MACRUBY IN ACTION Pdf Free

Announcing MacRuby In Action. Chapter 1. Introducing MacRuby – MacRuby in Action. Verify your identity.

- Brendan Lim
- 300 pages
- 27 Apr 2012
- Manning Publications
- 9781935182498
- English
- New York, United States

MacRuby in Action [Book]

Lim is a professional web and mobile developer. He is a Y Combinator alum and is the co-founder of Kicksend, a mobile-focused startup that lets people send and print photo albums. Brendan is the author of MacRuby in Action. Jeremy McAnally has been programming for about eight years and doing graphic design for four years. He is curerntly a freelance Ruby and Rails developer, consultant, and author. He has over three years' experience with Ruby and two years' with Rails; in that time has has developed a number of small, localized intranet systems and mediumt- large-scale systems in Ruby.

Get our latest book recommendations, author news, and competitions right to your inbox. Tell us what you like and we'll recommend books you'll love. Join our mailing list! Published by Manning. About The Book. About The Authors. We touched on enough parts of the Cocoa development framework for you to be familiar with the framework. To learn more about the framework, we recommend looking there first. The goal of this section is to introduce you to both Ruby and Objective-C. Smalltalk is an object-oriented language that pre-dates Ruby and Objective-C. A dynamically typed , reflective programming language, Smalltalk is historically significant because of its influence on languages, such as Objective-C and Ruby. A language is said to be dynamically typed when it does most of its type-checking at runtime rather than during compilation. A reflective language allows you to write code that can effectively observe and modify its own behavior.

At a high level, Objective-C and Ruby share a surprising number of features also found in Smalltalk. Both languages provide a dynamic runtime, use message passing, allow runtime access to class and object information, and support metaclasses, to name a few. Message passing is similar to method calling. The main difference is that message passing requires a lookup to make sure the method exists before jumping to it during runtime, whereas message calling calls the method directly without doing a check. Message-passing support is one of the reasons that languages such as Smalltalk, Objective-C, and Ruby are considered dynamic. One of the most significant similarities between Ruby and Objective-C is the dynamic runtime, which makes it much easier to model Ruby in Objective-C. Sharing a dynamic runtime allows MacRuby to fully support both standard Ruby and Objective-C functionality.

You get two for the price of one. Also, the dynamic nature that makes Ruby so flexible is available in MacRuby. The ability to make runtime class modifications and handle message redirection both of which could be considered advanced programming topics are also key features of the Ruby language. Objective-C was developed as an extension to the C language. Unlike Ruby, Objective-C is a compiled language. Xcode 4 comes with Interface Builder built in, which allows you to create user interfaces for your Cocoa applications. The message-passing syntax is the most important Objective-C concept you need to know.

First, create an instance of an array:. In the next line, you call the instance method, init , on the resulting NSArray object. The important thing to take away from this example is the message-passing syntax. In Objective-C, you can invoke a method on an object using the bracket syntax you just used. To illustrate how the syntax works, consider the following code:. In this example, someObject is the receiver of the

message, someMethod. The result of this message then becomes the receiver of the message, anotherMethod. A parameter is denoted by a colon at the end of its name. The following method has five parameters:. Yes, that whole line is the name of one single method. The parameters and colons make up the name of the function. You first call a class method on the NSTimer class. As you can see, NSTimer is the receiver of the method call. You then call the method that we just described, but this time using real parameters:. Objective-C supports dynamic message-passing using something called a method selector.

In strict terms, a method selector can be thought of in two ways. Second, the method selector is the unique identifier that replaces the method name when you reference the method. Simply put, selectors allow you to dynamically denote method names and identifiers. You can use a selector to invoke an SEL method on an object. This technique supports the target-action design pattern used heavily in Cocoa development.

In code, you can refer to a compiled selector using the selector directive. The most common use for this is when you invoke a method that expects a callback method as a parameter. To specify which method should be invoked, you use the selector reference. Look again at the method signature for the NSTimer example, and examine the selector parameter:. Remember the second parameter, target? You pass in self , which means that the object from which you call this function is the target for the doSomethingOnTimer selector. The combination of the target: and selector: parameters is the equivalent of having the timer run this line of code:. Selectors are a powerful feature when implementing dynamic functionality in your applications. When working with selectors, be aware of a common error. Consider the following two selector directives:. Notice the difference?

The second has a trailing colon, which is a critical distinction. A selector with no trailing colon indicates a method with no parameters, whereas a selector with a trailing colon indicates a method with one parameter. Creating classes in Objective-C requires two files. One file acts as the interface and the other acts as the implementation. The interface file, typically saved with a. The implementation file, saved with a. You first import the Cocoa framework to pull in the resources you need for a Cocoa application. Inside the curly braces, you define two instance variables, which are of class NSString. The next two lines outside the curly braces are getter methods for the instance variables, title , and authors. Notice how NSString is inside parentheses on these two lines? In the implementation of the Book class, you first import the interface file you created. You then declare the implementation and specify the class name, Book.

The next lines are custom getters for the instance variables, implemented in a simple fashion. You return a new NSString for each method. Manual memory management can certainly be a big barrier to entry for those who are new to iOS development. Frameworks are collections of classes that provide a set of functionality. Frameworks is the term Cocoa uses for libraries. Frameworks provide an efficient code-organization mechanism for separation of disparate functionality. As you make use of different OS X capabilities in your applications, the variety of frameworks that Apple

provides will come in handy. The Apple developer documentation is a great tool for learning about the methods available on any of the Cocoa classes. Figure 1. In figure 1. Use the search bar to search for common classes. Ruby is a concise and simple language.

Ruby classes are defined in a single class file ending with a. Typically, to run a Ruby script, you use the Ruby executable ruby to run the script. You can use any text editor to write and edit Ruby source files. At the least, we recommend using a text editor that supports syntax highlighting for Ruby which will work for most MacRuby syntax as well. Ruby is all about objects. Everything in Ruby is an object, and we mean everything! In Ruby, when you type 1 , you have an object of class Fixnum , and you can call methods directly on what looks like a literal. To get a feel for Ruby, try out a few more things in Irb right now. Use the method, methods , to see what methods are available on an object; use the method, class , to see what the class of an object is.

Explore a little, and call some methods on the string "Ruby" to see what you get. Ruby is a flexible language. So what, right? Many programming languages are flexible. What makes Ruby stand out is that the language provides more than one way to do something, even important things like calling methods. Consider the most common operation you can do in Ruby: calling methods. There are a few ways you can do this.

Suppose you have a string, and you want to reverse the characters in it. You can use the String reverse method to accomplish this. In message-passing terms, the string racecar is the receiver and the method reverse is the message. All three of these statements are equivalent:. The second option is the most typically used way to call methods. In Ruby, parentheses are optional on method calls, and, often, you need to use them only to help with readability when you make nested method calls. Ruby uses a dot notation for method calls that looks like this: receiver. Determining what self is in any given context can be confusing, although the most common cases are either the class or instance context for a given object.

This example calls the kernel method puts to output a string to stdout :. You can call many kernel methods without an explicit receiver. The most common kernel methods are used for raising exceptions, requiring other Ruby source files, and accessing metaprogramming functionality, such as creating anonymous functions. Ruby classes are simple to define. The result is that you have getter and setter behavior on Book instances for those two attributes and a local variable for each in the Book class.

You can take an instance of Book and call book. This is the method used to initialize the instance of the object. Here you implement an initialize method that sets the title and price of the Book variables to the values that are passed in. Methods are defined using def , which is given the method name, a parameter list, and a block that defines the functionality of the method. This is an instance method available to instances of the Book class. If you instantiate a new Book without passing any parameters to new , you get the following error:. When you initialize a new Book , you need to pass in the title and price parameters:. To add amounts to the price, you can use float or integer values.

Take advantage of the plethora of Ruby resources out there because a solid understanding of Ruby will help you write MacRuby applications. Want to get more out of the basic search box? Read about Search Operators for some powerful new tools. Corporate Author: Safari Books Online. Edition: 1st ed. Subjects: Mac OS. Ruby Computer program language. Object-oriented programming Computer science.

Published: Mac OS X Published:

About this Book · MacRuby in Action

If you would like to read more about this check out the Privacy Policy page. Important Message. MacRuby in Action by Brendan G. Macruby In Action by Brendan G. Price may vary. I Add to my wishlist. Overview Readers reviews 0 Product Details. Your rating Click on the stars for rating. Your review Headline characters remaining Review characters remaining. Add to shopping bag. You can also use ILLiad to request chapter scans and articles. Online version. Phrase Searching You can use double quotes to search for a series of words in a particular order. Wildcard Searching If you want to search for multiple variations of a word, you can substitute a special symbol called a "wildcard" for one or more letters. You can use? Advanced Searching Our Advanced Search tool lets you easily search multiple fields at the same time and combine terms in complex ways.

See the help page for more details. Want to get more out of the basic search box? Read about Search Operators for some powerful new tools. Corporate Author: Safari Books Online. Edition: 1st ed. Subjects: Mac OS. Ruby Computer program language.

Holdings: MacRuby :

You'll also pick up high-value techniques including system scripting, automated testing practices, and getting your apps ready for the Mac App Store. MacRuby in Action bridges that gap between two great development communities: Mac desktop development and Ruby. Macruby gives you the full power of the Cocoa framework to help you build native desktop apps that are indistinguishable from their Objective-C counterparts. The Ruby community prides itself on test driven development, and the same practices can be applied to MacRuby.

The book also cover gotchas and other tips to watch out for when bridging between these two technologies. The examples showcase commonly used components and help you get your feet wet in practical development while showing key development concepts. After you've had a chance to digest some of the content, we'd love to hear your thoughts on the book. Leave your comments below or let us know on Twitter! We recommend to install revision of branch 2. Note that compiling LLVM can take quite a while. However, doing this can make your machine unresponsive during compilation. So for a Core Duo the command would look like:. If you would prefer to update an existing LLVM Subversion working copy you need to be careful to remove any previous build files before you compile:. If you use the Xcode 4. Then you should be all set! Please report us any problem you will find the macruby.

Create a pull request: github. Skip to content. View license. Branches Tags. Could not load branches. Could not load tags. Latest commit. Git stats 6, commits.