

Addison-Wesley Professional Ruby Series



# ELOQUENT RUBY

*Foreword by* **Obie Fernandez**, *Series Editor*

**RUSS OLSEN**

## Praise for *Eloquent Ruby*

“Reading *Eloquent Ruby* is like programming in Ruby itself: fun, surprisingly deep, and you’ll find yourself wishing it was always done this way. Wherever you are in your Ruby experience from novice to Rails developer, this book is a must read.”

—Ethan Roberts

Owner, Monkey Mind LLC

“*Eloquent Ruby* lives up to its name. It’s a smooth introduction to Ruby that’s both well organized and enjoyable to read, as it covers all the essential topics in the right order. This is the book I wish I’d learned Ruby from.”

—James Kebinger

Senior Software Engineer, PatientsLikeMe

[www.monkeyatlarge.com](http://www.monkeyatlarge.com)

“Ruby’s syntactic and logical aesthetics represent the pinnacle for elegance and beauty in the ALGOL family of programming languages. *Eloquent Ruby* is the perfect book to highlight this masterful language and Russ’s blend of wit and wisdom is certain to entertain and inform.”

—Michael Fogus

Contributor to the Clojure programming  
language and author of *The Joy of Clojure*

# Eloquent Ruby

## Table of Contents

Contents

Foreword

Preface

Acknowledgments

About the Author

### PART I: The Basics

#### Chapter 1: Write Code That Looks Like Ruby

- The Very Basic Basics

- Go Easy on the Comments

- Camels for Classes, Snakes Everywhere Else

- Parentheses Are Optional but Are Occasionally Forbidden

- Folding Up Those Lines

- Folding Up Those Code Blocks

- Staying Out of Trouble

- In the Wild

- Wrapping Up

#### Chapter 2: Choose the Right Control Structure

- If, Unless, While, and Until

- Use the Modifier Forms Where Appropriate

- Use each, Not for

- A Case of Programming Logic

- Staying Out of Trouble

- In the Wild

# **Table of Contents**

Wrapping Up

## **Chapter 3: Take Advantage of Rubys Smart Collections**

Literal Shortcuts

Instant Arrays and Hashes from Method Calls

Running Through Your Collection

Beware the Bang!

Rely on the Order of Your Hashes

In the Wild

Staying Out of Trouble

Wrapping Up

## **Chapter 4: Take Advantage of Rubys Smart Strings**

Coming Up with a String

Another API to Master

The String: A Place for Your Lines, Characters, and Bytes

In the Wild

Staying Out of Trouble

Wrapping Up

## **Chapter 5: Find the Right String with Regular Expressions**

Matching One Character at a Time

Sets, Ranges, and Alternatives

The Regular Expression Star

Regular Expressions in Ruby

Beginnings and Endings

In the Wild

Staying Out of Trouble

Wrapping Up

## **Chapter 6: Use Symbols to Stand for Something**

The Two Faces of Strings

# Table of Contents

Not Quite a String

Optimized to Stand for Something

In the Wild

Staying Out of Trouble

Wrapping Up

## Chapter 7: Treat Everything Like an ObjectBecause Everything Is

A Quick Review of Classes, Instances, and Methods

Objects All the Way Down

The Importance of Being an Object

Public, Private, and Protected

In the Wild

Staying Out of Trouble

Wrapping Up

## Chapter 8: Embrace Dynamic Typing

Shorter Programs, But Not the Way You Think

Extreme Decoupling

Required Ceremony Versus Programmer-Driven Clarity

Staying Out of Trouble

In the Wild

Wrapping Up

## Chapter 9: Write Specs!

Test::Unit: When Your Documents Just Have to Work

A Plethora of Assertions

Dont Test It, Spec It!

A Tidy Spec Is a Readable Spec

Easy Stubs

. . . And Easy Mocks

# **Table of Contents**

In the Wild

Staying Out of Trouble

Wrapping Up

## **PART II: Classes, Modules, and Blocks**

### **Chapter 10: Construct Your Classes from Short, Focused Methods**

Compressing Specifications

Composing Methods for Humans

Composing Ruby Methods

One Way Out?

Staying Out of Trouble

In the Wild

Wrapping Up

### **Chapter 11: Define Operators Respectfully**

Defining Operators in Ruby

A Sampling of Operators

Operating Across Classes

Staying Out of Trouble

In the Wild

Wrapping Up

### **Chapter 12: Create Classes That Understand Equality**

An Identifier for Your Documents

An Embarrassment of Equality

Double Equals for Everyday Use

Broadening the Appeal of the == Method

Well-Behaved Equality

Triple Equals for Case Statements

Hash Tables and the eql? Method

# **Table of Contents**

Building a Well-Behaved Hash Key

Staying Out of Trouble

In the Wild

Wrapping Up

## **Chapter 13: Get the Behavior You Need with Singleton and Class Methods**

A Stubby Puzzle

A Hidden, but Real Class

Class Methods: Singletons in Plain Sight

In the Wild

Staying Out of Trouble

Wrapping Up

## **Chapter 14: Use Class Instance Variables**

A Quick Review of Class Variables

Wandering Variables

Getting Control of the Data in Your Class

Class Instance Variables and Subclasses

Adding Some Convenience to Your Class Instance Variables

In the Wild

Staying Out of Trouble

Wrapping Up

## **Chapter 15: Use Modules as Name Spaces**

A Place for Your Stuff, with a Name

A Home for Those Utility Methods

Building Modules a Little at a Time

Treat Modules Like the Objects That They Are

Staying Out of Trouble

In the Wild

# **Table of Contents**

Wrapping Up

## **Chapter 16: Use Modules as Mixins**

Better Books with Modules

Mixin Modules to the Rescue

Extending a Module

Staying Out of Trouble

In the Wild

Wrapping Up

## **Chapter 17: Use Blocks to Iterate**

A Quick Review of Code Blocks

One Word after Another

As Many Iterators as You Like

Iterating over the Ethereal

Enumerable: Your Iterator on Steroids

Staying Out of Trouble

In the Wild

Wrapping Up

## **Chapter 18: Execute Around with a Block**

Add a Little Logging

When It Absolutely Must Happen

Setting Up Objects with an Initialization Block

Dragging Your Scope along with the Block

Carrying the Answers Back

Staying Out of Trouble

In the Wild

Wrapping Up

## **Chapter 19: Save Blocks to Execute Later**

Explicit Blocks



# **Table of Contents**

The Call Back Problem  
Banking Blocks  
Saving Code Blocks for Lazy Initialization  
Instant Block Objects  
Staying Out of Trouble  
In the Wild  
Wrapping Up

## **PART III: Metaprogramming**

### **Chapter 20: Use Hooks to Keep Your Program Informed**

Waking Up to a New Subclass  
Modules Want To Be Heard Too  
Knowing When Your Time Is Up  
. . . And a Cast of Thousands  
Staying Out of Trouble  
In the Wild  
Wrapping Up

### **Chapter 21: Use `method_missing` for Flexible Error Handling**

Meeting Those Missing Methods  
Handling Document Errors  
Coping with Constants  
In the Wild  
Staying Out of Trouble  
Wrapping Up

### **Chapter 22: Use `method_missing` for Delegation**

The Promise and Pain of Delegation  
The Trouble with Old-Fashioned Delegation  
The `method_missing` Method to the Rescue

# Table of Contents

More Discriminating Delegation

Staying Out of Trouble

In the Wild

Wrapping Up

## Chapter 23: Use `method_missing` to Build Flexible APIs

Building Form Letters One Word at a Time

Magic Methods from `method_missing`

It's the Users That Count All of Them

Staying Out of Trouble

In the Wild

Wrapping Up

## Chapter 24: Update Existing Classes with Monkey

### Patching

Wide-Open Classes

Fixing a Broken Class

Improving Existing Classes

Renaming Methods with `alias_method`

Do Anything to Any Class, Anytime

In the Wild

Staying Out of Trouble

Wrapping Up

## Chapter 25: Create Self-Modifying Classes

Open Classes, Again

Put Programming Logic in Your Classes

Class Methods That Change Their Class

In the Wild

Staying Out of Trouble

Wrapping Up

# **Table of Contents**

## **Chapter 26: Create Classes That Modify Their Subclasses**

- A Document of Paragraphs
- Subclassing Is (Sometimes) Hard to Do
- Class Methods That Build Instance Methods
- Better Method Creation with `define_method`
- The Modification Sky Is the Limit
- In the Wild
- Staying Out of Trouble
- Wrapping Up

## **PART IV: Pulling It All Together**

### **Chapter 27: Invent Internal DSLs**

- Little Languages for Big Problems
- Dealing with XML
- Stepping Over the DSL Line
- Pulling Out All the Stops
- In the Wild
- Staying Out of Trouble
- Wrapping Up

### **Chapter 28: Build External DSLs for Flexible Syntax**

- The Trouble with the Ripper
- Internal Is Not the Only DSL
- Regular Expressions for Heavier Parsing
- Treetop for Really Big Jobs
- Staying Out of Trouble
- In the Wild
- Wrapping Up

### **Chapter 29: Package Your Programs as Gems**

# **Table of Contents**

Consuming Gems

Gem Versions

The Nuts and Bolts of Gems

Building a Gem

Uploading Your Gem to a Repository

Automating Gem Creation

In the Wild

Staying Out of Trouble

Wrapping Up

## **Chapter 30: Know Your Ruby Implementation**

A Fistful of Rubies

MRI: An Enlightening Experience for the C Programmer

YARV: MRI with a Byte Code Turbocharger

JRuby: Bending the J in the JVM

Rubinius

In the Wild

Staying Out of Trouble

Wrapping Up

## **Chapter 31: Keep an Open Mind to Go with Those Open Classes**

## **Appendix: Going Further**

## **Index**