contents

preface   xix
preface to the first edition   xxi
acknowledgments   xxiii
about this book   xxv
about the cover illustration   xxx

PART 1   RUBY FOUNDATIONS 1

1 *Bootstrapping your Ruby literacy   3*

1.1   Basic Ruby language literacy   4

A Ruby syntax survival kit   5 ▪ The variety of Ruby identifiers   7
Method calls, messages, and Ruby objects   8 ▪ Writing and saving a simple program   10 ▪ Feeding the program to Ruby   11
Keyboard and file I/O   13

1.2   Anatomy of the Ruby installation   15

The Ruby standard library subdirectory (RbConfig::CONFIG[rubylibdir])   17 ▪ The C extensions directory (RbConfig::CONFIG[archdir])   17 ▪ The site_ruby (RbConfig::CONFIG[sitedir]) and vendor_ruby (RbConfig::CONFIG[vendordir]) directories   17
The gems directory   18

1.3 Ruby extensions and programming libraries 18

Loading external files and extensions 18 ▪ "Load"-ing a file in the default load path 19 ▪ "Require"-ing a feature 21 require_relative 22

1.4 Out-of-the-box Ruby tools and applications 22

Interpreter command-line switches 23 ▪ A closer look at interactive Ruby interpretation with irb 26 ▪ ri and RDoc 28 The rake task-management utility 29 ▪ Installing packages with the gem command 31

1.5 Summary 33

2 ***Objects, methods, and local variables 34***

2.1 Talking to objects 35

Ruby and object orientation 35 ▪ Creating a generic object 36 Methods that take arguments 38 ▪ The return value of a method 39

2.2 Crafting an object: The behavior of a ticket 40

The ticket object, behavior first 40 ▪ Querying the ticket object 41 Shortening the ticket code via string interpolation 42 Ticket availability: Expressing Boolean state in a method 42

2.3 The innate behaviors of an object 44

Identifying objects uniquely with the object_id method 45 Querying an object's abilities with the respond_to? method 46 Sending messages to objects with the send method 47

2.4 A close look at method arguments 48

Required and optional arguments 48 ▪ Default values for arguments 49 ▪ Order of parameters and arguments 50 What you can't do in argument lists 52

2.5 Local variables and variable assignment 54

Variables, objects, and references 55 ▪ References in variable assignment and reassignment 57 ▪ References and method arguments 58 ▪ Local variables and the things that look like them 59

2.6 Summary 60

3 ***Organizing objects with classes 62***

3.1 Classes and instances 63

Instance methods 64 ▪ Overriding methods 64 Reopening classes 65

3.2 Instance variables and object state 67

Initializing an object with state 68

3.3 Setter methods 70

The equal sign (=) in method names 70 ▪ Syntactic sugar for assignment-like methods 72 ▪ Setter methods unleashed 72

3.4 Attributes and the attr_* method family 74

Automating the creation of attributes 75 ▪ Summary of attr_* methods 77

3.5 Inheritance and the Ruby class hierarchy 77

Single inheritance: One to a customer 79 ▪ Object ancestry and the not-so-missing link: The Object class 79 ▪ El Viejo's older brother: BasicObject 80

3.6 Classes as objects and message receivers 81

Creating class objects 81 ▪ How class objects call methods 82 A singleton method by any other name… 83 ▪ When, and why, to write a class method 84 ▪ Class methods vs. instance methods 85

3.7 Constants up close 86

Basic use of constants 87 ▪ Reassigning vs. modifying constants 88

3.8 Nature vs. nurture in Ruby objects 89

3.9 Summary 91

4 ***Modules and program organization 92***

4.1 Basics of module creation and use 93

A module encapsulating "stacklikeness" 94 ▪ Mixing a module into a class 96 ▪ Using the module further 98

4.2 Modules, classes, and method lookup 99

Illustrating the basics of method lookup 100 ▪ Defining the same method more than once 103 ▪ How prepend works 105 The rules of method lookup summarized 106 ▪ Going up the method search path with super 107

4.3 The method_missing method 109

Combining method_missing and super 110

4.4 Class/module design and naming 114

Mix-ins and/or inheritance 114 ▪ Nesting modules and classes 116

4.5 Summary 117

5 *The default object (self), scope, and visibility* **119**

5.1 Understanding self, the current/default object 120

Who gets to be self, and where 120 ▪ The top-level self object 122 Self inside class, module, and method definitions 123 ▪ Self as the default receiver of messages 126 ▪ Resolving instance variables through self 128

5.2 Determining scope 129

Global scope and global variables 130 ▪ Local scope 132 The interaction between local scope and self 135 ▪ Scope and resolution of constants 136 ▪ Class variable syntax, scope, and visibility 138

5.3 Deploying method-access rules 144

Private methods 145 ▪ Protected methods 148

5.4 Writing and using top-level methods 149

Defining a top-level method 149 ▪ Predefined (built-in) top-level methods 150

5.5 Summary 151

6 *Control-flow techniques* **152**

6.1 Conditional code execution 153

The if keyword and friends 153 ▪ Assignment syntax in condition bodies and tests 157 ▪ case statements 160

6.2 Repeating actions with loops 164

Unconditional looping with the loop method 165 Conditional looping with the while and until keywords 166 Looping based on a list of values 168

6.3 Iterators and code blocks 168

The ingredients of iteration 168 ▪ Iteration, home-style 169 The anatomy of a method call 170 ▪ Curly braces vs. do/end in code block syntax 170 ▪ Implementing times 171 The importance of being each 173 ▪ From each to map 175 Block parameters and variable scope 176

6.4 Error handling and exceptions 179

Raising and rescuing exceptions 180 ▪ The rescue keyword to the rescue! 181 ▪ Raising exceptions explicitly 182 Capturing an exception in a rescue clause 183 ▪ The ensure clause 185 ▪ Creating your own exception classes 186

6.5 Summary 187

PART 2 BUILT-IN CLASSES AND MODULES 189

7 *Built-in essentials 191*

7.1 Ruby's literal constructors 192

7.2 Recurrent syntactic sugar 193

Defining operators by defining methods 194
Customizing unary operators 196

7.3 Bang (!) methods and "danger" 197

Destructive (receiver-changing) effects as danger 197
Destructiveness and "danger" vary independently 198

7.4 Built-in and custom to_* (conversion) methods 200

*String conversion: to_s 200 ▪ Array conversion with to_a and the * operator 202 ▪ Numerical conversion with to_i and to_f 203*
Role-playing to_* methods 205

7.5 Boolean states, Boolean objects, and nil 207

True and false as states 207 ▪ true and false as objects 209
The special object nil 211

7.6 Comparing two objects 212

Equality tests 212 ▪ Comparisons and the Comparable module 213

7.7 Inspecting object capabilities 215

Listing an object's methods 215 ▪ Querying class and module objects 216 ▪ Filtered and selected method lists 217

7.8 Summary 217

8 *Strings, symbols, and other scalar objects 219*

8.1 Working with strings 220

String notation 220 ▪ Basic string manipulation 224
Querying strings 228 ▪ String comparison and ordering 230
String transformation 231 ▪ String conversions 234
String encoding: A brief introduction 235

8.2 Symbols and their uses 237

Chief characteristics of symbols 237 ▪ Symbols and identifiers 238 ▪ Symbols in practice 240 ▪ Strings and symbols in comparison 242

8.3 Numerical objects 243

Numerical classes 244 ▪ Performing arithmetic operations 244

8.4 Times and dates 246

Instantiating date/time objects 246 ▪ Date/time query methods 249 ▪ Date/time formatting methods 249 Date/time conversion methods 251

8.5 Summary 252

9 ***Collection and container objects 254***

9.1 Arrays and hashes in comparison 255

9.2 Collection handling with arrays 256

Creating a new array 257 ▪ Inserting, retrieving, and removing array elements 261 ▪ Combining arrays with other arrays 264 Array transformations 265 ▪ Array querying 266

9.3 Hashes 267

Creating a new hash 268 ▪ Inserting, retrieving, and removing hash pairs 269 ▪ Specifying default hash values and behavior 271 ▪ Combining hashes with other hashes 272 Hash transformations 273 ▪ Hash querying 274 Hashes as final method arguments 275 ▪ A detour back to argument syntax: Named (keyword) arguments 276

9.4 Ranges 277

Creating a range 278 ▪ Range-inclusion logic 279

9.5 Sets 281

Set creation 281 ▪ Manipulating set elements 282 Subsets and supersets 284

9.6 Summary 285

10 ***Collections central: Enumerable and Enumerator 286***

10.1 Gaining enumerability through each 287

10.2 Enumerable Boolean queries 289

10.3 Enumerable searching and selecting 291

Getting the first match with find 292 ▪ Getting all matches with find_all (a.k.a. select) and reject 293 ▪ Selecting on threequal matches with grep 294 ▪ Organizing selection results with group_by and partition 295

10.4 Element-wise enumerable operations 297

The first method 297 ▪ The take and drop methods 298 The min and max methods 299

10.5 Relatives of each 300

reverse_each 300 ▪ The each_with_index method (and each.with_index) 301 ▪ The each_slice and each_cons methods 302 ▪ The cycle method 303 ▪ Enumerable reduction with inject 303

10.6 The map method 304

The return value of map 305 ▪ In-place mapping with map! 306

10.7 Strings as quasi-enumerables 306

10.8 Sorting enumerables 308

Where the Comparable module fits into enumerable sorting (or doesn't) 310 ▪ Defining sort-order logic with a block 310 Concise sorting with sort_by 311

10.9 Enumerators and the next dimension of enumerability 311

Creating enumerators with a code block 312 ▪ Attaching enumerators to other objects 314 ▪ Implicit creation of enumerators by blockless iterator calls 316

10.10 Enumerator semantics and uses 316

How to use an enumerator's each method 316 ▪ Protecting objects with enumerators 318 ▪ Fine-grained iteration with enumerators 320 ▪ Adding enumerability with an enumerator 320

10.11 Enumerator method chaining 322

Economizing on intermediate objects 322 ▪ Indexing enumerables with with_index 324 ▪ Exclusive-or operations on strings with enumerators 324

10.12 Lazy enumerators 326

FizzBuzz with a lazy enumerator 327

10.13 Summary 328

11 ***Regular expressions and regexp-based string operations 330***

11.1 What are regular expressions? 331

11.2 Writing regular expressions 331

Seeing patterns 331 ▪ Simple matching with literal regular expressions 332

11.3 Building a pattern in a regular expression 333

Literal characters in patterns 334 ▪ The dot wildcard character (.) 334 ▪ Character classes 334

11.4 Matching, substring captures, and MatchData 336

Capturing submatches with parentheses 336 ▪ Match success and failure 338 ▪ Two ways of getting the captures 339 Other MatchData information 341

11.5 Fine-tuning regular expressions with quantifiers, anchors, and modifiers 342

Constraining matches with quantifiers 342 ▪ Greedy (and non-greedy) quantifiers 344 ▪ Regular expression anchors and assertions 346 ▪ Modifiers 349

11.6 Converting strings and regular expressions to each other 350

String-to-regexp idioms 351 ▪ Going from a regular expression to a string 352

11.7 Common methods that use regular expressions 353

String#scan 353 ▪ String#split 355 ▪ sub/sub! and gsub/gsub! 356 ▪ Case equality and grep 357

11.8 Summary 359

12 ***File and I/O operations 360***

12.1 How Ruby's I/O system is put together 361

The IO class 361 ▪ IO objects as enumerables 362 STDIN, STDOUT, STDERR 363 ▪ A little more about keyboard input 364

12.2 Basic file operations 364

The basics of reading from files 365 ▪ Line-based file reading 365 ▪ Byte- and character-based file reading 366 Seeking and querying file position 367 ▪ Reading files with File class methods 368 ▪ Writing to files 369 ▪ Using blocks to scope file operations 370 ▪ File enumerability 371 ▪ File I/O exceptions and errors 372

12.3 Querying IO and File objects 373

Getting information from the File class and the FileTest module 373 Deriving file information with File::Stat 374

12.4 Directory manipulation with the Dir class 375

Reading a directory's entries 375 ▪ Directory manipulation and querying 378

12.5 File tools from the standard library 379

The FileUtils module 379 ▪ The Pathname class 381 The StringIO class 382 ▪ The open-uri library 384

12.6 Summary 384

PART 3 RUBY DYNAMICS 387

13 **Object individuation 389**

13.1 Where the singleton methods are: The singleton class 390

Dual determination through singleton classes 391 ▪ Examining and modifying a singleton class directly 392 ▪ Singleton classes on the method-lookup path 394 ▪ The singleton_class method 398 Class methods in (even more) depth 399

13.2 Modifying Ruby's core classes and modules 400

The risks of changing core functionality 401 ▪ Additive changes 405 ▪ Pass-through overrides 406 ▪ Per-object changes with extend 408 ▪ Using refinements to affect core behavior 411

13.3 BasicObject as ancestor and class 412

Using BasicObject 413 ▪ Implementing a subclass of BasicObject 414

13.4 Summary 416

14 **Callable and runnable objects 418**

14.1 Basic anonymous functions: The Proc class 419

Proc objects 419 ▪ Procs and blocks and how they differ 420 Block-proc conversions 422 ▪ Using Symbol#to_proc for conciseness 424 ▪ Procs as closures 426 ▪ Proc parameters and arguments 428

14.2 Creating functions with lambda and -> 428

14.3 Methods as objects 430

Capturing Method objects 430 ▪ The rationale for methods as objects 431

14.4 The eval family of methods 433

Executing arbitrary strings as code with eval 433 ▪ The dangers of eval 435 ▪ The instance_eval method 435 ▪ Using class_eval (a.k.a. module_eval) 437

14.5 Parallel execution with threads 438

Killing, stopping, and starting threads 440 ▪ A threaded date server 441 ▪ Writing a chat server using sockets and threads 443 Threads and variables 445 ▪ Manipulating thread keys 446

14.6 Issuing system commands from inside Ruby programs 449

The system method and backticks 449 ▪ Communicating with programs via open and popen3 451

14.7 Summary 454

15 ***Callbacks, hooks, and runtime introspection 456***

15.1 Callbacks and hooks 457

Intercepting unrecognized messages with method_missing 457 Trapping include and prepend operations 461 ▪ Intercepting extend 462 ▪ Intercepting inheritance with Class#inherited 464 The Module#const_missing method 465 ▪ The method_added and singleton_method_added methods 465

15.2 Interpreting object capability queries 467

Listing an object's non-private methods 468 ▪ Listing private and protected methods 470 ▪ Getting class and module instance methods 471 ▪ Listing objects' singleton methods 473

15.3 Introspection of variables and constants 474

Listing local and global variables 475
Listing instance variables 475

15.4 Tracing execution 476

Examining the stack trace with caller 476 ▪ Writing a tool for parsing stack traces 477

15.5 Callbacks and method inspection in practice 480

MicroTest background: MiniTest 480 ▪ Specifying and implementing MicroTest 482

15.6 Summary 485

index 487