MANNING

Developing iOS and Android apps with JavaScript

React Native IN ACTION

Nader Dabit

React Native in Action

Developing iOS and Android apps with JavaScript

NADER DABIT

MANNING
SHELTER ISLAND

For online information and ordering of this and other Manning books, please visit www.manning.com. The publisher offers discounts on this book when ordered in quantity.

For more information, please contact

Special Sales Department
Manning Publications Co.
20 Baldwin Road
PO Box 761
Shelter Island, NY 11964
Email: orders@manning.com

©2019 by Manning Publications Co. All rights reserved.

No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by means electronic, mechanical, photocopying, or otherwise, without prior written permission of the publisher.

Many of the designations used by manufacturers and sellers to distinguish their products are claimed as trademarks. Where those designations appear in the book, and Manning Publications was aware of a trademark claim, the designations have been printed in initial caps or all caps.

Recognizing the importance of preserving what has been written, it is Manning's policy to have the books we publish printed on acid-free paper, and we exert our best efforts to that end. Recognizing also our responsibility to conserve the resources of our planet, Manning books are printed on paper that is at least 15 percent recycled and processed without the use of elemental chlorine.

Manning Publications Co.
20 Baldwin Road
PO Box 761
Shelter Island, NY 11964

Development editor: Marina Michaels
Project editor: Tiffany Taylor
Copy editor: Tiffany Taylor
Proofreader: Melody Dolab
Typesetter: Happenstance Type-O-Rama
Cover designer: Marija Tudor

ISBN: 9781617294051
Printed in the United States of America
1 2 3 4 5 6 7 8 9 10 – SP – 24 23 22 21 20 19

brief contents

PART 1 GETTING STARTED WITH REACT NATIVE 1

1 ■ Getting started with React Native 3
2 ■ Understanding React 27
3 ■ Building your first React Native app 45

PART 2 DEVELOPING APPLICATIONS IN REACT NATIVE 73

4 ■ Introduction to styling 75
5 ■ Styling in depth 115
6 ■ Navigation 145
7 ■ Animations 162
8 ■ Using the Redux data architecture library 179

PART 3 API REFERENCE 197

9 ■ Implementing cross-platform APIs 199
10 ■ Implementing iOS-specific components and APIs 222
11 ■ Implementing Android-specific components and APIs 242

PART 4 BRINGING IT ALL TOGETHER 261

12 ■ Building a Star Wars app using cross-platform components 263

contents

preface xi
acknowledgments xiii
about this book xv
about the author xviii
about the cover illustration xix

PART 1 GETTING STARTED WITH REACT NATIVE 1

1 *Getting started with React Native* 3

1.1 Introducing React and React Native 4
A basic React class 5 ▪ *React lifecycle* 6

1.2 What you'll learn 7

1.3 What you should know 7

1.4 Understanding how React Native works 8
JSX 8 ▪ *Threading* 8 ▪ *React* 8 ▪ *Unidirectional data flow* 8 ▪ *Diffing* 8 ▪ *Thinking in components* 9

1.5 Acknowledging React Native's strengths 10
Developer availability 11 ▪ *Developer productivity* 11 *Performance* 12 ▪ *One-way data flow* 12 ▪ *Developer experience* 13 ▪ *Transpilation* 13 ▪ *Productivity and efficiency* 13 ▪ *Community* 14 ▪ *Open source* 14 ▪ *Immediate updates* 14 ▪ *Other solutions for building cross-platform mobile applications* 14

1.6 React Native's drawbacks 15

1.7 Creating and using basic components 15

An overview of components 16 ▪ Native components 16 Component composition 17 ▪ Exportable components 19 Combining components 21

1.8 Creating a starter project 22

Create React Native App CLI 22 ▪ React Native CLI 23

2 *Understanding React 27*

2.1 Managing component data using state 28

Correctly manipulating component state 28

2.2 Managing component data using props 32

2.3 React component specifications 39

Using the render method to create a UI 39 ▪ Using property initializers and constructors 40

2.4 React lifecycle methods 41

The static getDerivedStateFromProps method 42 The componentDidMount lifecycle method 42 The shouldComponentUpdate lifecycle method 43 The componentDidUpdate lifecycle method 43 The componentWillUnmount lifecycle method 44

3 *Building your first React Native app 45*

3.1 Laying out the todo app 46

3.2 Coding the todo app 47

3.3 Opening the developer menu 52

Opening the developer menu in the iOS simulator 52 ▪ Opening the developer menu in the Android emulator 53 ▪ Using the developer menu 53

3.4 Continuing building the todo app 55

PART 2 DEVELOPING APPLICATIONS IN REACT NATIVE 73

4 *Introduction to styling 75*

4.1 Applying and organizing styles in React Native 76

Applying styles in applications 76 ▪ Organizing styles 78 ▪ Styles are code 80

4.2 Styling view components 83

Setting the background color 84 ▪ Setting border properties 86 ▪ Specifying margins and padding 92 ▪ Using position to place components 97 ▪ Profile Card positioning 99

4.3 Styling Text components 100

Text components vs. View components 100 ▪ Font styles 104 ▪ Using decorative text styles 107

5 *Styling in depth 115*

5.1 Platform-specific sizes and styles 116

Pixels, points, and DPs 116 ▪ Creating drop shadows with ShadowPropTypesIOS and Elevation 118 ▪ Putting it into practice: drop shadows in the ProfileCard 121

5.2 Using transformations to move, rotate, scale, and skew components 122

3D effects with perspective 123 ▪ Moving elements along the x- and y-axes with translateX and translateY 123 ▪ Rotating elements with rotateX, rotateY, and rotateZ (rotate) 124 ▪ Setting visibility when rotating an element more than 90° 127 ▪ Scaling objects on the screen with scale, scaleX, and scaleY 128 ▪ Using the scale transform to create a thumbnail of the ProfileCard 129 ▪ Skewing elements along the x- and y-axes with skewX and skewY 132 Transformation key points 134

5.3 Using flexbox to lay out components 135

Altering a component's dimensions with flex 135 ▪ Specifying the direction of the flex with flexDirection 136 ▪ Defining how space is used around a component with justifyContent 137 ▪ Aligning children in a container with alignItems 139 ▪ Overriding the parent container's alignment with alignSelf 140 ▪ Preventing clipped items with flexWrap 142

6 *Navigation 145*

6.1 React Native navigation vs. web navigation 146

6.2 Building a navigation-based app 146

6.3 Persisting data 159

6.4 Using DrawerNavigator to create drawer-based navigation 160

7 *Animations 162*

7.1 Introducing the Animated API 163

7.2 Animating a form input to expand on focus 165
7.3 Creating a custom loading animation using interpolation 167
7.4 Creating multiple parallel animations 170
7.5 Creating an animated sequence 172
7.6 Using Animated.stagger to stagger animation start times 175
7.7 Other useful tips for using the Animated library 177
Resetting an animated value 177 ▪ Invoking a callback 177 ▪ Offloading animations to the native thread 177 ▪ Creating a custom animatable component using createAnimatedComponent 178

8 *Using the Redux data architecture library 179*
8.1 What is Redux? 179
8.2 Using context to create and manage global state in a React application 180
8.3 Implementing Redux with a React Native app 181
8.4 Creating Redux reducers to hold Redux state 183
8.5 Adding the provider and creating the store 184
8.6 Accessing data using the connect function 185
8.7 Adding actions 187
8.8 Deleting items from a Redux store in a reducer 192

PART 3 API REFERENCE..... 197

9 *Implementing cross-platform APIs 199*
9.1 Using the Alert API to create cross-platform notifications 200
Use cases for alerts 200 ▪ Example of using alerts 201
9.2 Using the AppState API to detect the current application state 202
Use cases for AppState 203 ▪ Example of using AppState 203
9.3 Using the AsyncStorage API to persist data 204
Use cases for AsyncStorage 204 ▪ Example of using AsyncStorage 205

9.4 Using the Clipboard API to copy text into the user's clipboard 207
Use cases for Clipboard 207 ▪ Example of using Clipboard 207

9.5 Using the Dimensions API to get the user's screen information 208
Use cases for the Dimensions API 209 ▪ Example of using the Dimensions API 209

9.6 Using the Geolocation API to get the user's current location information 209
Use cases for the Geolocation API 210 ▪ Example of using Geolocation 210

9.7 Using the Keyboard API to control the location and functionality of the native keyboard 212
Use cases for the Keyboard API 212 ▪ Example of using the Keyboard API 213

9.8 Using NetInfo to get the user's current online/offline status 214
Use cases for NetInfo 215 ▪ Example of using NetInfo 216

9.9 Getting information about touch and gesture events with PanResponder 216
Use cases for the PanResponder API 217 ▪ Example of using PanResponder 218

10 ***Implementing iOS-specific components and APIs 222***

10.1 Targeting platform-specific code 223
iOS and Android file extensions 223 ▪ Detecting the platform using the Platform API 224

10.2 DatePickerIOS 226
Example of using DatePickerIOS 226

10.3 Using PickerIOS to work with lists of values 228
Example of using PickerIOS 230

10.4 Using ProgressViewIOS to show loading indicators 231
Use cases for ProgressViewIOS 232 ▪ Example of using ProgressViewIOS 232

10.5 Using SegmentedControlIOS to create horizontal tab bars 233
Use cases for SegmentedControlIOS 234 ▪ Example of using SegmentedControlIOS 234

10.6 Using TabBarIOS to render tabs at the bottom of the UI 235
Use cases for TabBarIOS 236 ▪ Example of using TabBarIOS 237

10.7 Using ActionSheetIOS to show action or share sheets 238
Use cases for ActionSheetIOS 239 ▪ Example of using ActionSheetIOS 239

11 *Implementing Android-specific components and APIs 242*
11.1 Creating a menu using DrawerLayoutAndroid 243
11.2 Creating a toolbar with ToolbarAndroid 247
11.3 Implementing scrollable paging with ViewPagerAndroid 248
11.4 Using the DatePickerAndroid API to show a native date picker 251
11.5 Creating a time picker with TimePickerAndroid 253
11.6 Implementing Android toasts using ToastAndroid 256

PART 4 BRINGING IT ALL TOGETHER 261

12 *Building a Star Wars app using cross-platform components 263*
12.1 Creating the app and installing dependencies 265
Importing the People component and creating the Container component 266 ▪ Creating the navigation component and registering routes 267 ▪ Creating the main class for the initial view 267

12.2 Creating the People component using FlatList, Modal, and Picker 270
Creating the state and setting up a fetch call to retrieve data 271 ▪ Adding the remaining class methods 273 ▪ Implementing the render method 274

12.3 Creating the HomeWorld component 276
Creating the HomeWorld class and initializing state 276 ▪ Fetching data from the API using the url prop 278 ▪ Wrapping up the HomeWorld component 279

appendix 281

index 285

preface

I've always been fascinated with the idea of mobile application development. Building mobile apps was one of the reasons I wanted to learn how to code. This fascination has lead me down many paths, from Objective-C to jQuery mobile to Cordova and now to React Native.

Because my career has centered around writing JavaScript, I've also always been drawn to technologies that increase my efficiency by using my existing skillset, allowing me to do more than just web development. Finding ways to be more efficient has been core to my career when choosing paths to follow and rabbit holes to dive into.

When React Native first landed, I knew that it was going to be something significant. There were already thousands of React and JavaScript developers in the world. React Native gave these developers a way to extend their existing skillset into the realm of mobile application development in a way that Cordova and other options didn't, and also appealed heavily to React developers who were at the time the most rapidly growing segment of all frontend developers. The framework also delivered a substantial increase in quality of applications that could be built versus other options available in the same space.

After writing my first application and shipping it to the app store, I had learned quite a bit and decided to start answering questions on Stack Overflow. I quickly realized that I had valuable knowledge I could share, while helping the community as well my career, so I began hanging out there more and more, answering questions.

I learned a lot while answering these questions, and eventually I made a conscious decision to specialize 100% in the React Native framework. I heard from many successful developers and consultants that specializing had helped them in their careers: they were more productive, got more business, and could demand a higher rate. So, I

decided to try being a specialist for the first time in my career. This decision turned out to be great for me; I quickly began getting leads for consulting and, later, training.

I've watched the React Native framework grow from its infancy to what it is today and have seen many developers and companies rapidly increase their efficiency and productivity by taking advantage of what the framework has to offer. I think we're at an exciting time for React Native: many Fortune 500 companies and enterprises are picking it up, finally solidifying it as a first-class choice in their developer toolkits and giving more confidence to people who are considering betting their companies and applications on the framework. It will be exciting to watch the framework evolve and to see the new apps that will be shipped using React Native!

acknowledgments

This is the first time I've written a book. It has been a good learning experience, and also much more work than I anticipated. While I've been writing, my career has changed a couple of times and my obligations along with it, affecting the amount of time I could commit to the book. Nickie Buckner and Marina Michaels are the reason this book is complete. If it wasn't for them, it would have been in editing indefinitely; I was unable to rewrite a couple of chapters in a reasonable amount of time, and Nickie stepped up in a huge way to finish the book. Marina also did more than what was called for in helping the book make it the last 20% of the way as my time became increasingly constrained.

Thank you to my wife, Lilly, who worked overtime in addition to her already exceedingly high normal duties as I worked late nights in the office and sometimes at home to write this book. Thank you to my kids, Victor and Eli, who are awesome; I love them very much. And thank you to my parents for putting me in a position to be able to learn things and get second, third, and fourth chances at life.

My thanks go to many groups and individuals: to the React Native community and the React Native team (Jordan Walke, Christopher Chedeau, Adam Wolff, and everyone at Facebook over the years whom I didn't mention); to Monte Thakkar, who took over React Native Elements' open source while I was writing (and to all React Native Training open source contributors); to Eric Vicenti and Brent Vatne and all the people who have worked on Navigation and many other projects I use day to day; to Charlie Cheever, who has, with Expo, pushed the development of many React Native projects and, by extension, of Expo, and who has helped many open source projects; to Parasharum N, who has been committed to building things around React Native for years, now works on React Native at Facebook, and has always been a great asset to the community

and ecosystem; to Peter Piekarczyk, Kevin Old, Lee Johnson, Gant Laborde, and Spencer Carli, who have consistently helped with the "React Native Radio" podcast; to Russ Davis and SchoolStatus, for the opportunity to learn React Native on the job, which is how I got started with it in the first place; to Orta Therox and the people at Artsy, for their commitment to the React Native community with their amazing blog and open source; to Leland Richardson, Devin Abbott, and the team at Airbnb, who gave React Native a fair shot and contributed extensively to the ecosystem even though the framework didn't work out for Airbnb in the long run; to the Wix team, who have contributed many amazing projects to the React Native open source ecosystem; to Mike Grabowski and Anna Lankauf, of Callstack, for being in charge of releasing React Native open source, for many contributions to the React Native open source ecosystem, and for collaborating with me on things over the years; and to Jason Brown for pushing amazing blog posts and teaching me about animations early on. I'm sure I left out many people, and if that person is you, I apologize and thank you for your contribution, as well.

Finally, I want to thank the people at Manning who made this book possible: publisher Marjan Bace and everyone behind the scenes on the editorial and production teams. My thanks also to the technical peer reviewers led by Aleksandar Dragosavljević: Alessandro Campeis, Andriy Kharchuk, Francesco Strazzullo, Gonzalo Barba López, Ian Lovell, Jason Rogers, Jose San Leandro, Joseph Tingsanchali, Markus Matzker, Matej Strašek, Mattias Lundell, Nickie Buckner, Olaoluwa Oluro, Owen Morris, Roger Sperberg, Stuart Rivero, Thomas Overby Hansen, Ubaldo Pescatore, and Zhuo Hong Wei. On the technical side, my thanks to Michiel Trimpe, who served as the book's technical editor; and Jason Rogers, who served as the book's technical proofreader.

about this book

React Native in Action was written to get you up and running with the React Native framework as quickly and seamlessly as possible. It uses a combination of real-world examples, discussions around APIs and development techniques, and a focus on learning things that will translate into real-world scenarios.

The book begins with an overview of React Native in chapter 1, following by a look at how React works in chapter 2. From chapter 3 through the end of the book, you build applications containing functionality you'll use to build applications in the real world. The book dives deep into topics such as data architecture, navigation, and animations, giving you a well-rounded understanding of how to build mobile apps using React Native.

The book is divided into 4 parts and 12 chapters:

- Part 1, "Getting Started with React Native":
  - Chapter 1 gets you up and running with React Native by going over what React Native is, how it works, its relationship with React, and when you might want to use React Native (and when you might not). This chapter includes an overview of React Native's components, which are at the core of React Native. It concludes with creating a small React Native project.
  - Chapter 2 covers state and props: what they are, how they work, and why they're important in React Native application development. It also covers the React Component specification and React lifecycle methods.
  - In chapter 3, you build your first React Native app—a todo app—from the ground up, and you'll learn about using the developer menu in iOS and Android to, among other things, debug your app.

- Part 2, "Developing Applications in React Native." With the basics covered, you can start adding features to your React Native app. The chapters in this part cover styling, navigation, animations, and elegant ways to handle data using data architectures (with a focus on Redux):
  - Chapters 4 and 5 teach you how to apply styles: either in line, with components, or in stylesheets that components can reference. Because React Native components are the main buildings blocks of your app's UI, chapter 4 spends some time teaching useful things you can do with the `View` component. Chapter 5 builds on the skills taught in chapter 4; it covers aspects of styling that are platform-specific, as well as some advanced techniques, including using flexbox to make it easier to lay out applications.
  - Chapter 6 shows how to use the two most-recommended and most-used navigation libraries: React Navigation and React Native Navigation. We'll walk through creating the three main types of navigators—tabs, stack, and drawer—and discuss how to control the navigation state.
  - Chapter 7 covers the four things you need to do to create animations, the four types of animatable components that ship with the Animated API, how to create custom animatable components, and several other useful skills.
  - In chapter 8, we explore handling data with data architectures. Because Redux is the most widely adopted method of handling data in the React ecosystem, you'll use it to build an app. Through doing so, you'll learn the skills needed to handle data. You'll see how to use the Context API and how to implement Redux with a React Native app by using reducers to hold the Redux state and delete items from the example app. You'll also learn how to use providers to pass global state to the rest of the app, how to use the `connect` function to access the example app from a child component, and how to use actions to add functionality.
- Part 3, "API Reference." React Native offers a wealth of APIs. The chapters in this part cover cross-platform APIs as well as APIs that are specific to the iOS and Android platforms:
  - Chapter 9 explores using React Native's cross-platform APIs: APIs that can be used on either iOS or Android to create alerts; detect whether the app is in the foreground, in the background, or inactive; persist, retrieve, and remove data; store and update text to the device clipboard; and perform a number of other useful features.
  - Chapters 10 and 11 look at React Native's APIs that are specific to either the iOS platform or the Android platform.
- Part 4, "Bringing It All Together." This part pulls together everything covered in the previous chapters—styling, navigation, animations, and some of the cross-platform components—into a single app:

- Chapter 12 starts by looking at the final design and walking through a basic overview of what the app will do. Then, you'll create a new React Native application and install the React Navigation library, dive deep into styling both the components as well as the navigation UI, work with data from external network resources by using the fetch API, and ultimately build out an application that allows users to view information about their favorite *Star Wars* characters.

Source code

This book contains many examples of source code, both in numbered listings and inline with normal text. In both cases, source code is formatted in a `fixed-width font like this` to separate it from ordinary text.

In many cases, the original source code has been reformatted; we've added line breaks and reworked indentation to accommodate the available page space in the book. In rare cases, even this was not enough, and listings include line-continuation markers (➥).

Additionally, comments in the source code have often been removed from the listings when the code is described in the text. Code annotations accompany many of the listings, highlighting important concepts.

Source code for the book's examples is available from the publisher's website at www.manning.com/books/react-native-in-action and on GitHub at https://github.com/dabit3/react-native-in-action.

Book forum

Purchase of *React Native in Action* includes free access to a private web forum run by Manning Publications where you can make comments about the book, ask technical questions, and receive help from the author and from other users. To access the forum, go to https://livebook.manning.com/#!/book/react-native-in-action/discussion. You can also learn more about Manning's forums and the rules of conduct at https://livebook.manning.com/#!/discussion.

Manning's commitment to our readers is to provide a venue where a meaningful dialogue between individual readers and between readers and the author can take place. It is not a commitment to any specific amount of participation on the part of the author, whose contribution to the forum remains voluntary (and unpaid). We suggest you try asking the author some challenging questions lest his interest stray! The forum and the archives of previous discussions will be accessible from the publisher's website as long as the book is in print.

about the author

Nader Dabit is a developer advocate at AWS Mobile, where he works on tools and services to allow developers to build full-stack web and mobile applications using their existing skillset. He is also the founder of React Native Training and the host of the "React Native Radio" podcast.

about the cover illustration

The figure on the cover of *React Native in Action* is captioned "Insulaire D'Amboine" or "Islander of Amboine." The illustration is taken from a nineteenth-century edition of Sylvain Maréchal's four-volume compendium of regional dress customs published in France. Each illustration is finely drawn and colored by hand. The rich variety of Maréchal's collection reminds us vividly of how culturally apart the world's towns and regions were just 200 years ago. Isolated from each other, people spoke different dialects and languages. Whether on city streets, in small towns, or in the countryside, it was easy to identify where they lived and what their trade or station in life was just by their dress.

Dress codes have changed since then and the diversity by region and class, so rich at the time, has faded away. It is now hard to tell apart the inhabitants of different continents, let alone different towns or regions. Perhaps we have traded cultural diversity for a more varied personal life—certainly for a more varied and fast-paced technological life.

At a time when it is hard to tell one computer book from another, Manning celebrates the inventiveness and initiative of the computer business with book covers based on the rich diversity of regional life of two centuries ago, brought back to life by Maréchal's pictures.

Part 1

Getting started with React Native

Chapter 1 will get you up and running by going over what React Native is, how it works, what its relationship with React is, and when you might want to use React Native (and when you might not). This chapter provides an overview of React Native's components, which are at the core of React Native. It concludes with creating a small React Native project.

Chapter 2 covers state and properties: what they are, how they work, and why they're important in React Native application development. It also covers the React Component specification and React lifecycle methods.

In chapter 3, you'll build your first React Native app—a Todo app—from the ground up. You'll also learn about using the developer menu in iOS and Android for, among other things, debugging apps.

1 Getting started with React Native

This chapter covers

- Introducing React Native
- The strengths of React Native
- Creating components
- Creating a starter project

Native mobile application development can be complex. With the complicated environments, verbose frameworks, and long compilation times developers face, developing a quality native mobile application is no easy task. It's no wonder the market has seen its share of solutions come onto the scene that attempt to solve the problems that go along with native mobile application development and try to make it easier.

At the core of this complexity is the obstacle of cross-platform development. The various platforms are fundamentally different and don't share much of their development environments, APIs, or code. Because of this, we must have separate teams working on each platform, which is both expensive and inefficient.

But this is an exciting time in mobile application development. We're witnessing a new paradigm in the mobile development landscape, and React Native is on the forefront of this shift in how we build and engineer mobile applications. It's now

possible to build native performing cross-platform apps as well as web applications with a single language and a single team. With the rise of mobile devices and the subsequent increase in demand for talent driving developer salaries higher and higher, React Native brings to the table the ability to deliver quality applications across all platforms at a fraction of the time and cost, while still delivering a high-quality user experience and a delightful developer experience.

1.1 Introducing React and React Native

React Native is a framework for building native mobile apps in JavaScript using the React JavaScript library; React Native code compiles to real native components. If you're not sure what React is, it's a JavaScript library open sourced by and used within Facebook. It was originally used to build user interfaces for web applications. It has since evolved and can now also be used to build server-side and mobile applications (using React Native).

React Native has a lot going for it. In addition to being backed and open sourced by Facebook, it also has a tremendous community of motivated people behind it. Facebook groups, with their millions of users, are powered by React Native as well as Facebook Ads Manager. Airbnb, Bloomberg, Tesla, Instagram, Ticketmaster, SoundCloud, Uber, Walmart, Amazon, and Microsoft are some of the other companies either investing in or using React Native in production.

With React Native, developers can build native views and access native platform-specific components using JavaScript. This sets React Native apart from other hybrid app frameworks like Cordova and Ionic, which package web views built using HTML and CSS into a native application. Instead, React Native takes JavaScript and compiles it into a true native application that can use platform-specific APIs and components. Alternatives like Xamarin take the same approach, but Xamarin apps are built using C#, not JavaScript. Many web developers have JavaScript experience, which helps ease the transition from web to mobile app development.

There are many benefits to choosing React Native as a mobile application framework. Because the application renders native components and APIs directly, speed and performance are much better than with hybrid frameworks such as Cordova and Ionic. With React Native, we're writing entire applications using a single programming language: JavaScript. We can reuse a lot of code, thereby reducing the time it takes to ship a cross-platform application. And hiring and finding quality JavaScript developers is much easier and cheaper than hiring Java, Objective C, or Swift developers, leading to an overall less-expensive process.

> **NOTE** React Native applications are built using JavaScript and JSX. We'll discuss JSX in depth in this book, but for now think of it as a JavaScript syntax extension that looks like HTML or XML.

We'll dive much deeper into React in chapter 2. Until then, let's touch on a few core concepts as an introduction.

1.1.1 A basic React class

Components are the building blocks of a React or React Native application. The entry point of an application is a component that requires and is made of other components. These components may also require other components, and so on.

There are two main types of React Native components: *stateful* and *stateless*. Here's an example of a stateful component using an ES6 class:

```
class HelloWorld extends React.Component {
  constructor() {
    super()
    this.state = { name: 'Chris' }
  }

  render () {
    return (
      <SomeComponent />
    )
  }
}
```

And here's an example of a stateless component:

```
const HelloWorld = () => (
  <SomeComponent />
)
```

The main difference is that stateless components don't hook into any lifecycle methods and hold no state of their own, so any data to be rendered must be received as properties (props). We'll go through the lifecycle methods in depth in chapter 2, but for now let's take a first look at them and look at a class.

Listing 1.1 Creating a basic React Native class

```
import React from 'react'
import { View, Text, StyleSheet } from 'react-native'

class HelloWorld extends React.Component {
  constructor () {
    super()
    this.state = {
      name: 'React Native in Action'
    }
  }
  componentDidMount () {
    console.log('mounted..')
  }
  render () {
    return (
      <View style={styles.container}>
        <Text>{this.state.name}</Text>
      </View>
    )
  }
```

Constructor sets a state object with a name property

Final lifecycle method

Calls render()

```
}

const styles = StyleSheet.create({
  container: {
    marginTop: 100,
    flex: 1
  }
})
```

> **NOTE** Something to keep in mind when we discuss the following methods is the concept of *mounting*. When a component is created, the React component lifecycle is instantiated, triggering the methods used in listing 1.1.

At the top of the file, you require `React` from `'react'`, as well as `View`, `Text`, and `StyleSheet` from `'react-native'`. `View` is the most fundamental building block for creating React Native components and the UI in general and can be thought of like a `div` in HTML. `Text` allows you to create text elements and is comparable to a `span` tag in HTML. `StyleSheet` lets you create style objects to use in an application. These two packages (`react` and `react-native`) are available as npm modules.

When the component first loads, you set a `state` object with the property `name` in the constructor. For data in a React Native application to be dynamic, it needs to be either set in the state or passed down as props. Here, you set the state in the constructor and can therefore change it if desired by calling

```
this.setState({
  name: 'Some Other Name'
})
```

which rerenders the component. Setting the variable in state allows you to update the value elsewhere in the component.

`render` is then called: it examines the props and state and then must return a single React Native element, `null`, or `false`. If you have multiple child elements, they must be wrapped in a parent element. Here, the components, styles, and data are combined to create what will be rendered to the UI.

The final method in the lifecycle is `componentDidMount`. If you need to do any API calls or AJAX requests to reset the state, this is usually the best place to do so. Finally, the UI is rendered to the device, and you can see the result.

1.1.2 React lifecycle

When a React Native class is created, methods are instantiated that you can hook into. These methods are called *lifecycle methods*, and we'll cover them in depth in chapter 2. The methods in listing 1.1 are `constructor`, `componentDidMount`, and `render`, but there are a few more, and they all have their own use cases.

Lifecycle methods happen in sync and help manage the state of components as well as execute code at each step of the way, if you wish. The only required lifecycle method is `render`; all the others are optional. When working with React Native, you're fundamentally working with the same lifecycle methods and specifications you'd use with React.

1.2 What you'll learn

In this book, we'll cover everything you need to know to build robust mobile applications for iOS and Android using the React Native framework. Because React Native is built using the React library, we'll begin in chapter 2 by covering and thoroughly explaining how React works.

We'll then cover styling, touching on most of the styling properties available in the framework. Because React Native uses flexbox for laying out the UI, we'll dive deep into how flexbox works and discuss all the flexbox properties. If you've used flexbox in CSS for layout on the web, all of this will be familiar to you, but keep in mind that the flexbox implementation used by React Native isn't 100% the same.

We'll then go through many of the native components that come with the framework out of the box and walk through how each of them works. In React Native, a component is basically a chunk of code that provides a specific functionality or UI element and can easily be used in the application. Components are covered extensively throughout this book because they're the building blocks of a React Native application.

There are many ways to implement navigation, each with its own nuances, pros, and cons. We'll discuss navigation in depth and cover how to build robust navigation using the most important of the navigation APIs. We'll cover not only the native navigation APIs that come out of the box with React Native, but also a couple of community projects available through npm.

Next, we'll discuss in depth both cross-platform and platform-specific APIs available in React Native and how they work. It will then be time for you to start working with data using network requests, AsyncStorage (a form of local storage), Firebase, and WebSocket. Then we'll dive into the different data architectures and how each of them works to handle the state of the application. Finally, we'll look at testing and a few different ways to test in React Native.

1.3 What you should know

To get the most out of this book, you should have beginner to intermediate knowledge of JavaScript. Much of your work will be done with the command line, so a basic understanding of how to use the command line is also needed. You should also understand what npm is and how it works on at least a fundamental level. If you'll be building in iOS, a basic understanding of Xcode is beneficial and will speed things along but isn't required. Similarly, if you're building for Android, a basic understanding of Android Studio will be beneficial but not required.

Fundamental knowledge of newer JavaScript features implemented in the ES2015 release of the JavaScript programming language is beneficial but not necessary. Some conceptual knowledge of MVC frameworks and single-page architecture is also good but not required.

1.4 Understanding how React Native works

Let's look at how React Native works by discussing JSX, the threading model, React, unidirectional data flow, and more.

1.4.1 JSX

React and React Native both encourage the use of JSX. JSX is basically a syntax extension to JavaScript that looks similar to XML. You can build React Native components without JSX, but JSX makes React and React Native a lot more readable and easier to maintain. JSX may seem strange at first, but it's extremely powerful, and most people grow to love it.

1.4.2 Threading

All JavaScript operations, when interacting with the native platform, are done in a separate thread, allowing the user interface as well as any animations to perform smoothly. This thread is where the React application lives, and where all API calls, touch events, and interactions are processed. When there's a change to a native-backed component, updates are batched and sent to the native side. This happens at the end of each iteration of the event loop. For most React Native applications, the business logic runs on the JavaScript thread.

1.4.3 React

A great feature of React Native is that it uses React. React is an open source JavaScript library that's also backed by Facebook. It was originally designed to build applications and solve problems on the web. This framework has become extremely popular since its release, with many established companies taking advantage of its quick rendering, maintainability, and declarative UI, among other things.

Traditional DOM manipulation is slow and expensive in terms of performance and should be minimized. React bypasses the traditional DOM with something called the *virtual DOM*: basically, a copy of the actual DOM in memory that only changes when comparing new versions of the virtual DOM to old versions of the virtual DOM. This minimizes the number of DOM operations required to achieve the new state.

1.4.4 Unidirectional data flow

React and React Native emphasize unidirectional, or one-way, data flow. Because of how React Native applications are built, this one-way data flow is easy to achieve.

1.4.5 Diffing

React takes the idea of diffing and applies it to native components. It takes your UI and sends the smallest amount of data to the main thread to render it with native components. The UI is declaratively rendered based on the state, and React uses diffing to send the necessary changes over the bridge.

1.4.6 Thinking in components

When building a UI in React Native, it's useful to think of your application as being composed of a collection of components. Thinking about how a page is set up, you already do this conceptually, but using concepts, names, or class names like *header*, *footer*, *body*, *sidebar*, and so on. With React Native, you can give these components names that make sense to you and other developers who may be using your code, making it easy to bring new people into a project or hand a project off to someone else.

Suppose a designer has handed you the example mockup shown in figure 1.1. Let's think of how to conceptualize this into components.

The first thing to do is to mentally break the UI elements into what they represent. The example mockup has a header bar, and within the header bar are a title and a

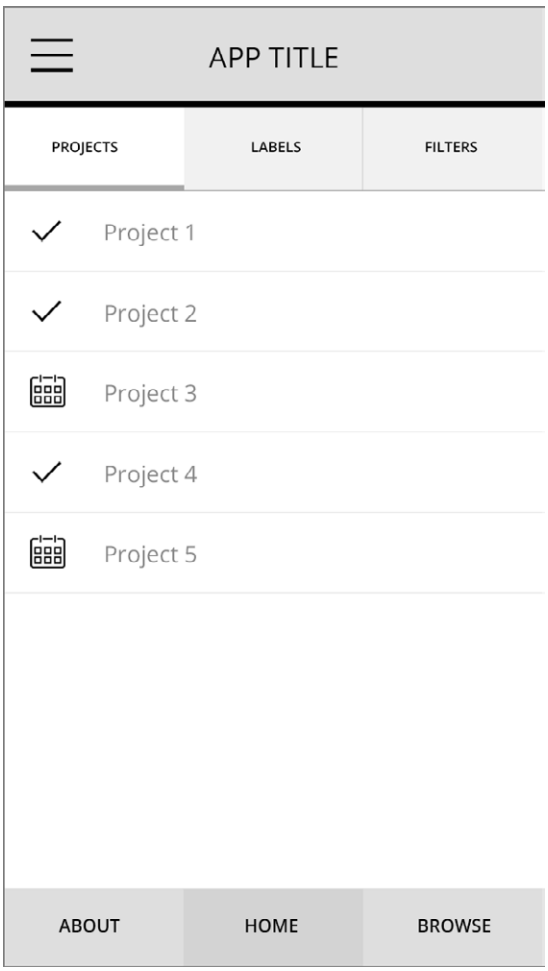

Figure 1.1 Example app design