

*Increase Your Productivity—Write Better Code*



# Ruby Best Practices

**O'REILLY®**

*Gregory T. Brown*  
*Foreword by Yukihiro Matsumoto*

# Table of Contents

<b>Foreword</b> .....	ix
<b>Preface</b> .....	xi
<b>1. Driving Code Through Tests</b> .....	<b>1</b>
A Quick Note on Testing Frameworks	2
<i>Designing for Testability</i>	2
Testing Fundamentals	10
Well-Focused Examples	10
Testing Exceptions	11
Run the Whole Suite at Once	13
Advanced Testing Techniques	14
Using Mocks and Stubs	14
Testing Complex Output	22
Keeping Things Organized	26
Embedding Tests in Library Files	27
Test Helpers	27
Custom Assertions	29
Conclusions	30
<b>2. Designing Beautiful APIs</b> .....	<b>31</b>
Designing for Convenience: Ruptor's Table() feature	31
Ruby's Secret Power: Flexible Argument Processing	35
Standard Ordinal Arguments	36
Ordinal Arguments with Optional Parameters	36
Pseudo-Keyword Arguments	37
Treating Arguments As an Array	38
Ruby's Other Secret Power: Code Blocks	40
Working with Enumerable	41
Using Blocks to Abstract Pre- and Postprocessing	43
Blocks As Dynamic Callbacks	45
<i>Blocks for Interface Simplification</i>	47

Avoiding Surprises	48
Use attr_reader, attr_writer, and attr_accessor	48
<i>Understand What method? and method! Mean</i>	50
Make Use of Custom Operators	53
Conclusions	55
<b>3. Mastering the Dynamic Toolkit .....</b>	<b>57</b>
BlankSlate: A BasicObject on Steroids	57
Building Flexible Interfaces	62
Making instance_eval() Optional	63
Handling Messages with method_missing() and send()	65
Dual-Purpose Accessors	69
Implementing Per-Object Behavior	70
Extending and Modifying Preexisting Code	74
Adding New Functionality	75
Modification via Aliasing	79
Per-Object Modification	81
Building Classes and Modules Programmatically	84
Registering Hooks and Callbacks	88
Detecting Newly Added Functionality	89
Tracking Inheritance	91
Tracking Mixins	93
Conclusions	96
<b>4. Text Processing and File Management .....</b>	<b>99</b>
Line-Based File Processing with State Tracking	99
Regular Expressions	103
Don't Work Too Hard	105
Anchors Are Your Friends	105
Use Caution When Working with Quantifiers	106
Working with Files	109
Using Pathname and FileUtils	109
The tempfile Standard Library	112
Automatic Temporary Directory Handling	113
Collision Avoidance	113
Same Old I/O Operations	114
Automatic Unlinking	114
Text-Processing Strategies	115
<i>Advanced Line Processing</i>	116
Atomic Saves	118
Conclusions	120

<b>5. Functional Programming Techniques</b> .....	<b>121</b>
Laziness Can Be a Virtue (A Look at lazy.rb)	121
Minimizing Mutable State and Reducing Side Effects	129
Modular Code Organization	133
Memoization	138
Infinite Lists	145
Higher-Order Procedures	149
Conclusions	152
<b>6. When Things Go Wrong</b> .....	<b>153</b>
A Process for Debugging Ruby Code	153
Capturing the Essence of a Defect	157
Scrutinizing Your Code	160
Utilizing Reflection	160
Improving inspect Output	162
Finding Needles in a Haystack	166
Working with Logger	168
Conclusions	176
<b>7. Reducing Cultural Barriers</b> .....	<b>177</b>
m17n by Example: A Look at Ruby's CSV Standard Library	178
Portable m17n Through UTF-8 Transcoding	182
Source Encodings	183
Working with Files	183
Transcoding User Input in an Organized Fashion	185
m17n in Standalone Scripts	188
Inferring Encodings from Locale	189
Customizing Encoding Defaults	191
m17n-Safe Low-Level Text Processing	193
Localizing Your Code	195
Conclusions	204
<b>8. Skillful Project Maintenance</b> .....	<b>205</b>
Exploring a Well-Organized Ruby Project (Haml)	205
Conventions to Know About	210
What Goes in a README	211
Laying Out Your Library	213
Executables	216
Tests	216
Examples	217
API Documentation via RDoc	219
Basic Documentation Techniques and Guidelines	220
Controlling Output with RDoc Directives	222

The RubyGems Package Manager	227
Writing a Gem::Specification	228
Working with Dependencies	231
Rake: Ruby's Built-in Build Utility	234
Conclusions	237
<b>A. Writing Backward-Compatible Code</b> .....	<b>239</b>
<b>B. Leveraging Ruby's Standard Library</b> .....	<b>251</b>
<b>C. Ruby Worst Practices</b> .....	<b>283</b>
<b>Index</b> .....	<b>299</b>