implement rendering targets like native iOS, Android, and even hardware components.

But the most important thing about React and the main reason why you should read this book: not only will you use it to **make great applications for your users**, it will also **make you a better developer**. Libraries come and go all the time and React is likely going to be no exception. What makes it special is that it teaches you **concepts that can be reused throughout your entire career**.

**You will become better at JavaScript** because React doesn't come with a templating system. Instead, React pushes you to use the **full power of JavaScript** to build your user interface.

You are going to practice using parts of *functional programming* with `map` and `filter` and also encouraged to use the **latest features of JavaScript** (including ES6). By not abstracting away data management, React will force you to think about how to architect your app and encourage you to consider concepts like immutability.

I'm very proud that the community built around React is not afraid of "rethinking best practices." The community challenges the status quo in many areas. My advice to you is to read this **excellent**



Christopher Chedeau - Front-end Engineer at Facebook



book to learn and understand the fundamentals of React. Learning new concepts may feel strange but “give it 5 minutes” and practice them until you feel comfortable.

Then, **try to break the rules**. There is no one best way to build software and React is no exception. React actually embraces this fact by providing you with escape hatches when you want to do things outside of the React-way.

Come up with crazy ideas and who knows, maybe you are going to invent the successor to React!

– Christopher Chedeau [@vjeux](#)<sup>5</sup> Front-end Engineer at Facebook

---

<sup>5</sup><https://twitter.com/Vjeux>