

# Contents

---

*Foreword* xix

*Preface* xxi

*Acknowledgments* xxv

*About the Author* xxvii

## **PART I: The Basics** 1

### **Chapter 1: Write Code That Looks Like Ruby** 3

The Very Basic Basics 4

Go Easy on the Comments 6

Camels for Classes, Snakes Everywhere Else 8

Parentheses Are Optional but Are Occasionally Forbidden 9

Folding Up Those Lines 10

Folding Up Those Code Blocks 11

Staying Out of Trouble 12

In the Wild 13

Wrapping Up 15

### **Chapter 2: Choose the Right Control Structure** 17

If, Unless, While, and Until 17

Use the Modifier Forms Where Appropriate 19

Use each, Not for 20

A Case of Programming Logic 21

Staying Out of Trouble 23

In the Wild 25

Wrapping Up 27

### **Chapter 3: Take Advantage of Ruby's Smart Collections 29**

Literal Shortcuts 29

Instant Arrays and Hashes from Method Calls 30

Running Through Your Collection 33

Beware the Bang! 36

Rely on the Order of Your Hashes 38

In the Wild 38

Staying Out of Trouble 40

Wrapping Up 42

### **Chapter 4: Take Advantage of Ruby's Smart Strings 43**

Coming Up with a String 44

Another API to Master 47

The String: A Place for Your Lines, Characters, and Bytes 49

In the Wild 50

Staying Out of Trouble 51

Wrapping Up 52

### **Chapter 5: Find the Right String with Regular Expressions 53**

Matching One Character at a Time 54

Sets, Ranges, and Alternatives 55

The Regular Expression Star 57

Regular Expressions in Ruby 58

Beginnings and Endings 60

In the Wild 62

Staying Out of Trouble 63

Wrapping Up 64

### **Chapter 6: Use Symbols to Stand for Something 65**

The Two Faces of Strings 65

Not Quite a String 66

Optimized to Stand for Something 67

- In the Wild 69
- Staying Out of Trouble 70
- Wrapping Up 71

## **Chapter 7: Treat Everything Like an Object—Because Everything Is 73**

- A Quick Review of Classes, Instances, and Methods 74
- Objects All the Way Down 76
- The Importance of Being an Object 77
- Public, Private, and Protected 79
- In the Wild 81
- Staying Out of Trouble 82
- Wrapping Up 84

## **Chapter 8: Embrace Dynamic Typing 85**

- Shorter Programs, But Not the Way You Think 85
- Extreme Decoupling 89
- Required Ceremony Versus Programmer-Driven Clarity 92
- Staying Out of Trouble 93
- In the Wild 94
- Wrapping Up 96

## **Chapter 9: Write Specs! 97**

- Test::Unit: When Your Documents Just Have to Work 98
- A Plethora of Assertions 101
- Don't Test It, Spec It! 101
- A Tidy Spec Is a Readable Spec 104
- Easy Stubs 105
- ... And Easy Mocks 107
- In the Wild 108
- Staying Out of Trouble 110
- Wrapping Up 113

## **PART II: Classes, Modules, and Blocks 115**

### **Chapter 10: Construct Your Classes from Short, Focused Methods 117**

- Compressing Specifications 117
- Composing Methods for Humans 121

Composing Ruby Methods	122
One Way Out?	123
Staying Out of Trouble	126
In the Wild	127
Wrapping Up	128
<b>Chapter 11: Define Operators Respectfully</b>	<b>129</b>
Defining Operators in Ruby	129
A Sampling of Operators	131
Operating Across Classes	134
Staying Out of Trouble	135
In the Wild	137
Wrapping Up	139
<b>Chapter 12: Create Classes That Understand Equality</b>	<b>141</b>
An Identifier for Your Documents	141
An Embarrassment of Equality	142
Double Equals for Everyday Use	143
Broadening the Appeal of the == Method	145
Well-Behaved Equality	146
Triple Equals for Case Statements	149
Hash Tables and the eql? Method	150
Building a Well-Behaved Hash Key	152
Staying Out of Trouble	153
In the Wild	154
Wrapping Up	156
<b>Chapter 13: Get the Behavior You Need with Singleton and Class Methods</b>	<b>157</b>
A Stubby Puzzle	158
A Hidden, but Real Class	160
Class Methods: Singletons in Plain Sight	162
In the Wild	164
Staying Out of Trouble	165
Wrapping Up	167

**Chapter 14: Use Class Instance Variables 169**

- A Quick Review of Class Variables 169
- Wandering Variables 171
- Getting Control of the Data in Your Class 174
- Class Instance Variables and Subclasses 175
- Adding Some Convenience to Your Class Instance Variables 176
- In the Wild 177
- Staying Out of Trouble 179
- Wrapping Up 179

**Chapter 15: Use Modules as Name Spaces 181**

- A Place for Your Stuff, with a Name 181
- A Home for Those Utility Methods 184
- Building Modules a Little at a Time 185
- Treat Modules Like the Objects That They Are 186
- Staying Out of Trouble 189
- In the Wild 190
- Wrapping Up 191

**Chapter 16: Use Modules as Mixins 193**

- Better Books with Modules 193
- Mixin Modules to the Rescue 195
- Extending a Module 197
- Staying Out of Trouble 198
- In the Wild 202
- Wrapping Up 205

**Chapter 17: Use Blocks to Iterate 207**

- A Quick Review of Code Blocks 207
- One Word after Another 209
- As Many Iterators as You Like 210
- Iterating over the Ethereal 211
- Enumerable: Your Iterator on Steroids 213
- Staying Out of Trouble 215
- In the Wild 217
- Wrapping Up 218

**Chapter 18: Execute Around with a Block 219**

- Add a Little Logging 219
- When It Absolutely Must Happen 224
- Setting Up Objects with an Initialization Block 225
- Dragging Your Scope along with the Block 225
- Carrying the Answers Back 227
- Staying Out of Trouble 228
- In the Wild 229
- Wrapping Up 231

**Chapter 19: Save Blocks to Execute Later 233**

- Explicit Blocks 233
- The Call Back Problem 234
- Banking Blocks 236
- Saving Code Blocks for Lazy Initialization 237
- Instant Block Objects 239
- Staying Out of Trouble 240
- In the Wild 243
- Wrapping Up 244

**PART III: Metaprogramming 247****Chapter 20: Use Hooks to Keep Your Program Informed 249**

- Waking Up to a New Subclass 250
- Modules Want To Be Heard Too 253
- Knowing When Your Time Is Up 255
- ... And a Cast of Thousands 256
- Staying Out of Trouble 257
- In the Wild 259
- Wrapping Up 261

**Chapter 21: Use `method_missing` for Flexible Error Handling 263**

- Meeting Those Missing Methods 264
- Handling Document Errors 266
- Coping with Constants 267
- In the Wild 268

Staying Out of Trouble 270

Wrapping Up 271

## **Chapter 22: Use `method_missing` for Delegation 273**

The Promise and Pain of Delegation 274

The Trouble with Old-Fashioned Delegation 275

The `method_missing` Method to the Rescue 277

More Discriminating Delegation 278

Staying Out of Trouble 279

In the Wild 281

Wrapping Up 283

## **Chapter 23: Use `method_missing` to Build Flexible APIs 285**

Building Form Letters One Word at a Time 286

Magic Methods from `method_missing` 287

It's the Users That Count—All of Them 289

Staying Out of Trouble 289

In the Wild 290

Wrapping Up 292

## **Chapter 24: Update Existing Classes with Monkey Patching 293**

Wide-Open Classes 294

Fixing a Broken Class 295

Improving Existing Classes 296

Renaming Methods with `alias_method` 297

Do Anything to Any Class, Anytime 299

In the Wild 299

Staying Out of Trouble 303

Wrapping Up 303

## **Chapter 25: Create Self-Modifying Classes 305**

Open Classes, Again 305

Put Programming Logic in Your Classes 308

Class Methods That Change Their Class 309

In the Wild 310

Staying Out of Trouble	314
Wrapping Up	315
<b>Chapter 26: Create Classes That Modify Their Subclasses</b>	<b>317</b>
A Document of Paragraphs	317
Subclassing Is (Sometimes) Hard to Do	319
Class Methods That Build Instance Methods	321
Better Method Creation with <code>define_method</code>	324
The Modification Sky Is the Limit	324
In the Wild	327
Staying Out of Trouble	330
Wrapping Up	332
<b>PART IV: Pulling It All Together</b>	<b>333</b>
<b>Chapter 27: Invent Internal DSLs</b>	<b>335</b>
Little Languages for Big Problems	335
Dealing with XML	336
Stepping Over the DSL Line	341
Pulling Out All the Stops	344
In the Wild	345
Staying Out of Trouble	347
Wrapping Up	349
<b>Chapter 28: Build External DSLs for Flexible Syntax</b>	<b>351</b>
The Trouble with the Ripper	352
Internal Is Not the Only DSL	353
Regular Expressions for Heavier Parsing	356
Treetop for Really Big Jobs	358
Staying Out of Trouble	360
In the Wild	362
Wrapping Up	364
<b>Chapter 29: Package Your Programs as Gems</b>	<b>367</b>
Consuming Gems	367
Gem Versions	368



The Nuts and Bolts of Gems	369
Building a Gem	370
Uploading Your Gem to a Repository	374
Automating Gem Creation	375
In the Wild	376
Staying Out of Trouble	377
Wrapping Up	380

### **Chapter 30: Know Your Ruby Implementation 381**

A Fistful of Rubies	381
MRI: An Enlightening Experience for the C Programmer	382
YARV: MRI with a Byte Code Turbocharger	385
JRuby: Bending the “J” in the JVM	387
Rubinius	388
In the Wild	389
Staying Out of Trouble	389
Wrapping Up	390

### **Chapter 31: Keep an Open Mind to Go with Those Open Classes 391**

### **Appendix: Going Further 393**

<i>Index</i>	397
--------------	-----