O'REILLY®

Learning React Native

BUILDING NATIVE MOBILE APPS WITH JAVASCRIPT

Bonnie Eisenman

O'REILLY

Learning React Native

Get a practical introduction to React Native, the JavaScript framework for writing and deploying fully featured mobile apps that look and feel native. With this hands-on guide, you'll learn how to build applications that target iOS, Android, and other mobile platforms instead of browsers. You'll also discover how to access platform features such as the camera, user location, and local storage.

With code examples and step-by-step instructions, author Bonnie Eisenman shows web developers and frontend engineers how to build and style interfaces, use mobile components, and debug and deploy apps. Along the way, you'll build several increasingly sophisticated example apps with React Native before putting everything together at the end.

- Learn how React Native provides an interface to native UI components
- Examine how the framework uses native components analogous to HTML elements
- Create and style your own React Native components and applications
- Install modules for APIs and features not supported by the framework
- Get tools for debugging your code, and for handling issues outside of JavaScript
- Put it all together with the Zebreto effective-memorization flashcard app
- Deploy apps to the iOS App Store and Google Play Store

Bonnie Eisenman is a software engineer at Twitter with previous experience at Codecademy, Google, and Fog Creek Software. She has spoken at several conferences on topics ranging from Scala functional programming and Arduino

"Bonnie does a fantastic job at covering the most important topics for building a solid foundation of knowledge in React Native and, if you put in the work, the book will leave you confident to go out exploring on your own to learn even more."

—Brent Vatne
Developer at Exponent,
Core Contributor to React Native

JAVASCRIPT

US $39.99 CAN $45.99

ISBN: 978-1-491-92900-1

Twitter: @oreillymedia
facebook.com/oreilly

Learning React Native

Building Mobile Applications with JavaScript

Bonnie Eisenman

Beijing • Boston • Farnham • Sebastopol • Tokyo O'REILLY®

Learning React Native
by Bonnie Eisenman

Printed in the United States of America.

Published by O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472.

O'Reilly books may be purchased for educational, business, or sales promotional use. Online editions are also available for most titles (*http://safaribooksonline.com*). For more information, contact our corporate/institutional sales department: 800-998-9938 or *corporate@oreilly.com*.

Editor: Meg Foley
Production Editor: Nicholas Adams
Copyeditor: Jasmine Kwityn
Proofreader: Christina Edwards
Indexer: Ellen Troutman-Zaig
Interior Designer: David Futato
Cover Designer: Randy Comer
Illustrator: Rebecca Demarest

December 2015: First Edition

Revision History for the First Edition
2015-12-01: First Release

See *http://oreilly.com/catalog/errata.csp?isbn=9781491929001* for release details.

978-1-491-92900-1

[LSI]

Table of Contents

Preface

This book is an introduction to React Native, Facebook's JavaScript framework for building mobile applications. Using your existing knowledge of JavaScript and React, you'll be able to build and deploy fully featured mobile applications for both iOS and Android that truly render natively. Just because it's JavaScript doesn't mean we should settle for less. There are plenty of advantages to working with React Native over traditional means of mobile development, and we don't need to sacrifice the native look and feel.

We'll start with the basics, and work our way up to deploying a full-fledged application to both the iOS App Store and the Google Play Store, with 100% code reuse between the two platforms. In addition to the essentials of the framework, we'll discuss how to work beyond it, including how to make use of third-party libraries and even how to write your own Java or Objective-C libraries to extend React Native.

If you're coming to mobile development from the perspective of a frontend software engineer or web developer, this is the book for you. React Native is a pretty amazing thing, and I hope you're as excited to explore it as I am!

Prerequisites

This book is not an introduction to React, in general. We'll assume that you have some working knowledge of React. If you're brand new to React, I suggest reading through a tutorial or two before coming back to take the plunge into mobile development. Specifically, you should be familiar with the role of props and state, the component lifecycle, and how to create React components.

We'll also be using some ES6 syntax, as well as JSX. If you aren't familiar with these, don't worry; we'll cover JSX in Chapter 2, and ES6 syntax in Appendix A. These features are essentially 1:1 translations of the JavaScript code you're already accustomed to writing.

This book assumes you are developing on OS X. Developing on OS X is a requirement for writing iOS apps. Linux and Windows support for writing Android applications is a work-in-progress. You can read more about Linux and Android support here.

Conventions Used in This Book

The following typographical conventions are used in this book:

Italic
: Indicates new terms, URLs, email addresses, filenames, and file extensions.

`Constant width`
: Used for program listings, as well as within paragraphs to refer to program elements such as variable or function names, databases, data types, environment variables, statements, and keywords.

`Constant width bold`
: Shows commands or other text that should be typed literally by the user.

`Constant width italic`
: Shows text that should be replaced with user-supplied values or by values determined by context.

This element signifies a tip or suggestion.

This element signifies a general note.

This element indicates a warning or caution.

Using Code Examples

Supplemental material (code examples, exercises, etc.) is available for download at: *https://github.com/bonniee/learning-react-native*.

This book is here to help you get your job done. In general, if example code is offered with this book, you may use it in your programs and documentation. You do not need to contact us for permission unless you're reproducing a significant portion of the code. For example, writing a program that uses several chunks of code from this book does not require permission. Selling or distributing a CD-ROM of examples from O'Reilly books does require permission. Answering a question by citing this book and quoting example code does not require permission. Incorporating a significant amount of example code from this book into your product's documentation does require permission.

We appreciate, but do not require, attribution. An attribution usually includes the title, author, publisher, and ISBN. For example: "*Learning React Native* by Bonnie Eisenman (O'Reilly). Copyright 2016 Bonnie Eisenman, 978-1-491-92900-1."

If you feel your use of code examples falls outside fair use or the permission given above, feel free to contact us at *permissions@oreilly.com*.

Safari® Books Online

Safari Books Online is an on-demand digital library that delivers expert content in both book and video form from the world's leading authors in technology and business.

Technology professionals, software developers, web designers, and business and creative professionals use Safari Books Online as their primary resource for research, problem solving, learning, and certification training.

Safari Books Online offers a range of plans and pricing for enterprise, government, education, and individuals.

Members have access to thousands of books, training videos, and prepublication manuscripts in one fully searchable database from publishers like O'Reilly Media, Prentice Hall Professional, Addison-Wesley Professional, Microsoft Press, Sams, Que, Peachpit Press, Focal Press, Cisco Press, John Wiley & Sons, Syngress, Morgan Kaufmann, IBM Redbooks, Packt, Adobe Press, FT Press, Apress, Manning, New Riders, McGraw-Hill, Jones & Bartlett, Course Technology, and hundreds more. For more information about Safari Books Online, please visit us online.

How to Contact Us

Please address comments and questions concerning this book to the publisher:

O'Reilly Media, Inc.
1005 Gravenstein Highway North
Sebastopol, CA 95472
800-998-9938 (in the United States or Canada)
707-829-0515 (international or local)
707-829-0104 (fax)

We have a web page for this book, where we list errata, examples, and any additional information. You can access this page at *http://bit.ly/learning-react-native*.

To comment or ask technical questions about this book, send email to *bookquestions@oreilly.com*.

For more information about our books, courses, conferences, and news, see our website at *http://www.oreilly.com*.

Find us on Facebook: *http://facebook.com/oreilly*

Follow us on Twitter: *http://twitter.com/oreillymedia*

Watch us on YouTube: *http://www.youtube.com/oreillymedia*

Resources

It's dangerous to go alone! Well, not really, but that doesn't mean you have to. Here are some resources you may find useful as you work through the book:

- The GitHub repository for this book contains all of the code samples we'll be discussing. If you get stumped, or want more context, try looking here first.
- Join the mailing list at LearningReactNative.com for follow-up articles, suggestions, and helpful resources.
- The official documentation has a lot of good reference material.

Additionally, the React Native community is a useful resource:

- Brent Vatne's React Native newsletter
- The react-native tag on Stack Overflow
- #reactnative (*irc://chat.freenode.net/reactnative*) on Freenode

Acknowledgments

As is traditional: this book would not have been possible without the help and support of many others. Thank you to my editor, Meg Foley, and the rest of the O'Reilly team, for bringing this project into the world. Thank you also to my technical reviewers, for your time and insightful feedback: David Bieber, Jason Brown, Erica Portnoy, and Jonathan Stark. I would also like to thank the React Native team, without whose stellar work this book would naturally be impossible. Thanks also to Zachary Elliot for his help with the Zebreto application and Android in general.

And many thanks are owed to my dear friends, who put up with me throughout this process and provided moral support, guidance, and distraction, as the situation required. Thank you.

CHAPTER 1

What Is React Native?

React Native is a JavaScript framework for writing real, natively rendering mobile applications for iOS and Android. It's based on React, Facebook's JavaScript library for building user interfaces, but instead of targeting the browser, it targets mobile platforms. In other words: web developers can now write mobile applications that look and feel truly "native," all from the comfort of a JavaScript library that we already know and love. Plus, because most of the code you write can be shared between platforms, React Native makes it easy to simultaneously develop for both Android and iOS.

Similar to React for the Web, React Native applications are written using a mixture of JavaScript and XML-esque markup, known as JSX. Then, under the hood, the React Native "bridge" invokes the native rendering APIs in Objective-C (for iOS) or Java (for Android). Thus, your application will render using real mobile UI components, *not* webviews, and will look and feel like any other mobile application. React Native also exposes JavaScript interfaces for platform APIs, so your React Native apps can access platform features like the phone camera, or the user's location.

React Native currently supports both iOS and Android, and has the potential to expand to future platforms as well. In this book, we'll cover both iOS and Android. The vast majority of the code we write will be cross-platform. And yes: you can really use React Native to build production-ready mobile applications! Some anecdota: Facebook, Palantir, and TaskRabbit are already using it in production for user-facing applications.

Advantages of React Native

The fact that React Native actually renders using its host platform's standard rendering APIs enables it to stand out from most existing methods of cross-platform appli-

cation development, like Cordova or Ionic. Existing methods of writing mobile applications using combinations of JavaScript, HTML, and CSS typically render using webviews. While this approach can work, it also comes with drawbacks, especially around performance. Additionally, they do not usually have access to the host platform's set of native UI elements. When these frameworks do try to mimic native UI elements, the results usually "feel" just a little off; reverse-engineering all the fine details of things like animations takes an enormous amount of effort, and they can quickly become out of date.

In contrast, React Native actually translates your markup to real, native UI elements, leveraging existing means of rendering views on whatever platform you are working with. Additionally, React works separately from the main UI thread, so your application can maintain high performance without sacrificing capability. The update cycle in React Native is the same as in React: when `props` or `state` change, React Native re-renders the views. The major difference between React Native and React in the browser is that React Native does this by leveraging the UI libraries of its host platform, rather than using HTML and CSS markup.

For developers accustomed to working on the Web with React, this means you can write mobile apps with the performance and look and feel of a native application, while using familiar tools. React Native also represents an improvement over normal mobile development in two other areas: the developer experience and cross-platform development potential.

Developer Experience

If you've ever developed for mobile before, you might be surprised by how easy React Native is to work with. The React Native team has baked strong developer tools and meaningful error messages into the framework, so working with robust tools is a natural part of your development experience.

For instance, because React Native is "just" JavaScript, you don't need to rebuild your application in order to see your changes reflected; instead, you can hit Command+R to refresh your application just as you would any other web page. All of those minutes spent waiting for your application to build can really add up, and in contrast React Native's quick iteration cycle feels like a godsend.

Additionally, React Native lets you take advantage of intelligent debugging tools and error reporting. If you are comfortable with Chrome or Safari's developer tools (Figure 1-1), you will be happy to know that you can use them for mobile development, as well. Likewise, you can use whatever text editor you prefer for JavaScript editing: React Native does not force you to work in Xcode to develop for iOS, or Android Studio for Android development.

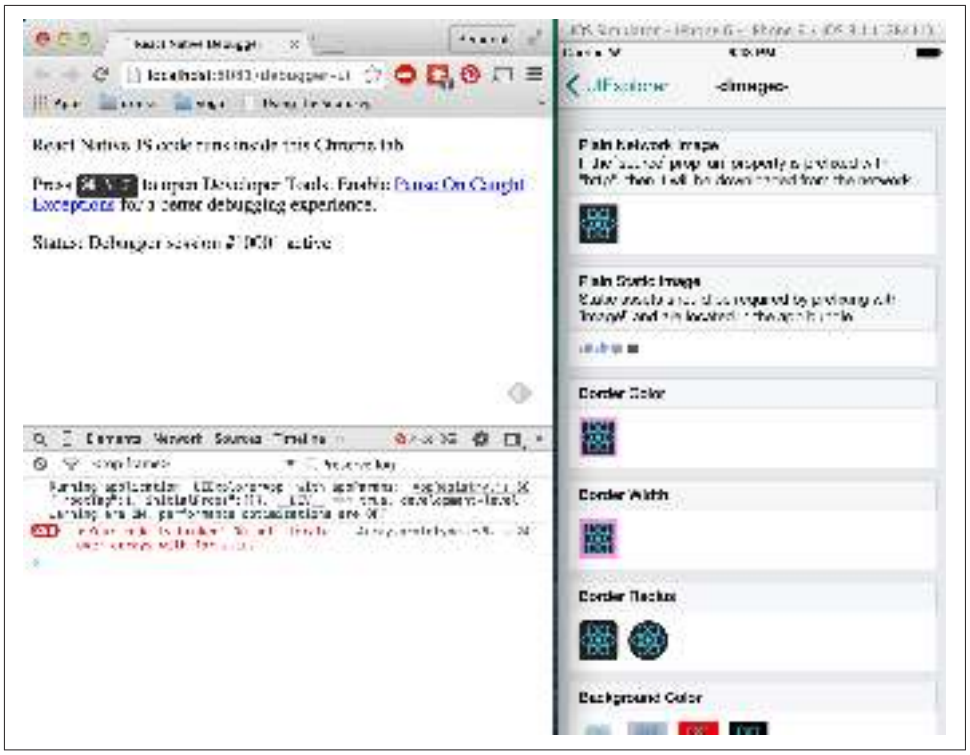

Figure 1-1. Using the Chrome Debugger

Besides the day-to-day improvements to your development experience, React Native also has the potential to positively impact your product release cycle. For instance, Apple permits JavaScript-based changes to an app's behavior to be loaded over the air with no additional review cycle necessary.

All of these small perks add up to saving you and your fellow developers time and energy, allowing you to focus on the more interesting parts of your work and be more productive overall.

Code Reuse and Knowledge Sharing

Working with React Native can dramatically shrink the resources required to build mobile applications. Any developer who knows how to write React code can now target the Web, iOS, and Android, all with the same skillset. By removing the need to "silo" developers based on their target platform, React Native lets your team iterate more quickly, and share knowledge and resources more effectively.

Besides shared knowledge, much of your code can be shared, too. Not *all* the code you write will be cross-platform, and depending on what functionality you need on a

specific platform, you may occasionally need to dip into Objective-C or Java. (Happily, this isn't too bad, and we'll cover how so-called native modules work in Chapter 7.) But reusing code across platforms is surprisingly easy with React Native. For example, the Facebook Ads Manager application for Android shares 87% of its codebase with the iOS version, as noted in the React Europe 2015 keynote. The final application we'll look at in this book, a flashcard app, has total code reuse between Android and iOS. It's hard to beat that!

Risks and Drawbacks

As with anything, using React Native is not without its downsides, and whether or not React Native is a good fit for your team really depends on your individual situation.

The largest risk is probably React Native's maturity, as the project is still relatively young. iOS support was released in March 2015, and Android support was released in September 2015. The documentation certainly has room for improvement, and continues to evolve. Some features on iOS and Android still aren't supported, and the community is still discovering best practices. The good news is that in the vast majority of cases, you can implement support for missing APIs yourself, which we'll cover in Chapter 7.

Because React Native introduces another layer to your project, it can also make debugging hairier, especially at the intersection of React and the host platform. We'll cover debugging for React Native in more depth in Chapter 8, and try to address some of the most common issues.

React Native is still young, and the usual caveats that go along with working with new technologies apply here. Still, on the whole, I think you'll see that the benefits outweigh the risks.

Summary

React Native is an exciting framework that enables web developers to create robust mobile applications using their existing JavaScript knowledge. It offers faster mobile development, and more efficient code sharing across iOS, Android, and the Web, without sacrificing the end user's experience or application quality. The tradeoff is that it's new, and still a work in progress. If your team can handle the uncertainty that comes with working with a new technology, and wants to develop mobile applications for more than just one platform, you should be looking at React Native.

In the next chapter, we'll go over some of the main ways in which React Native differs from React for the Web, and cover some key concepts. If you'd like to skip straight to

developing, feel free to jump to Chapter 3, in which we'll handle setting up our development environment and write our very first React Native application.

CHAPTER 2

Working with React Native

In this chapter, we'll cover the "bridge," and review how React Native works under the hood. Then, we'll look at how React Native components differ from their web counterparts, and cover what you'll need to know in order to create and style components for mobile.

If you'd prefer to dig into the development process and see React Native in action, feel free to jump ahead to the next chapter!

How Does React Native Work?

The idea of writing mobile applications in JavaScript feels a little odd. How is it possible to use React in a mobile environment? In order to understand the technical underpinnings of React Native, first we'll need to recall one of React's features, the Virtual DOM.

In React, the Virtual DOM acts as a layer between the developer's description of how things ought to look, and the work done to actually render your application onto the page. To render interactive user interfaces in a browser, developers must edit the browser's DOM, or Document Object Model. This is an expensive step, and excessive writes to the DOM have a significant impact on performance. Rather than directly render changes on the page, React computes the necessary changes by using an in-memory version of the DOM, and rerenders the minimal amount necessary. Figure 2-1 shows how this works.

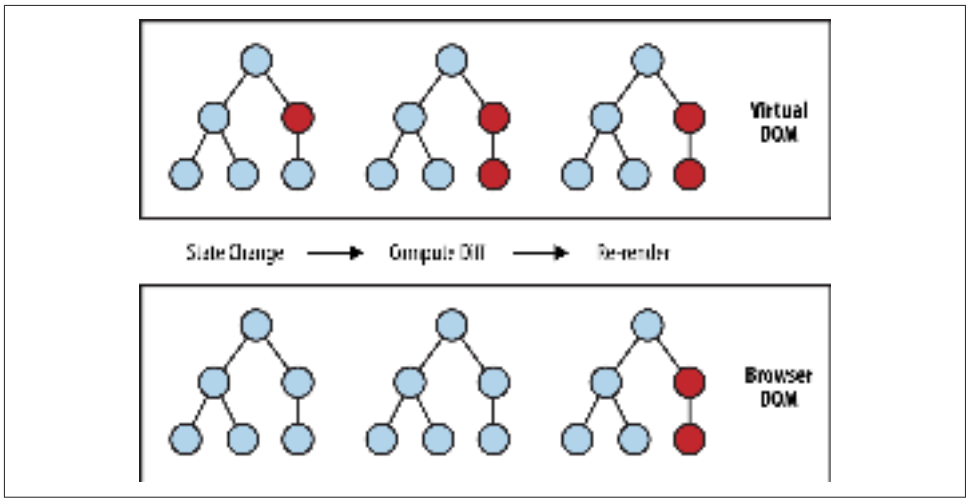

Figure 2-1. Performing calculations in the Virtual DOM limits rerendering in the Browser DOM

In the context of React on the Web, most developers think of the Virtual DOM primarily as a performance optimization. The Virtual DOM certainly has performance benefits, but its real potential lies in the power of its abstraction. Placing a clean abstraction layer between the developer's code and the actual rendering opens up a lot of interesting possibilities. What if React could render to a target other than the browser's DOM? After all, React already "understands" what your application is *supposed* to look like.

Indeed, this is how React Native works, as shown in Figure 2-2. Instead of rendering to the browser's DOM, React Native invokes Objective-C APIs to render to iOS components, or Java APIs to render to Android components. This sets React Native apart from other cross-platform app development options, which often end up rendering web-based views.

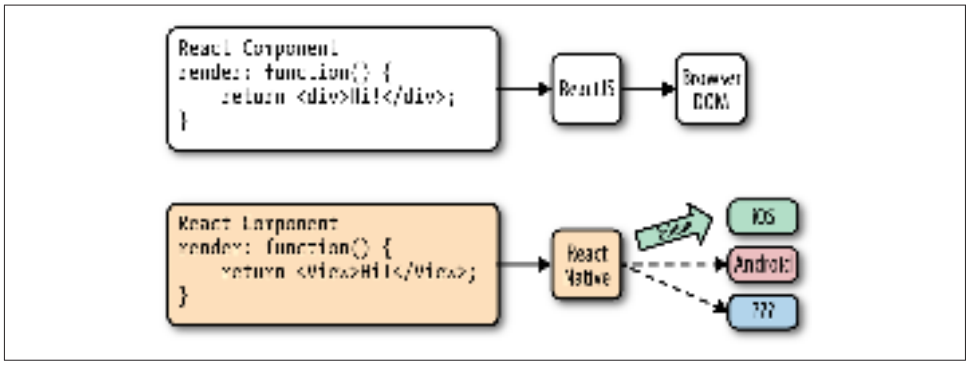

Figure 2-2. React can render to different targets