Extracted from:

RubyMotion

iOS Development with Ruby

The Pragmatic Bookshelf

Dallas, Texas • Raleigh, North Carolina

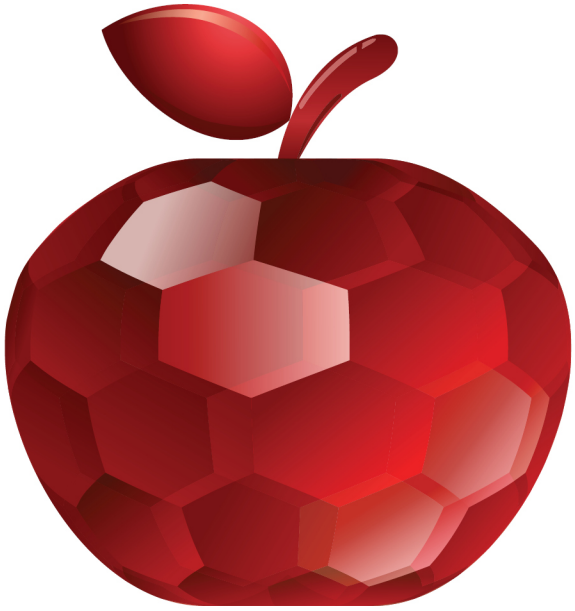
The Pragmatic Programmers

Pragmatic exPress

RubyMotion

iOS Development with Ruby

Updated for RubyMotion 2



Clay Allsopp

Foreword by Laurent Sansonetti,
lead developer of RubyMotion

edited by Fahmida Y. Rashid

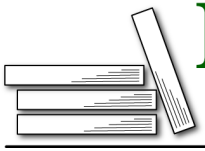
RubyMotion

iOS Development with Ruby

Clay Allsopp

The Pragmatic Bookshelf

Dallas, Texas • Raleigh, North Carolina



Our Pragmatic courses, workshops, and other products can help you and your team create better software and have more fun. For more information, as well as the latest Pragmatic titles, please visit us at *http://pragprog.com*.

The team that produced this book includes:

Fahmida Y. Rashid (editor)
Kim Wimpsett (copyeditor)
David J. Kelly (typesetter)
Janet Furlow (producer)
Ellie Callahan (support)

For international rights, please contact *rights@pragprog.com*.

Printed in the United States of America.
ISBN-13: 978-1-937785-28-4
Encoded using the finest acid-free high-entropy binary digits.
Book version: P3.0—July 2014

Ruby on iOS

The iPhone and iOS exceeded everyone's initial expectations. In the past five years, independent developers and companies have published more than *half a million* products to the App Store that have been downloaded more than *two billion* times. But despite the huge influx of new developers and programming resources, the process of building iOS apps has remained fundamentally unchanged.

The iOS SDK was first announced in early 2008, nearly a year after the first iPhone debuted. Mac developers felt right at home since it used the same Objective-C/Xcode workflow that had existed on OS X for years. For everyone else, that day was probably the first time they heard the term *Objective-C*.

Objective-C is a robust language, but its verbosity and compiled nature are a bit out of step with the dynamic languages embraced by many of today's developers. Since Objective-C's inception in the 1980s, programmers have shifted toward Perl, PHP, Python, Ruby, and JavaScript. These "scripting" languages allowed some of the biggest websites in the world to grow and iterate with unparalleled speed by empowering flexibility and reducing complexity.

So, why haven't we seen these languages prosper on mobile yet? I mean, that is why you're here, right? The answer is that there have been no alternatives to Objective-C that allow for the trademark iOS user experience without compromising performance...well, no alternatives until now.

Hello, RubyMotion

RubyMotion (http://rubymotion.com) is such an alternative. Put simply, it allows you to develop iOS apps in Ruby without degrading the app's quality. To accomplish this, RubyMotion compiles your Ruby files to machine code; in contrast to traditional nonmobile Ruby, there's no interpreter or garbage collector to hinder performance. Your Ruby code uses the iOS SDK frameworks

and classes exactly as intended by Apple, so all existing Objective-C code examples and tutorials are perfectly reusable.

Why Ruby instead of Python or some other language? For one, Ruby is already incredibly popular among web developers because of frameworks like Ruby on Rails; for new developers, coding in a familiar language means an easier transition from the Web to mobile. But most importantly, Ruby is a friendlier and more forgiving language for developers at any experience level. Whether you're a Rails veteran or just getting your feet wet with Ruby, this book will give you the foundations to create gorgeous iOS apps with no compromise in performance or developer happiness.

Why RubyMotion?

There are other alternatives to iOS development with Objective-C. HTML-based solutions like PhoneGap (http://phonegap.com) and Trigger (https://trigger.io/) are often attractive because they allow apps to be changed without additional Apple approval. However, this flexibility comes at a cost: the non-native interface elements created with HTML often create a jarring experience for users. Notably, Facebook and LinkedIn have moved away from HTML5 in their iOS apps and migrated to native versions.

Nu (https://github.com/timburks/nu) is the closest counterpart to RubyMotion: instead of using Ruby, it is a Lisp-like language you can use to write truly native iOS applications with Apple's frameworks. If you're a fan of Lisp or other functional languages, then it could be a good fit; however, RubyMotion offers more than a different language.

Unlike these other alternatives, RubyMotion is a complete tool chain that handles the entire process of creating, testing, and deploying iOS apps. Unlike the Xcode-centric Objective-C, RubyMotion development uses command-line tools such as Rake and Cocoapods (a popular iOS library manager) to increase the familiarity and ease with which developers can pick up coding for iOS. It also includes an interactive console to debug your apps and a robust, RSpec-like testing framework. No other tool or framework possesses this level of end-to-end integration for iOS development.

Reading This Book

The best programming books hit the ground running, and that's just what we'll do. Each chapter will introduce one concept and build a sample application around it. This book is intended to be read sequentially: every subsequent chapter builds on what we covered in the previous. We'll start off with the

basics, such as how to draw boxes on the screen, but in just a handful of chapters we'll be interacting with an HTTP API. That means we move fast and cover just the essentials.

This isn't a reference manual for iOS development; there are many other great and extensive resources on iOS development, including Apple's official documentation. Instead, this book will get your feet just wet enough in the major topics of native iOS development so you can understand and research new information on your own.

Before we begin, you should be aware of some requirements. Since we plan on moving fast, this book assumes you're familiar with Ruby. If you haven't played around with Ruby, check out a book like *Learn to Program [Pin06]* before diving into RubyMotion. RubyMotion is currently a commercial product from HipByte; that means if you want to play, you have to pay for a license. Additionally, the RubyMotion tools work only on OS X 10.7 or newer. The iOS SDK ships with a fast desktop simulator, so you won't need a physical iOS device to test your projects on.

Online Resources

What would a modern programming book be if we didn't help you outside of the text? You should check out this book's web page (http://pragprog.com/book/carubym/rubymotion) for updates and a discussion board; you also will find all of the source code for the examples used in this book. If you're reading this book electronically, you can click the little gray box above each code sample to download it immediately.

Additionally, RubyMotion has a very vibrant community that can provide guidance. The RubyMotion Developer Center[1] has more in-depth articles on many aspects of RubyMotion that we may cover only briefly. There's also the RubyMotion user group,[2] a great place to ask specific questions and get involved.

Acknowledgments

This book was definitely not a one-man effort. First, a big thanks to the whole team at HipByte for providing support and creating the great product that is RubyMotion. I'd also like to give several high-fives to Laurent Sansonetti, Christopher Adams, David Astels, Joel Clermont, Jeff Holland, Ethan Sherbondy, Mattt Thompson, Justin de Vesine, Colin Thomas-Arnold, and Mike

1. http://www.rubymotion.com/developer-center/
2. https://groups.google.com/group/rubymotion

Clark for reviewing drafts of this book and making sure all the technical aspects were kosher. Additionally, I believe I owe Fahmida Y. Rashid a few cups of coffee for making this whole thing happen. And finally, a warm hug to my mom and dad for letting me do all that crazy computer stuff when I was younger.

So, what are we waiting for? Let's start making our first iOS app in Ruby!