

The
Pragmatic
Programmers

Eloquent Ruby

Second Edition

Facets of Ruby
Series editor: Noel Rappin

Russ Olsen
edited by Michael Swaine



This extract shows the online version of this title, and may contain features (such as hyperlinks and colors) that are not available in the print version.

For more information, or to purchase a paperback or ebook copy, please visit <https://www.pragprog.com>.

Copyright © The Pragmatic Programmers, LLC.

Contents

<u>Change History</u>		?
<u>Acknowledgments</u>		?
<u>Preface</u>		?

Part I — The Basics

1.	<u>Write Code That Looks Like Ruby</u>		?
	<u>Starting With the Basics</u>		?
	<u>Going Easy on the Comments</u>		?
	<u>Choosing Camels and Snakes</u>		?
	<u>Adding and Omitting Parentheses</u>		?
	<u>Folding Up Those Lines</u>		?
	<u>Folding Up Those Code Blocks</u>		?
	<u>Staying Out of Trouble</u>		?
	<u>In the Wild</u>		?
	<u>Wrapping Up</u>		?
2.	<u>Choose the Right Control Structure</u>		?
	<u>Controlling the Flow of Your Programs</u>		?
	<u>Use the Modifier Forms Where Appropriate</u>		?
	<u>Using each, Avoiding for</u>		?
	<u>Choosing Between Multiple Alternatives</u>		?
	<u>Tightening Up Your Logic</u>		?
	<u>Staying Out of Trouble</u>		?

	In the Wild	?
	Wrapping Up	?
3.	Take Advantage of Ruby's Smart Strings	?
	Coming Up with a String	?
	Mastering the String API	?
	Running Through Your Lines, Characters, and Bytes	?
	In the Wild	?
	Staying Out of Trouble	?
	Wrapping Up	?
4.	Find the Right String with Regular Expressions	?
	Matching One Character at a Time	?
	Discovering Sets, Ranges, and Alternatives	?
	Embracing the Regular Expression Star	?
	Using Regular Expressions in Ruby	?
	Finding the Beginning and the End	?
	Capturing Your Matches	?
	In the Wild	?
	Staying Out of Trouble	?
	Wrapping Up	?
5.	Use Symbols to Stand for Something	?
	Seeing the Two Faces of a String	?
	Optimizing for Symbolism	?
	Applying the Rory Test	?
	In the Wild	?
	Staying Out of Trouble	?
	Wrapping Up	?
6.	Take Advantage of Ruby's Smart Collections	?
	Applying the Literal Shortcuts	?
	Making Instant Arrays and Hashes from Method Calls	?
	Running Through Your Collection	?
	Beware the Bang!	?
	Relying on the Order of Your Hashes	?
	In the Wild	?

	Staying Out of Trouble	?
	Wrapping Up	?
7.	Dig Into Your Data With Pattern Matching	?
	Matching Data Like Strings	?
	Pattern Matching Arrays	?
	Capturing Values From Your Pattern Matching	?
	Pattern Matching Hashes	?
	Inserting Values Into Your Patterns	?
	In the Wild	?
	Staying Out of Trouble	?
	Wrapping Up	?
8.	Treat Everything Like an Object—Because Everything Is	?
	Reviewing the Object Basics	?
	Seeing Objects Everywhere	?
	Being an Object	?
	In the Wild	?
	Staying Out of Trouble	?
	Wrapping Up	?
9.	Understand Dynamic Typing	?
	Related by Ability	?
	Extreme Decoupling	?
	Dealing With the Costs of Dynamic Typing	?
	Static Typing with RBS and Sorbet	?
	Staying Out of Trouble	?
	In the Wild	?
	Wrapping Up	?
10.	Write Tests!	?
	Test Your Code With Minitest	?
	RSpec: Tests Should be About the Code, Not the Test	?
	Setting Expectations And Common Values	?
	Creating the Illusion of Infrastructure	?
	Creating Objects With An Opinion	?
	Minitest Specs Back	?

Weighing the Testing Alternatives	?
In the Wild	?
Staying Out of Trouble	?
Wrapping Up	?

Part II — Classes, Modules and Blocks

11. Construct Your Classes from Short, Focused Methods	?
Compressing Specifications	?
Composing Methods for Humans	?
Composing Ruby Methods	?
One Way Out?	?
Controlling Access to Your Objects	?
Staying Out of Trouble	?
In the Wild	?
Wrapping Up	?
12. Define Operators Respectfully	?
Defining Operators in Ruby	?
A Sampling of Operators	?
Operating Across Classes	?
Staying Out of Trouble	?
In the Wild	?
Wrapping Up	?
13. Create Classes That Understand Equality	?
Enumerating Ruby Equality	?
Double Equals for Everyday Use	?
Broadening the Appeal of the == Method	?
Working Towards Well-Behaved Equality	?
Writing Triple Equals	?
Understanding eql?	?
Building a Well-Behaved Hash Key	?
Staying Out of Trouble	?
In the Wild	?
Wrapping Up	?

14. Get the Behavior You Need with Singleton and Class	
Methods	?
A Double Puzzle	?
A Hidden, but Real Class	?
Class Methods: Singletons in Plain Sight	?
In the Wild	?
Staying Out of Trouble	?
Wrapping Up	?
15. 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	?
Stashing Your Class Data in Constants	?
In the Wild	?
Staying Out of Trouble	?
Wrapping Up	?
16. 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	?
Wrapping Up	?
17. 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	?

18. Use Blocks to Iterate	?
A Quick Review of Code Blocks	?
One Word after Another	?
As Many Iterators as You Like	?
Iterating over the Ethereal	?
Enhance Your Collections With Enumerable	?
Make Pseudo-Collections With Enumerator	?
Staying Out of Trouble	?
In the Wild	?
Wrapping Up	?
19. 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	?
20. Save Blocks to Execute Later	?
Making Your Blocks Explicit	?
Calling Back From Documents	?
Banking Blocks	?
Saving Code Blocks for Lazy Initialization	?
Creating Instant Block Objects	?
Staying Out of Trouble	?
In the Wild	?
Wrapping Up	?

Part III — Metaprogramming

21. 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	?

Swimming in the Sea of TracePoint	?
Staying Out of Trouble	?
In the Wild	?
Wrapping Up	?
22. Use method_missing for Delegation	?
Embracing Delegation	?
Feeling the Pain of Old-Fashioned Delegation	?
Meeting Those Missing Methods	?
The method_missing Method to the Rescue	?
Creating More Discriminating Delegation	?
Staying Out of Trouble	?
In the Wild	?
Wrapping Up	?
23. Use method_missing to Build Flexible APIs	?
Building Spam One Word at a Time	?
Creating Magic Methods from method_missing	?
It's the Users That Count—All of Them	?
Staying Out of Trouble	?
In the Wild	?
Wrapping Up	?
24. Update Existing Classes with Monkey Patching	?
Modifying 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	?
25. Create Self-Modifying Classes	?
Thinking About Open Classes, Again	?
Putting Programming Logic in Your Classes	?
Modifying a Class With Class Methods	?

	Staying Out of Trouble	?
	In the Wild	?
	Wrapping Up	?
26.	Create Classes That Modify Their Subclasses	?
	Enhancing Document	?
	Subclassing Is (Sometimes) Hard to Do	?
	Using Class Methods to Build Instance Methods	?
	Better Method Creation With <code>define_method</code>	?
	The Modification Sky Is the Limit	?
	In the Wild	?
	Staying Out of Trouble	?
	Wrapping Up	?

Part IV — Going Further

27.	Invent Internal DSLs	?
	Creating Little Languages for Big Problems	?
	So Many Files...	?
	Introducing Rawk	?
	Stepping Over the DSL Line	?
	In the Wild	?
	Staying Out of Trouble	?
	Wrapping Up	?
28.	Be Concurrent	?
29.	Package Your Programs as Gems	?
30.	Know Your Ruby Implementation	?
31.	Remember, Guidelines Not Rules	?